Learning Action Representations for Self-supervised Visual Exploration

Changjae Oh and Andrea Cavallaro

***Abstract*— Learning to efficiently navigate an environment using only an on-board camera is a difficult task for an agent when the final goal is far from the initial state and extrinsic rewards are sparse. To address this problem, we present a self-supervised prediction network to train the agent with intrinsic rewards that relate to achieving the desired final goal. The network learns to predict its future camera view (the future state) from a current state-action pair through an Action Representation Module that decodes input actions as higher dimensional representations. To increase the representational power of the network during exploration we fuse the responses from the Action Representation Module in the transition network, which predicts the future state. Moreover, to enhance the discrimination capability between predictions from different input actions we introduce joint regression and triplet ranking loss functions. We show that, despite the sparse extrinsic rewards, by learning action representations we achieve a faster training convergence than state-of-the-art methods with only a small increase in the number of the model parameters.**

I. Introduction

Visual navigation based on first-person vision is important for task-oriented as well as exploratory applications, such as a drone searching for an object or a robot entering an unseen environment. Reinforcement Learning (RL) is a promising approach that uses the experience collected by the agent (robot) for sequential decision making for navigation [1], [2], [3] and manipulation [4], [5]. For visual navigation, the agent is trained to perform actions to reach, from the current camera view (state), the final goal state. Since the goal state can be several sequential actions (steps) away, a reward explicitly given only in the final state is too sparse to effectively train the agent.

Training the agent to collect experience about the environment through exploration is still a challenging task in deep RL. Conventional strategies rely on heuristics, such as the ϵ-greedy approach that chooses the action that most likely generates a long-term effect [6] or noise-based exploration that generates different actions from the same state [7]. Imitation learning [8] is widely used to acquire skills from expert demonstrations and to learn policies with behavioural cloning [8], [9], [10] or inverse reinforcement learning [11], [12], [13]. Other approaches reshape an original reward function to encourage the agent to obtain intrinsic rewards that relate the most to the achievement of the final goal [14], [15], [16], [17], [18].

Intrinsic motivation, e.g. the difference between predicted (future) state and ground truth, encourages the agent to explore novel states given the current state and action [17]. The input action is generally a simple one-hot encoding [1], [3], [17], [19], [20], which allows one to describe categorical variables as vectors. Examples are $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ to encode *turn left*, *move forward* and *turn right*, respectively. However, combining a one-hot code after the feature of the current state restricts the influence of the input action in predicting the future state during training. In fact, with first-person view, the distribution among future states from different input actions can be extremely different. We thus argue that the network should be able to describe various input actions to predict the state, which has not been handled in previous works [1], [3], [17], [19], [20].

This work was supported by the EPSRC Project NCNR (EP/R02572X/1).
Changjae Oh and Andrea Cavallaro are with the Centre for Intelligent Sensing, Queen Mary University of London, E1 4NS, London, United Kingdom. c.oh@qmul.ac.uk; a.cavallaro@qmul.ac.uk

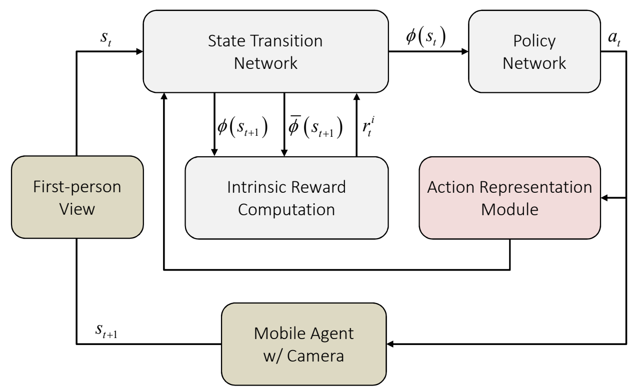


Fig. 1: Self-supervised visual exploration with intrinsic rewards, r_t^i. The agent reaches the new state, s_{t+1}, based on an input action, a_t, determined by the policy network. The proposed Action Representation Module generates higher dimensional representations of a_t enabling an effective prediction of s_{t+1} through the feature space $\bar{\phi}(\cdot)$.

