applied sciences

MDPI

Article

A2C: Attention-Augmented Contrastive Learning for State Representation Extraction

Haoqiang Chen, Yadong Liu *, Zongtan Zhou and Ming Zhang

College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China; gelieleo@hnu.edu.cn (H.C.); narcz@nudt.edu.cn (Z.Z.); zhangming@nudt.edu.cn (M.Z.)

* Correspondence: liuyadong@nudt.edu.cn

Received: 21 July 2020; Accepted: 23 August 2020; Published: 26 August 2020

Abstract: Reinforcement learning (RL) faces a series of challenges, including learning efficiency and generalization. The state representation used to train RL is one of the important factors causing these challenges. In this paper, we explore providing a more efficient state representation for RL. Contrastive learning is used as the representation extraction method in our work. We propose an attention mechanism implementation and extend an existing contrastive learning method by embedding the attention mechanism. Finally an attention-augmented contrastive learning method called A2C is obtained. As a result, using the state representation from A2C, the robot achieves better learning efficiency and generalization than those using state-of-the-art representations. Moreover, our attention mechanism is proven to be able to calculate the correlation of arbitrary distance among pixels, which is conducive to capturing more accurate obstacle information. What is more, we remove the attention mechanism from A2C. It is shown that the rewards available for the attention-removed A2C are reduced by more than 70%, which indicates the important role of the attention mechanism.

Keywords: state representation learning; contrastive learning; reinforcement learning; attention mechanism; robot

1. Introduction

Reinforcement learning (RL) has achieved great success in various domains [1], but it requires a large amount of interaction with the environment and is difficult to generalize to new situations. State representation used to train RL plays an important role in the sample efficiency and generalization. In general, those low-dimensional and robust representations are easier to associate with the reward signal [2], and they can help agents obtain better generalization with less training episodes. In this paper, we explore providing a more efficient state representation for RL.

In vision-based RL, most existing work directly uses raw images (RGB usually) as the state representation [3–7]. Although RGB is efficient for accomplishing tasks, it results in two disadvantages for RL to some extent, i.e., sample inefficiency and poor generalization. RGB has a very powerful expression capability that can express the state of almost any visible scenes. However, it is this flexibility that makes its expression non-robust, because pixels in RGB are independent of each other, and it is difficult to maintain any robust features. For example, even if the camera is placed in the same place with the same perspective, the representations in the RGB space may be very different due to some changes of the environmental conditions such as light and object position.

To overcome the shortcomings of RGB, researchers have sought other ways to obtain more effective state representations. Currently, there are mainly two kinds of methods to improve the state representation. (1) The first is joint optimization of state representation and policy. This method generally embeds raw data processing mechanisms in the RL model and simultaneously optimizes it

with a policy network. A suitable example can be found in [3] where a pair of shared Siamese networks was used to output a joint representation of the current/target state, which was immediately taken into the policy network as input. The parameters of the Siamese network also participate in gradient back-propagation when the policy network is updated. It is shown that this joint representation can indeed improve the learning efficiency and generalization of agents. Another example comes from [8]: instead of using RGB directly, it extracts agents' neighbour objects like obstacles as nodes and embeds them into a graph [9] that is able to make the representation more compact. In fact, we tend to believe that the effectiveness of the method of joint optimization should be attributed more to the innovation of the network structure rather than the state representation, because the parameters of the raw data processing mechanism actually participate in the policy learning directly. In other words, the state representation used by this method is still RGB. (2) The second is to decouple the representation learning from policy learning and specifically optimize it for the state representation. Recently, deep learning [10] technology has become the mainstream way to obtain effective representations for agents. There were various neural network-based representation learning methods described in [2], and they can be divided into supervised and unsupervised. It has been attempted to use the state representations extracted from the supervised vision tasks, such as segmentation, edge detection, etc., to accomplish the robot navigation task [11]. Compared with RGB, these compact features are more robust, that is less affected by changes in ambient light and the scene. Therefore, it is conducive to improving the learning efficiency and generalization of agents. However, as labels are not always available, researchers seek to distil state representations through unsupervised learning, such as the generative adversarial network (GAN) [12–14], autoencoder [15], contrastive learning [16–18], etc. One common goal of these unsupervised methods is to compress data into a low-dimensional latent space and force the model to learn the most critical compression information with self-reliant tasks such as image reconstruction and similarity comparison. Compared with supervised, unsupervised learning can largely reduce labour costs and avoid prior supervision signal errors due to human cognitive deviations.

In general, most of the existing supervised and unsupervised visual representation learning methods use the convolutional neural network as the main component to extract features. However, the convolution kernel only has a local receptive field that has difficulty capturing long-range correlation among pixels. The deeper network is required to get a larger receptive field, but it may be difficult for optimizers to discover the parameters that carefully coordinate multiple layers.

In this paper, we explore providing a state representation learning method that can capture long-range dependencies among pixels. We put forward a feasible attention mechanism [19] implementation and extend the work of [17] by embedding the attention mechanism. As a result, an attention-augmented contrastive learning method called A2C is obtained. The reason why we chose to combine contrastive learning and the attention mechanism is that the training target of contrastive learning is to maximize the similarity of two pictures in the same augmented image pair, and whether the contrastive model can summarize the efficient information of an image is very important. Therefore, we argue that contrastive learning should break through the limitation of the distance among pixels and more fully query the correlation information among pixels. At the same time, the attention mechanism is capable of querying the dependencies of pixels directly without significantly deepening the depth of the model. Thus, both of them are put together in our work.

Our attention mechanism is helpful in extracting more accurate compressed vectors. Our contributions can be summarized as follows:

1. To the best of our knowledge, this is the first time to apply the attention mechanism to the contrastive learning for visual representation extraction.
2. We propose a feasible implementation of the attention mechanism.
3. The state representation distilled from A2C improves the sample efficiency and generalization of RL, which can be seen in our experiments.

The rest of the paper is organized as follows. Related works contributing to the realization of our ideas are given in Section 2. Section 3 gives our methodology. Experimental studies to verify the effectiveness of our methods are carried out in Section 4. Detailed analysis of the experiment results is given in Section 5. Finally, we draw our conclusions and look ahead to the future work in Section 6.

2. Related Works

In addition to solving from the perspective of state representation, there are currently other ways to solve the sample inefficiency and poor generalization of RL. We will introduce related works in this section.

2.1. Sample Efficiency

It is most intuitive to consider this issue from the perspective of engineering. A suitable example comes from [20]. Using multiple threads, it developed an asynchronous version of Actor-Critic [21], which has higher sample efficiency than the vanilla version. Reference [22] adopted a more effective method that directly employs the graphics processing unit (GPU) to accelerate the learning process. Another perspective to improve the sample efficiency is to reduce the search space of the state/action. A mainstream approach is to build an environment model for RL [23–30]. Compared with model-free RL, model-based RL tends to be more efficient because it can predict the future state and reward based on the model, which narrows the policy search space to some extent. For example, multi-agent cooperation can indeed benefit from the environment model. It is difficult for each individual agent to predict the behaviours of its opponents, and the model is able to provide related information. Reference [31] has succeeded in reducing the uncertainty of the environment by modelling opponents for each individual agent. What is more, Reference [32] optimized policy learning through trajectory optimization and obtained higher data efficiency. Even in [33], agents only needed about four minutes to learn a complex task like lock-stacking, which is unimaginable to be model-free. Imitation learning [34] is another feasible way to accelerate learning processing. Considering learning from zero can result in a large search space, a d imitation learning adopts the way of feeding expert's demonstration to the model to reduce the search difficulty. The training goal of imitation learning is to match the state-action trajectory distribution generated by the model with the expert trajectory distribution. A representative example can be seen in [35].