In this paper, we propose a self-supervised prediction network for the agent to learn from sparse extrinsic rewards as well as to manage intrinsic rewards. While previous works designed intrinsic rewards [14], [15], [16], [17], we introduce a deep network that learns features for effective self-supervised visual exploration. In particular, we explicitly increase the dimensions of the input action with a decoding process. This higher dimensional representation is then fused in the state transition network to predict the future state (Fig. 1).

The main contributions of this paper are as follows. We propose an Action Representation Module for efficient RL that can be easily integrated with existing convolutional neural network (CNN) architectures. This module expands the dimensions of an input action to improve the representational power of the network during training. We use joint regression and triplet ranking loss functions to predict the future state while discriminating predictions from different actions to encode meaningful features effectively during training. The proposed module equipped with these losses has a faster training convergence than state-of-the-art methods with only a 0.5% increase in the total number of parameters. Our approach is generic and, in this work, we evaluate it using as state a first-person view (camera view).

II. Related Work

This section covers the main approaches to address the problem of sparse extrinsic rewards: imitation learning, self-supervised learning, and curiosity-driven exploration.

An agent (robot) can learn behaviours from expert demonstrations with *imitation learning*, which is used for autonomous driving [21], [22], drone navigation [23], [24], locomotion [25], and manipulation [11], [26]. Imitation learning methods can be classified as behavioural cloning or inverse reinforcement learning. *Behavioural cloning* consists in directly learning a policy through state-action pairs provided by an expert, without the robot interacting with the environment during training [8], [9], [10]. *Inverse reinforcement learning* consists in receiving state-action pairs from an expert and in learning to estimate a reward function that leads to expert actions [24]. The reward function is used to infer an imitation policy combined with RL [11], [12], [13]. Imitation learning has good performance, but the expert supervision is labour intensive, prone to bias, and needed for each new task.

When expert supervision is unavailable or insufficient to learn features with a deep neural network, *self-supervision* can be used [27]. A common approach is to train a network with a proxy task that relates to the final goal of the task itself [27], [28], [29]. Self-supervised strategies can also be used to address the problem of sparse extrinsic rewards in RL [5], [20], [30], [31]. As the robot moves sequentially, the future state and action can be naturally obtained without supervision. Therefore a prediction model can be used to estimate, based on the current state-action pair, the future state [20], [30] or action [31], which relates to the final goal for manipulation [5], [20], [30] or navigation [17], [20].

Finally, *curiosity-driven exploration* reshapes the original reward function by designing intrinsic rewards that encourage the agent to explore unseen areas of the environment [32], [33]. An alternative is to maximise the information gain for the agent and to reduce the uncertainty about the environment [14], [16]. For exploration, a robot can use state visitation counts [15], [34], [35] or the error between predicted and real future states [17], [36]. While the above-mentioned methods proposed the design of intrinsic reward functions, we focus on the design of a model to effectively learn meaningful features during exploration to facilitate navigation in unknown environments. Our work is inspired by a curiosity-driven exploration with prediction error [17], but we investigate the model and loss functions for exploration.

III. Problem Formulation

Let an agent interact with an environment over a number of discrete time steps. At each time step, t, the agent receives a state, s_t, the image from its first-person view, and selects an action, a_t, from a set of possible actions according to a policy $\pi : s_t \to a_t$. In return, the agent receives the next state, s_{t+1}, and a reward, r_t. The goal of the agent is to maximise the expected reward from each state s_t. This process ends when the agent reaches its goal or a maximum predefined number of time steps. During training, the process is repeated until the number of steps reaches a predefined value.

If θ_P represents the parameters of the neural network that comprises the policy to determine the action, the goal is to maximise the expected sum of rewards during T steps by optimizing a parameterised policy, $\pi\left(s_t;\theta_P\right)$, as follows [37], [38]:

$$\max_{\theta_P} E_{\pi(s_t;\theta_P)}\left[\sum\nolimits_{t=0}^{T} r_t\right]. \tag{1}$$

A challenge for this optimization is that the agent can learn from extrinsic rewards only when it succeeds to reach the final goal state. In the real world, the agent may operate in an environment where there are no extrinsic rewards or where extrinsic rewards are sparsely distributed. To address this challenge, self-supervised approaches for RL equip the agent with intrinsic rewards that closely relate to the understanding of the environment in order to reach the final goal [14], [15], [16], [17].

IV. Action Representations for Self-supervised Exploration

The proposed model for exploring unseen environments relies on the theory of curiosity-driven exploration [17], where the agent undertakes a systematic exploration to unveil unseen regions. We encode the input image (state) and an action into CNNs with two models (see Fig. 2). A *Forward Model* predicts the future state from the current state-action pair, providing a prediction error as an intrinsic reward to the agent. Here we introduce an Action Representation Module that decodes an input action for an effective prediction of the state. To effectively train the Forward Model, we use joint regression and triplet ranking loss functions. The *Inverse Model* classifies the action performed to become a future state from a current state. This action classification task enables the features from the Forward Model to encode information to determine the action.

A. Forward Model

Given a state-action pair (s_t, a_t) at time t, the Forward Model $f\left(\cdot\right)$ predicts s_{t+1} as a high-level feature representation $\bar{\phi}\left(s_{t+1}\right)$ to constrain the state transition and to encode information more efficiently:

$$\bar{\phi}\left(s_{t+1}\right) = f\left(s_t, a_t; \theta_A, \theta_F\right), \tag{2}$$

where the network parameters θ_A and θ_F are learned for the Action Representation Module and the state transition network, respectively.

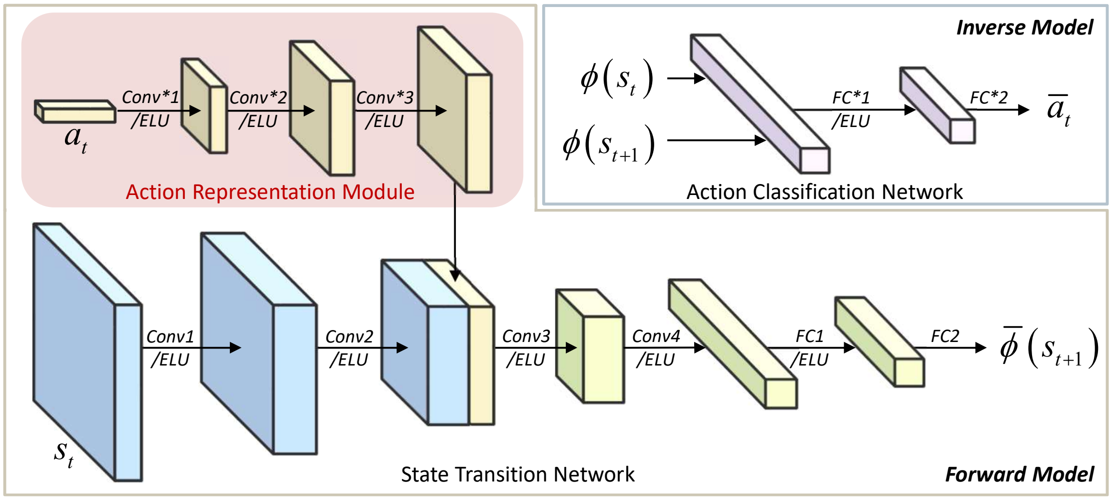


Fig. 2: The proposed architecture for self-supervised visual exploration. The Forward Model, which consists of a state transition network combined with an Action Representation Module, estimates the feature of the (future) state $\bar{\phi}\left(s_{t+1}\right)$ from an input state-action pair $\left(s_t, a_t\right)$. The one-hot code of an input action passes through three convolution layers (Conv*) with Exponential Linear Unit (ELU) to generate decoded responses that are then concatenated to an intermediate response of the state transition network. The concatenated responses estimate the feature of the future state after subsequent convolutions (Conv) and fully connected (FC) layers. The Inverse Model classifies the action $\bar{a}_t$ that should be performed to reach the future state from the current state.