2.2. Generalization

There are mainly two ways to investigate the generalization of RL. One is similar to robust control, which tends to learn a set of fixed parameters to keep agents robust to disturbance, but this approach would sacrifice the performance of agents in the testing environments. Over distribution of environments, Reference [36] developed a policy that can maximize the conditional value of risk, which makes the agent sensitive to the risk, so it becomes more robust. Similar to that, Reference [37] proposed a robust variant of Actor-Critic through maximizing the minimum value of reward over all possible disturbances. Reference [38] employed an adversarial way to learn a robust policy. Another method to improve the generalization is more like adaptive control, which attempts to adjust the parameters of the model online to adapt to the certain disturbance. Meta learning is a representative example. Its training target is task-agnostic, so it has good adaptability to different tasks. For instance, Reference [39] sampled trajectories in real time as features to identify the environment, which was able to make the policy adaptive to various environments. Reference [40] used a large set of Markov decision process (MDP) to train the weights of the recurrent neural network. Thus, it could generate a specific MDP model to adapt specific tasks.

3. Methodology

The purpose of this paper is to learn an efficient state representation for RL. Firstly, contrastive learning is used as the main learning paradigm to learn representation. To make the contrastive learning

model able to calculate the dependencies among pixels at any distance without deepening the model, we embed the attention mechanism into the contrastive learning model. Secondly, the representation extracted from A2C is taken into an RL framework as the input to learn an efficient policy.

3.1. State Representation Learning

The reason why contrastive learning was chosen as the learning paradigm to learn the state representation is that it is capable of making those similar scenes closer in the low-dimensional latent space so those robust features can be identified. Such features should be conducive to improving the learning efficiency and generalization of RL. We use the architecture proposed in [17] to distil the essential structural information about the environment. There are mainly two differences from their model in our work: (1) differing from its training over offline data, we firstly sample thousands of images in the simulator with a random policy and augment each of them twice to form a pair of similar images; (2) we redesign the encoders where the attention mechanism is embedded.

3.1.1. Contrastive Learning

As shown in Figure 1a, there are two encoders used to learn the efficient state representation from images. The key encoder θ_k is responsible for generating a key vector k that can summarize the information of an image, while the query encoder θ_q generates a vector q to query its similarity between the key vector. If the query vector can successfully obtain the similarity, it means that the encoders are capable of extracting the efficient similarity information from images. To train the contrastive learning model, firstly, there are thousands of images sampled from the environment. Each of them is augmented twice to form an image pair and stored in an image buffer, which is the source of training data. Secondly, at every epoch, a batch of image pairs is sampled from the buffer. The query and the key encoders respectively take the images from the same pairs as input. The key vector obtained here is defined as positive sample k_+. At every epoch, the key vectors are pushed into a queue to provide a large number of negative samples, which is helpful in identifying the similarity information. According to [17], each query vector will be queried with one positive example and all the negative examples. The training target of contrastive learning is to maximize the similarity between the query vector and the positive sample, as well as minimize the negative samples. Therefore, the loss function $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = -\log \frac{\exp\left(q \cdot k_+\right)}{\sum_{i=0}^{K} \exp\left(q \cdot k_i\right)} \tag{1}$$

where $K-1$ is the capacity of the queue which contains all the negative samples. Then, the loss is backpropagated to update the parameters of the encoders. Note that the gradient is only propagated along the query encoder, and the key encoder is updated with $\gamma\theta_k + (1-\gamma)\theta_q$ where γ is a constant [17].

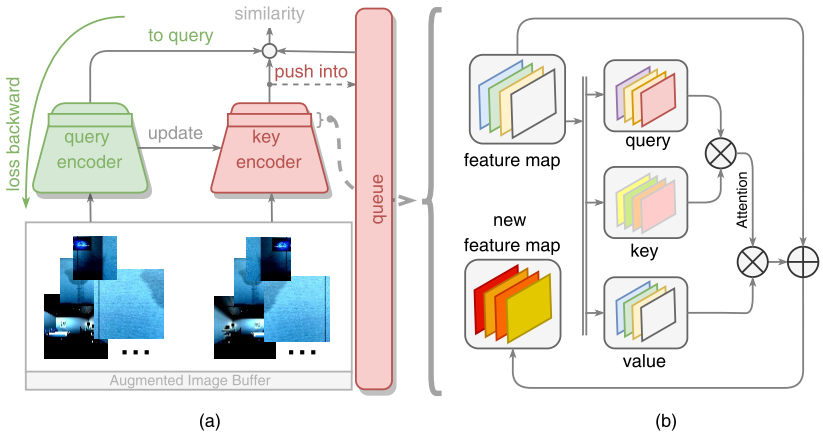

Figure 1. (**a**) Contrastive learning model. The query encoder and key encoder respectively take augmented images from the buffer as the input and maximize their similarity as much as possible. (**b**) Attention mechanism (self-attention) implementation. The $\otimes$ represents matrix multiplication operation and the $\oplus$ is matrix addition operation.

3.1.2. Image Augmentation

Image augmentation is an important step in contrastive learning because it is the source of similarity. Specifically, for the same picture, we apply two different augmentation methods to it, respectively. Thus, there would be some similarities between these two augmented pictures. In this paper, we use six augmentations [17], which are listed in Table 1. All augmentation operations are supported by the Python package Torchvision [41]. Due to the importance of geometric information for robot navigation, those augmentations that would distort the picture are not applied. In addition, we also find that image augmentation with different intensities can significantly affect the result of representation learning. Excessive augmentation may cause the two augmented images to be completely dissimilar, while weak augmentation can make it impossible to identify the key features. Thus, it is valuable to find the most appropriate augmentation methods for contrastive learning. However, note that in this paper, we do not investigate this topic in depth. We will discuss it in Section 6.

Table 1. Augmentation methods used in this paper.

Augmentations	Parameters	Descriptions
transforms.RandomHorizontalFlip	$p = 0.5$	Horizontally flip the given image randomly with a given probability.
transforms.RandomVerticalFlip	$p = 0.5$	Vertically flip the given image randomly with a given probability.
transforms.RandomResizedCrop	$scale \in [0.2, 1.0]$	Crop the given image at a random location.
transforms.RandomGrayscale	$p = 0.2$	Randomly convert image to grayscale with a probability
transforms.ColorJitter	$brightness = 0.4$, $contrast = 0.4$, $saturation = 0.4$, $hue = 0.1$	Randomly change the brightness, contrast, saturation and hue of an image.
transforms.Normalize	$mean = (0.485, 0.456, 0.406)$, $std = (0.229, 0.224, 0.225)$	Normalize a tensor image with the mean and standard deviation.

transforms: a module from Torchvision package version 0.4.0 [41].