The proposed Action Representation Module consists of three convolutional layers with nonlinear activations (Exponential Linear Unit or ELU) [39]. The input action a_t, represented as a one-hot code, passes through three convolutional layers and ELUs to generate a decoded response that has a higher dimensionality than the input one-hot vector representing the action code. Then the output of the Action Representation Module is concatenated with an intermediate response of the state transition network and fused to predict the state in the higher dimensionality feature space, $\bar{\phi}(s_{t+1})$, which is more expressive for the training. This solution considerably differs from previous works that represented an input action as a one-hot code that was simply concatenated with the response from the fully-connected (FC) layers of the state transition network [1], [3], [17], [19], [20].

In training, the parameters from the Action Representation Module are implicitly learned from the input actions without an explicit loss function related to this module, which derives higher dimensional features from one-hot action codes. Fig. 3 shows a sample response generated by each action code.

To learn the Forward Model effectively, we use joint regression and triplet ranking loss functions. The *regression loss function*, L_{F_1}, minimises the prediction error:

$$L_{F_1}\left(\bar{\phi}\left(s_{t+1}\right), \phi\left(s_{t+1}\right)\right) = \left\|\bar{\phi}\left(s_{t+1}\right) - \phi\left(s_{t+1}\right)\right\|_2^2, \quad (3)$$

where $\phi\left(s_{t+1}\right)$ is the feature representation from the image (state) s_{t+1}. L_{F_1} ensures that the predicted state from an input state-action pair is close to the future state. As the prediction of the state without any actions is the current state itself, s_{t+1} can be encoded into a feature representation as:

$$\phi\left(s_{t+1}\right) = f\left(s_{t+1}, a_{t+1} = \emptyset; \theta_A, \theta_F\right), \quad (4)$$

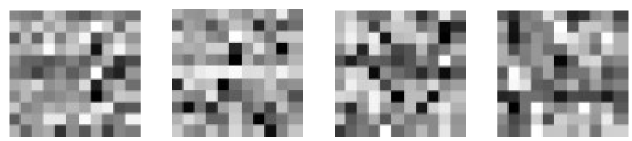
where $a_{t+1} = \emptyset$ denotes *no action*.

The *triplet ranking loss function* [40], L_{F_2}, pushes $\bar{\phi}\left(s_{t+1}\right)$ to be far from a prediction from s_t with a different input action, which we define as $\bar{\phi}\left(\tilde{s}_{t+1}\right)$ (see Fig. 4), while maintaining the property of (3), in the form:

$$L_{F_2}\left(\bar{\phi}\left(s_{t+1}\right), \phi\left(s_{t+1}\right), \bar{\phi}\left(\tilde{s}_{t+1}\right)\right) = \max\Big\{0, m + \left\|\bar{\phi}\left(s_{t+1}\right) - \phi\left(s_{t+1}\right)\right\|_2^2 - \left\|\bar{\phi}\left(s_{t+1}\right) - \bar{\phi}\left(\tilde{s}_{t+1}\right)\right\|_2^2\Big\}, \quad (5)$$

where m is a margin. Here an action, different from the current action a_t, is randomly chosen.

The Forward Model can be explicitly trained to discriminate among the features generated from the same state but with different actions. Therefore, we define the final loss function for the Forward Model as $L_F = L_{F_1} + \gamma L_{F_2}$, where γ controls the effect of the triplet ranking loss function and L_{F_1} can be seen as a regulariser for training L_{F_2}.



(a) (b) (c) (d)

Fig. 3: Sample responses from the Action Representation Module for (a) *move forward*, (b) *turn right*, (c) *turn left*, and (d) *no action*. These responses have a higher dimensionality than the original one-hot vectors representing the four actions.