3.1.3. Query/Key Encoder Architecture

In this paper, the discriminator component of GAN from [13] is employed as the key and query encoders. The reason why we chose the discriminator to construct the encoder is that it is capable of accurately distinguishing the generated realistic images, so its architecture should be conducive

to making an accurate summary of an entire image. This summary would contribute to the image similarity comparison in contrastive learning.

As shown in Table 2, there are seven layers in the encoder. Three convolutional layers are employed to locally capture dependencies among pixels, which would also help reduce the computational complexity in the attention calculation phase. Two attention layers follow to figure out the global dependencies among pixels. Through redundant attention calculation, it is possible to more accurately and comprehensively capture dependency information. What is more, compared with the original version of the discriminator, we keep almost all its parameters and settings except that all the convolutional layers are initialized orthogonally in our paper, and the number of final output channels is set as 512 to provide a feature vector for RL. Note that the key and query encoders employ the same network architecture in our work.

Table 2. Query/Key encoder architecture

NO.	Layers	Parameters	Activation	Parameters
1	Conv2d	(3, 64, kernel=4, stride=2, padding=1)	LeakyReLU	negative slope=0.1
2	Conv2d	(64, 128, kernel=4, stride=2, padding=1)	LeakyReLU	negative slope=0.1
3	Conv2d	(128, 256, kernel=4, stride=2, padding=1)	LeakyReLU	negative slope=0.1
4	Attention	–	–	–
5	Conv2d	(256, 512, kernel=4, stride=2, padding=1)	LeakyReLU	negative slope=0.1
6	Attention	–	–	–
7	Conv2d	(512, 512, kernel=4)	LeakyReLU	negative slope=0.1

3.1.4. Attention Module Design

Referring to [13], we employ self-attention to calculate the dependencies among pixels, which can be seen in Figure 1b. The main idea of our attention mechanism is to calculate the correlation among each pixel and the others then use the correlation to update the state of the pixel. The parameters and calculation process of the self-attention can be seen in Algorithm 1. We firstly define three convolutional layers to learn the query, key and value, respectively. Then matrix multiplication operation is applied to the query and key to calculate the attention. The kernel size of each layer is set as 1, which can increase the resolution of convolutional networks as well as making connections among neurons more sparse. To make the learning process more stable, the final returned value m' is obtained by weighted sum of the newly calculated value and the original value. Therefore, α is initialized to 0 while β is 1.

Algorithm 1 Self-Attention calculation.

Input: Feature map m with shape of $N \times c \times w \times h$;

Output: Feature map m' with shape of $N \times c \times w \times h$;

Definitions: $Conv2d_q = Conv2d(c, c|8, kernel = 1)$;

$Conv2d_k = Conv2d(c, c|8, kernel = 1)$;

$Conv2d_v = Conv2d(c, c, kernel = 1)$;

q:query vectors; k:key vectors; v:value vectors;

A:Attention map; mm:Matrix multiplication operation; α, β: trainable parameters; $*$: multiple operation.

$q = Conv2d_q(m)$
$k = Conv2d_k(m)$
$v = Conv2d_v(m)$
$A = mm(q, k)$
$m' = \alpha * mm(v, A) + \beta * m$
return m'

3.2. Policy Learning

The whole training process is divided into two phases: state representation learning and policy learning. Thus there should be a standard RL framework that can embed different state representations so that we can evaluate their performance.

3.2.1. Evaluation Framework

As seen in Figure 2, the Actor-Critic framework is employed to learn the policy. A dedicated representation extraction component is created to provide different state representations for RL. The intuition behind this framework is that: (1) when using different state representations, the remaining component of the framework should be unchanged so that the experimental results are comparable; (2) changing the state representation should be able to significantly affect the performance of RL. The framework of Actor-Critic can just meet our requirements. This framework is similar to the human learning process: (1) collecting and storing environmental information (perception and memory); (2) summarizing the collected information to form inductive knowledge (induction); (3) using the knowledge to deal with specific events (deduction). This process of human decision-making is not a fantasy. At the cognitive level, humans tend to use experience to build a model at the spiritual level (induction) and deduce the future through the learned model (deduction) [42,43]. Therefore, as the first step in the human learning process, i.e., perception and memory, the quality of state representation should have a significant impact on the performance of RL. Therefore, those compact and robust state representations should be helpful in the induction and deduction.

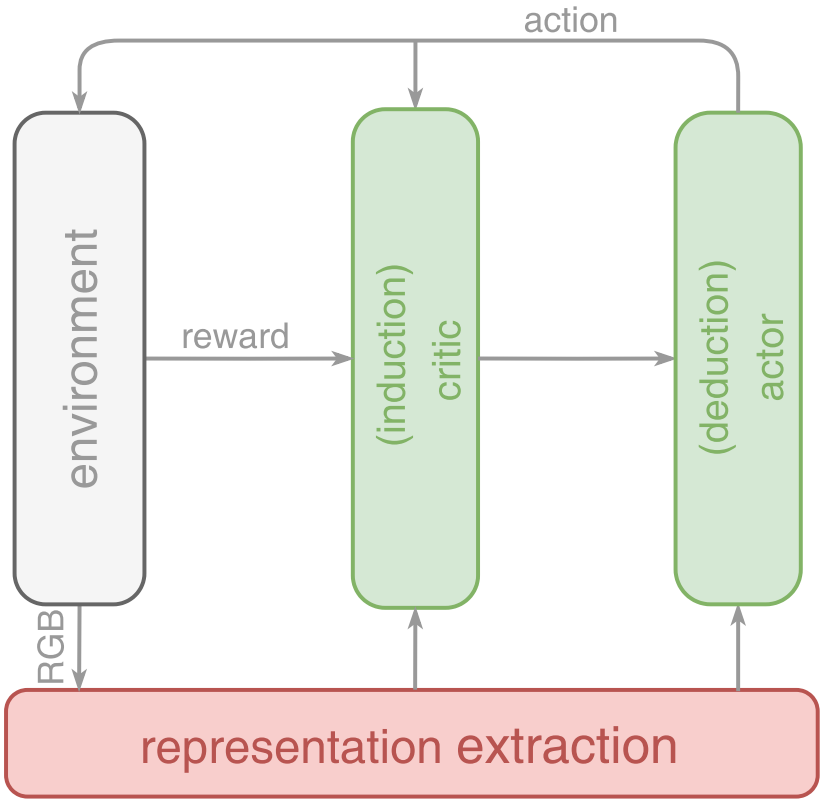

Figure 2. Evaluation framework. The representation learning is separated from policy learning so that different representation distillation modules can be embedded into reinforcement learning (RL) without changing the architecture and parameters of policy learning component.

3.2.2. Implementation

The proximal policy optimization (PPO) algorithm, which was fully described in [44], is used as a policy learning method to complete the navigation task. The detailed PPO algorithm implementation can be seen in [45]. Note that in our work: (1) the generalized advantage estimator (GAE) method [46] is employed to estimate the advantage of actions in each specific state; (2) to provide sufficient information for the robot, the vector of the target direction/location is appended to the visual representation. Figure 3 shows schematically the process of policy learning in our work. The top row is the training environment set, and the bottom row is the policy learning process. Specifically, there are four environments used to generate samples in parallel firstly. Then, state representations are distilled from these samples by the representation extraction module and stored in the experience buffer. Finally, a shared Actor-Critic model samples data from the buffer and updates the parameters of the neural network through the PPO algorithm. More training details will be described in Section 4. Moreover, since state representation learning is decoupled from policy learning, any other feasible policy learning implementations are allowed.

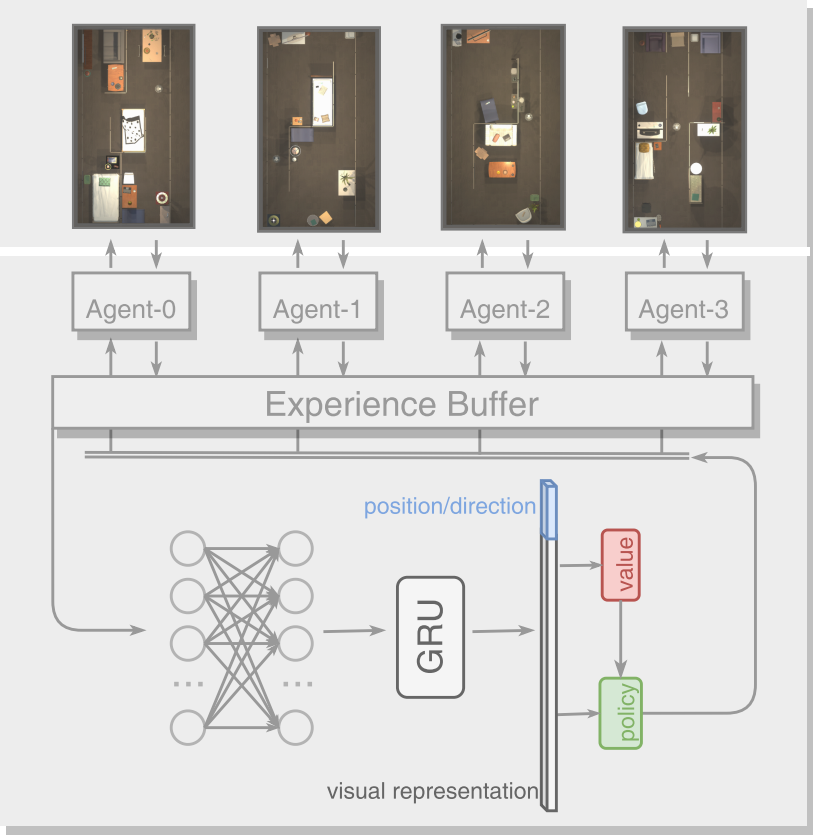

Figure 3. Policy learning implementation. The top row is the environment set used to train the agent. The bottom part is the shared Actor-Critic Model.

4. Experiments

In this section, we will describe our experiment settings and evaluate the performance of different state representations. Three metrics, i.e., reward curve, number of collisions, and success weighted by path length (SPL) [47], are used to evaluate the policies learned by agents with different state representations. Specifically, we use the reward curve to measure the learning efficiency caused by different state representations in training environments. Getting more rewards in a shorter time means faster learning. In the testing environment set, the number of collisions and SPL are employed to measure the generalization of RL. As mentioned in Section 2, the performance of the learned

policy would deteriorate to varying degrees in testing environments. Therefore, the purpose of our generalization experiment is to test the robustness of different state representations. Specifically, we focus on the generalization of obstacle and spatial geometry information stored in the state representation because this information plays a vital role in robot navigation. The fewer the number of collisions, the better the robustness of the obstacle information stored in the state representation is, and a higher SPL value means more general spatial geometry information is extracted.

4.1. Environments

The environment set RoboTHOR described in [48] is used to support our experiments. This is a 3D simulation provided for robots to interact with the environment, and robots perceive the environment through RGB images. However, it is not specially made for RL because there is no feedback reward when robots interact with the environment. Therefore, a gym-like [49] environment is made based on RoboTHOR in our work, and the detailed environment settings are described as follows.

There is a robot exploring to navigate to the target object, as well as to make it be in sight. At every time step, the robot can receive an RGB image and a target vector as the observation. Based on this perception information, the robot chooses an action in order to interact with the environment, which can bring a reward feedback to the robot. In order to complete the navigation task, the reward rule is set as follows: reaching and seeing the target object can win a reward of +10, while collision will bring -1. To encourage the robot to get closer to the target object, a dense reward of $(d_{t-1} - d_t)$ is given to it at every time step, and d_t is the distance between the robot and target object at time step t. The living penalty of -0.01 is also taken into consideration to make the robot find the shortest path to the destination. Available actions for interacting with the environment are listed as follows:

$$\mathcal{A} = \{\ \text{move\_forward}\ (0.25m),\ \text{turn\_left}\ (90^\circ),\ \text{turn\_right}\ (90^\circ)\}$$

where the number in the round brackets is the stride of each action. What is more, in our experiment, there were four environments chosen from RoboTHOR to be the training set, which can be seen in Figure 3 (top row). While in the testing phase, we randomly chose another one hundred environments (excluding the training set) from RoboTHOR to test the generalization of RL.

4.2. Baselines

1. Agents with visual priors. As shown in Figure 4, there are eight state-of-the-art representations extracted based on human prior knowledge [11]. Making no use of any priors should account for the learning inefficiency and generalization of RL to some extent. The main reason why we choose these representations as the baselines is that, as an unsupervised way, it is difficult to know how much prior information A2C can learn, especially the spatial geometry and obstacle information, which plays a crucial role in robot navigation. Through these visual priors formulated artificially, we can more intuitively know the prior information extracted by A2C. For example, if the performance of A2C exceeds the edge representation but is close to the segmentation, it is possible that the prior information extracted by A2C is closer to the object information rather than the edge information.
2. Random agent. The agent uniformly samples actions from the action space to interact with the environment regardless of the states. This baseline is used to test the difficulty of completing tasks in the simulator. If the random agent can obtain good performance, the simulator is not suitable for evaluating the effect of state representations.
3. Blind agent. The blind agent is developed from fixing the robot's state representation. Blind vision can indicate how well the agent performs without the help of correlations between the vision and the environment structure. By subtracting the performance of the blind agent, it is possible to know how much the state representation affects the performance of the non-blind agent. So this is a very important baseline.

4. Agent with representation from attention-removed A2C. Does the attention mechanism really work in A2C? In order to evaluate the effect of the attention mechanism, we remove it from A2C and learn another state representation for RL. In fact, this baseline is the vanilla contrastive learning method as mentioned before. The reason why we call it as attention-removed A2C is just to highlight the role of the attention mechanism in A2C.

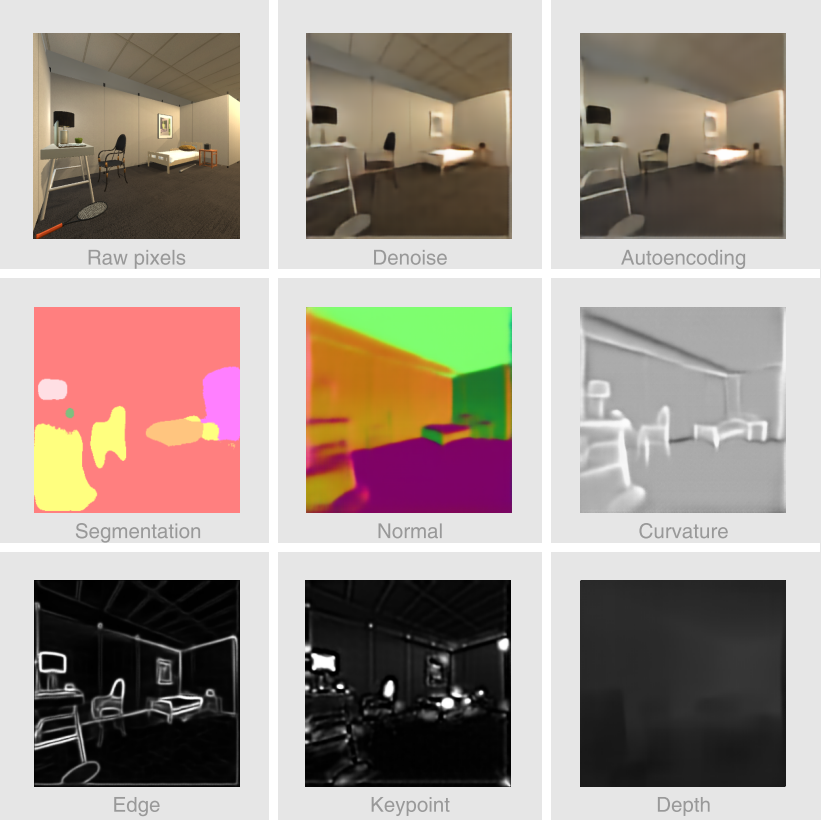

Figure 4. State representations with visual priors. Humans tend to observe the world with prior knowledge, e.g., it is more effective to use objects as the basic unit for understanding the world rather than the pixel. These representations contain higher-level prior knowledge about the environment, which is conducive to the learning efficiency and generalization of RL.

4.3. Metrics

1. Reward curve. Those state representations containing a regular pattern can significantly speed up the agent to discover patterns with high rewards. Therefore, the reward curve can indicate the effect of state representation on learning efficiency. In brief, getting more rewards in a shorter time means faster learning.
2. Number of collisions. In the navigation task, our goal is expressed in coordinates, as well as non-visual. Thus, the visual representations are mainly responsible for providing the environment structure information to make the robot avoid obstacles. Therefore, the number of collisions is an essential metric to measure the quality of state representations, especially the generalization.
3. Success Weighted by Path Length (SPL). This metric is another important one to measure the generalization performance. Here, we cite the definitions [47].

   We conducted N test episodes. In each episode, the agent is tasked with navigating to a goal. Let l_i be the shortest path distance from the agent's starting position to the goal in episode i and let p_i bet he length of the path actually taken by the agent. Let S_i be a binary indicator of success in

episode i. We define a summary measure of the agent's navigation performance across the test set as follows:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$$

4.4. Training Details

Pytorch [50] was used to construct neural networks and complete the network learning in our work. In the state representation learning phase, the batch size was set as 256, and the learning rate was 0.01. Each batch was used ten times to update the contrastive learning model with the Adam optimizer [51]. As for the policy learning, since the environment was partially observable to the robot, the gated recurrent unit (GRU) was employed. The value network and the policy network were implemented with the fully connected network (FCN) without a hidden layer, and the number of neurons was set as (512, 1), (512, 3), respectively. The learning rate was 0.00025, while the batch size was 128. As shown in Figure 3, we launched four processes to collect samples in different environments simultaneously. Moreover, a shared Actor-Critic model was updated using those training samples.

5. Results and Analysis

5.1. Sample Efficiency

As shown in Figure 5, agents with different state representations were trained with 2.5 million frames respectively, as well as rewarded with different efficiencies. The random agent failed in completing the navigation task, which indicated that the evaluation environment was valid. In conclusion, A2C is competitive with other representation learning methods. Even though it lost to autoencoding at the beginning, A2C surpassed autoencoding at the end. We explain this phenomenon as follows: In autoencoding, the agent needs to find the critical information that can restore the raw image in the latent space. If we regard the raw and the restored image as a pair of similar pictures, features extracted from autoencoding should be more compact than A2C because it tries to generate the same image rather than a similar one. Thus, it is not surprising that autoencoding had similar efficiency to A2C at the beginning. After the agent acquires the effective policy, A2C can help agents explore richer perception modes instead of being single-minded like autoencoding with the help of attention mechanism. What is more, the learning efficiency of autoencoding is not as stable as A2C. Table 3 shows that autoencoding had a higher standard deviation than A2C. There were even three phenomena in the reward curve of autoencoding in our experiments: basically unchanged, sharply rising, and suddenly falling, while A2C was always rising. The instability of the state representation is not acceptable in reality because it would cause the unreliability of control in reality. This concern is confirmed by Table 4. Autoencoding caused the most collisions in the testing phase. Therefore, although autoencoding was able to achieve higher learning efficiency, it did not have much practical value. On the contrary, A2C takes into account stability while ensuring better learning efficiency, and it has the best collision performance in Table 4. Therefore, in terms of learning efficiency, A2C obtained the best performance.

In addition, there was a gap between A2C and attention-removed A2C. Figure 6 displays two contrastive leaning loss curves of A2C and attention-removed A2C. Both of them basically converged to the same loss value, which shows that the encoders in both contrastive learning models were able to capture efficient similarity information. However, in Figure 5, A2C has higher learning efficiency than the attention-removed A2C when applied to RL. This contrast shows that the attention mechanism helps learn the essential similarity information, which should be attributed to its ability of calculating the correlation between two arbitrary pixels.

Table 3. Standard deviation of the reward distribution. A2C, attention-augmented contrastive learning method.

Autoencoding	Normal	Curvature	Denoise	Texture Edge	A2C	Attention-Removed A2C
164.16	78.18	30.68	51.15	53.62	80.84	32.03

Table 4. (Sorted by Mean value in ascending order) Number of collisions caused by agents with different state representations

NO.	Representations	Number	Mean Value	Standard Deviation	Maximum
1	A2C	100	0.71	1.46	7
2	Euclidean depth	100	1.15	4.03	27
3	Normal	100	2.72	6.90	45
4	Reshading	100	2.74	6.53	42
5	Segmentation	100	6.60	13.12	46
6	Denoise	100	9.24	23.55	119
7	Attention-removed A2C	100	9.90	28.22	241
8	Curvature	100	16.06	43.08	214
9	Texture edge	100	17.81	32.89	154
10	Autoencoding	100	30.19	77.65	256

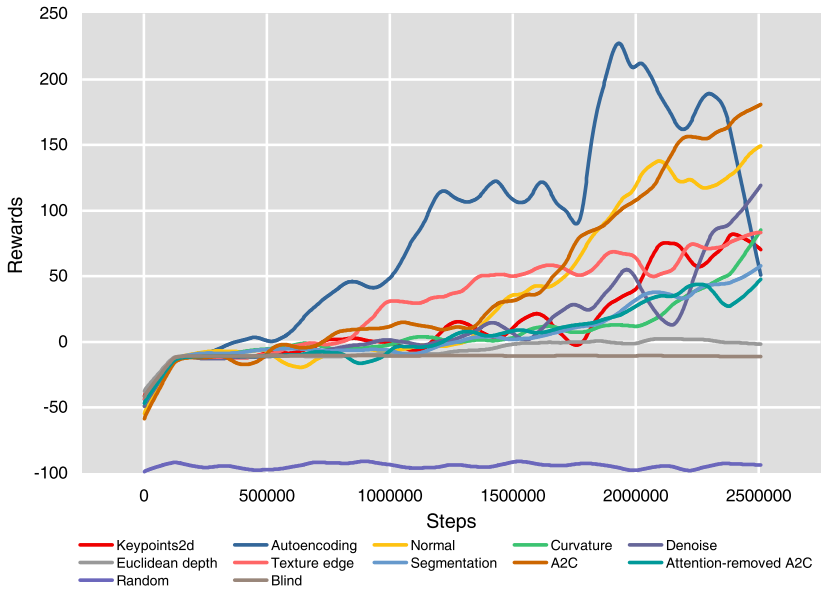

Figure 5. Reward curves. Rewards obtained by agents with different state representations in the policy learning phase.

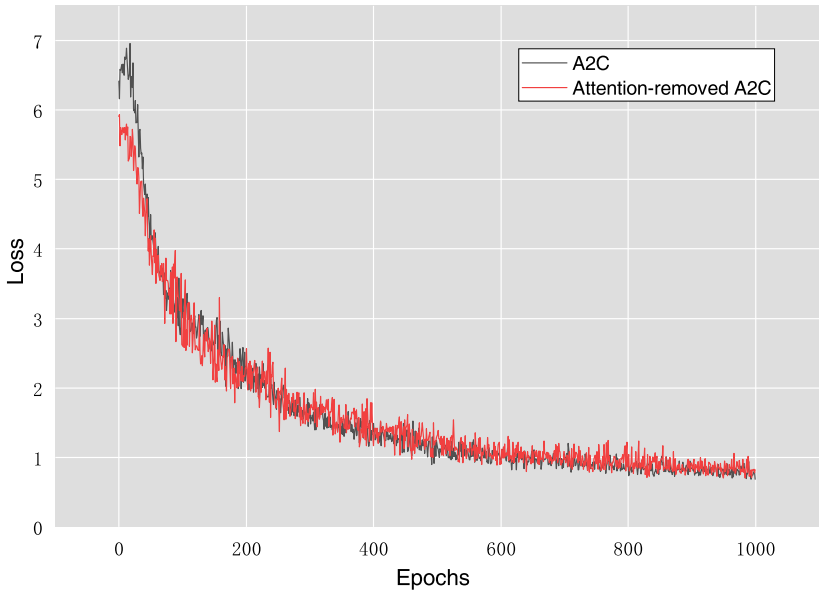

Figure 6. Loss curves of A2C and the attention-removed A2C in the state representation learning phase.

5.2. Generalization

There were one hundred environment settings used as the testing set to evaluate the generalization of RL. We recorded the number of collisions and SPL at each episode, which can be seen in Figure 7 and Table 4. A2C caused the fewest collisions and the smallest standard deviation, which indicates that the representation distilled from A2C provided more efficient obstacle information than others. Autoencoding did not achieve the same advanced performance as learning efficiency; even the number of collisions caused by it could reach about 250. If we regard autoencoding as a variant of attention-removed A2C (as mentioned above, in autoencoding, the raw image and the restored image can be regarded as a pair of similar pictures), lacking the attention mechanism may account for this phenomenon because removing the attention mechanism from A2C worsened the generalization performance of A2C. Hence, the attention mechanism is significantly helpful in capturing obstacle information.

In terms of SPL, the absolute value of SPL cannot reflect the advantages and disadvantages of state representations, because navigating to the target is not only related to the state representation, but also related to the performance of the policy modules like GRU. Therefore, we paid more attention to their relative performance. On the one hand, as shown in Table 5, A2C did not achieve the largest mean SPL value, which shows that it was relatively unsuitable for storing spatial geometry information. However, on the other hand, different from navigating in the simulator, avoiding collisions should be given much more priority in real robotic control because collisions may damage the robot and abort the navigation process. Therefore, although curvature, normal, and denoise achieved better SPL performance than A2C, they may perform worse than A2C in reality because their mean value, standard deviation, and maximum value in Table 4 are significantly larger than A2C, which means a greater probability of collisions. Moreover, considering that A2C had the best learning efficiency, we argue that A2C is the best choice for state representation in reality.

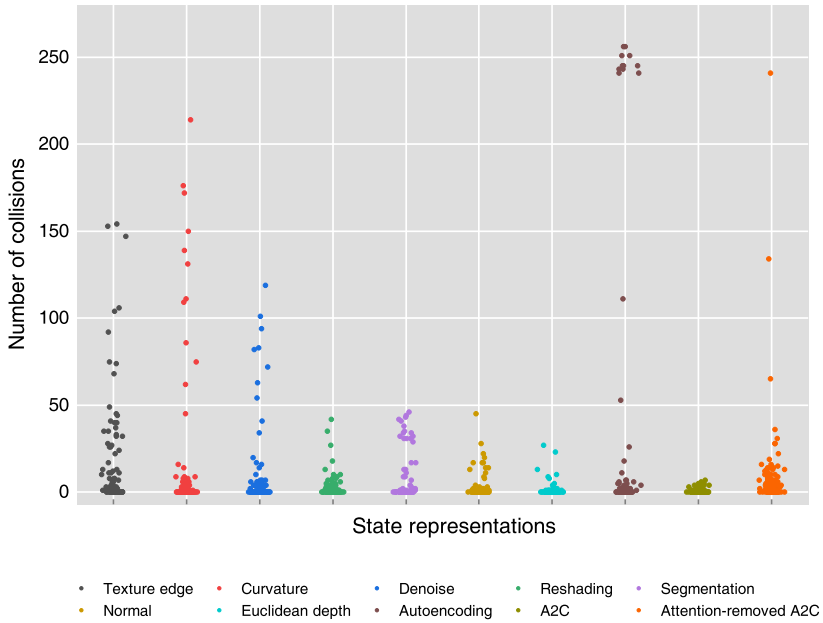

Figure 7. Number of collisions caused by agents with different state representations in the testing environments.

Table 5. (Sorted by mean success weighted by path length (SPL) in descending order) SPL obtained by agents with different state representations.

NO.	Representations	Number	Mean SPL	Standard Deviation	Maximum
1	Curvature	100	0.24	0.26	0.70
2	Normal	100	0.18	0.25	0.70
3	Denoise	100	0.15	0.24	0.70
4	A2C	100	0.15	0.23	0.70
5	Attention-removed A2C	100	0.13	0.22	0.70
6	Segmentation	100	0.13	0.23	0.70
7	Texture edge	100	0.12	0.21	0.65
8	Autoencoding	100	0.11	0.22	0.65
9	Reshading	100	0.10	0.21	0.66
10	Euclidean depth	100	0.05	0.15	0.62

5.3. Visualization

As shown in Figure 8, the white square over each image represents a query pixel (visual centre). We took fifteen snapshots from the simulator and visualized the attention relative to the visual centre. The brighter the part, the more attention is allocated. Unlike the traditional convolution operation, the correlation among pixels calculated by the attention mechanism was not limited to a fixed and regular shape or distance, which is conducive to capturing efficient information more accurately.

When faced with simple scenes (the first and last pictures in the second row, the third picture in the last row), the attention distribution was relatively uniform, while it was no longer uniform when faced with complicated scenes, and it did not show a specific pattern in our experiment. In fact, to maximize the similarity between the two augmented images, the contrastive learning model should make the query pixel learn to pay attention to those pixels maintaining similarity information as much as possible. Thus, it is not surprising that the learned attention had no regular pattern because the calculation of similarity was for an entire image rather than a specific pixel.

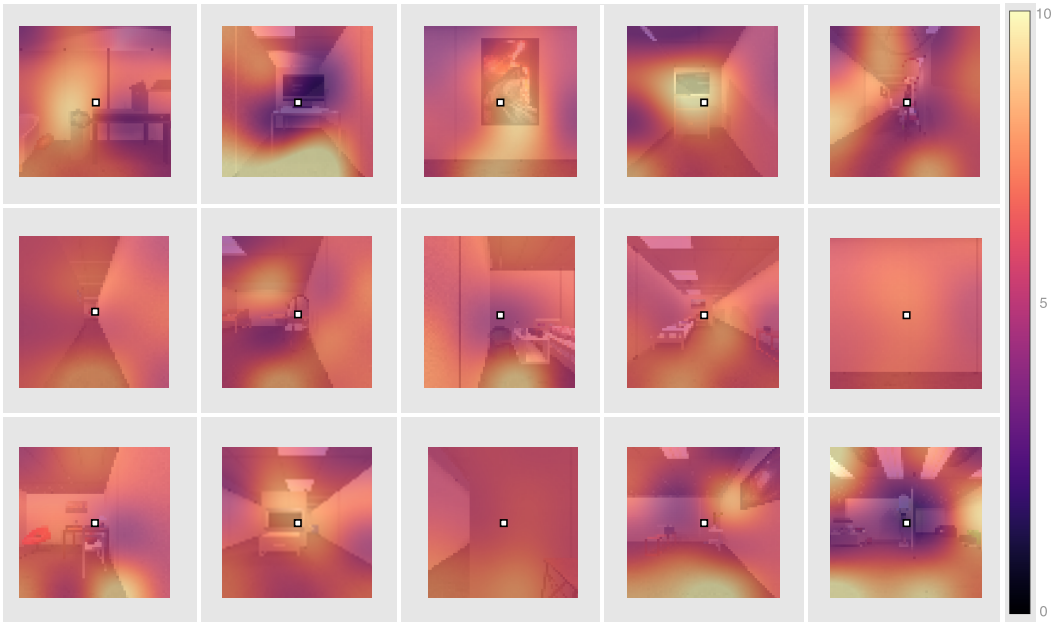

Figure 8. Attention Visualization. Fifteen snapshots are taken from the simulator. The visual central pixel is chosen as the query pixel.

6. Conclusions and Future Work

In this paper, we explore providing RL with more efficient state representations to improve its learning efficiency and generalization. It is shown that A2C achieves the best learning efficiency and collision performance. Although it is not the best performer in terms of SPL, it is still the most practical. Therefore, in conclusion, state representation learned by A2C has the best overall performance in improving the performance of RL. Our experiments show that by embedding the attention mechanism into contrastive learning, the encoders can more flexibly and effectively calculate the correlation among pixels without deepening the model, which improves the quality of learned state representations. Moreover, if we remove the attention mechanism from A2C, even if the same loss is reached in the representation learning phase, the performance of sample efficiency and generalization is not as good as A2C, which indicates that our attention mechanism is significantly helpful in contrastive learning.

Furthermore, the augmentation intensity applied to the image can greatly affect the result of contrastive learning. During our experiment, we found that excessive augmentation makes it difficult for encoders to learn the similarity between images. Therefore, it is valuable to find out the most appropriate augmentation methods and intensity in the future.

Author Contributions: Conceptualization, H.C.; methodology, H.C., Y.L., Z.Z., M.Z.; software, H.C.; validation, H.C., Y.L.; investigation, H.C.; resources, Y.L., Z.Z., M.Z.; data curation, H.C.; writing–original draft preparation, H.C.; writing–review and editing, Y.L., Z.Z., M.Z.; visualization, H.C.; supervision, Y.L.; project administration, Y.L., Z.Z., M.Z.; funding acquisition, Y.L., M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation OF China grant number U19A2083, the National Natural Science Foundation OF China grant number 61673389.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Li, Y. Deep Reinforcement Learning: An Overview. *arXiv* **2017**, arXiv:1701.07274.
2. Lesort, T.; Rodríguez, D.N.; Goudou, J.F.; Filliat, D. State Representation Learning for Control: An Overview. *Neural Netw.* **2018**, *108*, 379–392. [CrossRef]

3. Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J.J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In Prcoceedings of the IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017.
4. Arulkumaran, K.; Deisenroth, P.M.; Brundage, M.; Bharath, A.A. A Brief Survey of Deep Reinforcement Learning. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
5. Das, A.; Kottur, S.; Moura, J.M.F.; Lee, S.; Batra, D. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
6. Nair, A.; Pong, V.; Dalal, M.; Bahl, S.; Lin, S.; Levine, S. Visual Reinforcement Learning with Imagined Goals. *Adv. Neural Inf. Process. Syst.* **2018**, 9191–9200. Available online: http://papers.nips.cc/paper/8132-visual-reinforcement-learning-with-imagined-goals (accessed on 31 December 2018).
7. Moreira, I.; Rivas, J.; Cruz, F.; Dazeley, R.; Ayala, A.; Fernandes, B. Deep Reinforcement Learning with Interactive Feedback in a Human–Robot Environment. *Appl. Sci.* **2020**, *10*, 5574. [CrossRef]
8. Chen, C.; Liu, Y.; Kreiss, S.; Alahi, A. Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning. In Prcoceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal Convention Centre, Montreal, QB, Canada, 20–24 May 2019, pp. 6015-6022.
9. Reinhard, D. *Graph Theory*, 5th ed; Springer: Berlin, Germany, 2017.
10. Goodfellow, J.I.; Bengio, Y.; Courville, C.A. Deep Learning. In *Deep Learning*; The MIT Press: Cambridge, MA, USA, 2016; pp. 1–775.
11. Sax, A.; Emi, B.; Zamir, A.R.; Guibas, L.J.; Savarese, S.; Malik, J. Mid-Level Visual Representations Improve Generalization and Sample Efficiency for Learning Active Tasks. *arXiv* **2018**, arXiv:1812.11971.
12. radford, a.; metz, l.; chintala, s. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2015**, arXiv:1511.06434 .
13. Zhang, H.; Goodfellow, J.I.; Metaxas, N.D.; Odena, A. Self-Attention Generative Adversarial Networks. *arXiv* **2018**, arXiv:1805.08318.
14. Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: New York, NY, USA, 2016; pp. 2180–2188.
15. Zhuang, F.; Luo, D.; Jin, X.; Xiong, H.; Luo, P.; He, Q. Representation Learning via Semi-Supervised Autoencoder for Multi-task Learning. In Proceedings of the IEEE International Conference on Data Mining, Atlantic City, NJ, USA, 14–17 November 2015; pp. 1141–1146.
16. Arora, S.; Khandeparkar, H.; Khodak, M.; Plevrakis, O.; Saunshi, N. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. *arXiv* **2019**, arXiv:1902.09229.
17. Kaiming, H.; Haoqi, F.; Yuxin, W.; Saining, X.; Ross, G. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 19 August 2019.
18. Aravind, S.; Michael, L.; Pieter, A. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. *arXiv* **2020**, arXiv:2004.04136.
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2017. [CrossRef]
20. Mnih, V.; Badia, P.A.; Mirza, M.; Graves, A.; Lillicrap, P.T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, Edinburgh, UK, 26 June–1 July 2012; pp. 1928–1937.
21. Konda, R.V.; Tsitsiklis, N.J. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2003; pp. 1143–1166.
22. Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; Kautz, J. GA3C: GPU-based A3C for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1611.06256.
23. Kaiser, L.; Babaeizadeh, M.; Miłos, P.; Osiński, B.; Campbell, H.R.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. Model Based Reinforcement Learning for Atari. *arXiv* **2020**, arXiv:1903.00374.

24. Kamalapurkar, R.; Walters, P.; Dixon, E.W. Model-based reinforcement learning for approximate optimal regulation. *Automatica* **2016**, *64*, 94–104. [CrossRef]
25. Zhou, S.; Tan, B. Electrocardiogram soft computing using hybrid deep learning CNN-ELM. *Appl. Soft Comput.* **2019**, *86*, 105778. [CrossRef]
26. Chen, Y.; Xiong, J.; Xu, W.; Zuo, J. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Clust. Comput.* **2019**, *22*, 7435–7445 [CrossRef]
27. Tu, Y.; Lin, Y.; Wang, J. Semi-supervised Learning with Generative Adversarial Networks on Digital Signal Mod-ulation Classification. *Cmc-Comput. Mater. Contin.* **2018**, *55*, 243–254.
28. Zeng, D.; Dai, Y.; Li, F.; Sherratt, S.R.; Wang, J. Adversarial Learning for Distant Supervised Relation Extraction. *Cmc-Comput. Mater. Contin.* **2018**, *55*, 121–136.
29. Cai, Z.; Huang, L. Finite-Time Stabilization of Delayed Memristive Neural Networks: Discontinuous State-Feedback and Adaptive Control Approach. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 856–868. [CrossRef]
30. Gadekallu, R.T.; Khare, N.; Bhattacharya, S.; Singh, S.; Maddikunta, K.R.P.; Srivastava, G. Deep neural networks to predict diabetic retinopathy. *J. Ambient. Intell. Humaniz. Comput.* **2020**. [CrossRef]
31. Raileanu, R.; Denton, E.; Szlam, A.; Fergus, R. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. *arXiv* **2018**, arXiv:1802.09640.
32. Levine, S.; Koltun, V. Guided Policy Search. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1–9.
33. Deisenroth, P.M.; Rasmussen, E.C. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In Proceedings of the 28th International Conference on machine learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 465–472.
34. Schaal, S.; Schaal, S. Is imitation learning the route to humanoid robots? *Trends Cogn. Sci.* **1999**, *3*, 233–242. [CrossRef]
35. Ho, J.; Ermon, S. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA; 2016; pp. 4565–4573.
36. Tamar, A.; Glassner, Y.; Mannor, S. Optimizing the CVaR via Sampling. *arXiv* **2014**, arXiv:1404.3862.
37. Morimoto, J.; Doya, K. Robust reinforcement learning. *Neural Comput.* **2005**, *17*, 335–359. [CrossRef]
38. Pinto, L.; Davidson, J.; Sukthankar, R.; Gupta, A. Robust Adversarial Reinforcement Learning. *arXiv* **2017**, arXiv:1703.02702.
39. Mishra, N.; Rohaninejad, M.; Chen, X.; Abbeel, P. Meta-Learning with Temporal Convolutions. *arXiv* **2017**, arXiv:1707.03141.
40. Duan, Y.; Schulman, J.; Chen, X.; Bartlett, L.P.; Sutskever, I.; Abbeel, P. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv* **2017**, arXiv:1611.02779.
41. Marcel, S.; Rodriguez, Y. Torchvision the machine-vision package of torch. In Proceedings of the 18th ACM International Conference on Multimedia, Firenze, Italy, 25–29 October 2010; pp. 1485–1488.
42. Demis, H.; Dharshan, K.; Maguire, E.A. Using imagination to understand the neural basis of episodic memory. *J. Neurosci. Off. J. Soc. Neurosci.* **2007**, *27*, 14365–14374.
43. Schacter, D.L.; Addis, D.R.; Hassabis, D.; Martin, V.C.; Spreng, R.N.; Szpunar, K.K. The Future of Memory: Remembering, Imagining, and the Brain. *Neuron* **2012**, *76*, 677–694. [CrossRef]
44. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
45. Achiam, J. Spinning Up in Deep Reinforcement Learning. Available online: https://spinningup.openai.com (accessed on 31 December 2018).
46. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv* **2016**, arXiv:1506.02438.
47. Anderson, P.; Chang, X.A.; Chaplot, S.D.; Dosovitskiy, A.; Gupta, S.; Koltun, V.; Kosecka, J.; Malik, J.; Mottaghi, R.; Savva, M.; et al. On Evaluation of Embodied Navigation Agents. *arXiv* **2018**, arXiv:1807.06757.
48. Deitke, M.; Han, W.; Herrasti, A.; Kembhavi, A.; Kolve, E.; Mottaghi, R.; Salvador, J.; Schwenk, D.; VanderBilt, E.; Wallingford, M.; et al. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In Prcoeedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–18 June 2020.

49. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540.
50. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch - An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2019; pp. 8024–8035.
51. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.