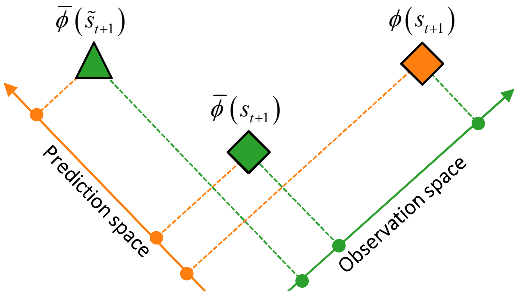


Fig. 4: The elements of the triplet ranking loss function for the self-supervised prediction network. $\bar{\phi}(s_{t+1})$ and $\bar{\phi}(\tilde{s}_{t+1})$ receive the same input state with different actions, thus having closer distance than $\phi(s_{t+1})$ in the observation space. The loss function L_{F_2} aims to encode $\bar{\phi}(s_{t+1})$ and $\phi(s_{t+1})$ close in the prediction feature space, while pulling $\bar{\phi}(s_{t+1})$ away from $\bar{\phi}(\tilde{s}_{t+1})$ to discriminate the predictions from different actions.

B. Intrinsic Reward

We estimate r_t^i, the intrinsic reward at time t, in a feature space rather than in the raw camera view [17]. We made this choice to enable the network to focus on action-relevant information during the self-supervised learning. We employ L_{F_1} as r_t^i as [17]:

$$r_t^i = \eta \left\| \bar{\phi}(s_{t+1}) - \phi(s_{t+1}) \right\|_2^2, \tag{6}$$

where η is a scaling factor. The larger the prediction error, the larger r_t^i, thus the agent is encouraged to explore unseen areas.

C. Inverse Model

The Inverse Model learns to recognise an actual action a_t from input states s_t and s_{t+1} and then generates a predicted action $\bar{a}_t$ to change from $\phi(s_t)$ to $\phi(s_{t+1})$ as:

$$\bar{a}_t = g\left(\phi(s_t), \phi(s_{t+1}); \theta_I\right), \tag{7}$$

where θ_I denotes the network parameters of the Inverse Model. The loss function $L_I(\bar{a}_t, a_t)$ we use for the Inverse Model is a soft-max as the problem in (7) generates a discrete action label, which can be considered as a classification among several possible discrete actions.

Based on the Forward and Inverse models, the overall optimization for training the agent is:

$$-E_{\pi(s_t;\theta_P)}\left[\sum_{t=0}^{k} r_t\right] + \beta L_F + (1-\beta) L_I, \tag{8}$$

where $r_t = r_t^e + r_t^i$, and r_t^e and r_t^i are extrinsic and intrinsic rewards, respectively; the hyper-parameter β controls the weight between the Forward and Inverse models; and k, which can vary for each episode but is upper bounded, is the number of time steps when the agent has moved.

The rationale behind training for navigation with (8) is that, in the early stage of training, the agent focuses on learning features by pursuing intrinsic rewards since it takes time to get to the final goal where the extrinsic reward is obtained. Therefore the action a_t sampled from the policy tends to move the agent where a higher prediction error is generated so as to receive a higher intrinsic reward. This enables the agent to explore unseen portions of the environment rather than remaining in previously experienced states. As the training progresses, the intrinsic rewards decrease with the forward loss function L_{F_1} and the relative importance of the extrinsic reward increases, thus enabling the agent to move to maximise the extrinsic reward.

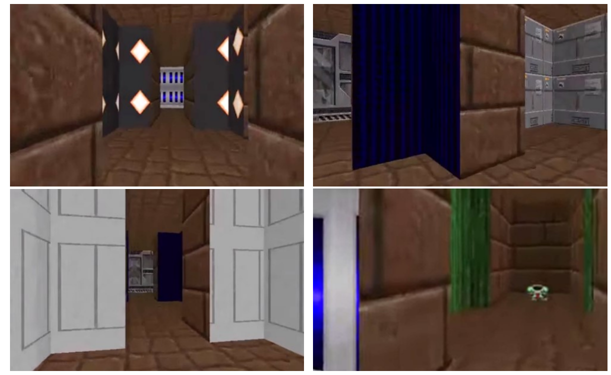


Fig. 5: Sample first-person views (states) from the *VizDoom MyWayHome* environment. The agent explores the environment until it finds the armour (see the bottom-right view) that triggers an extrinsic reward.

V. Results

A. Simulation Setup

We compare our model, AR4E (Action Representations for Exploration), with another self-supervised network, ICM (Intrinsic Curiosity Module) [17], and a network that only considers extrinsic rewards in training, A3C (a vanilla Asynchronous Advantage Actor-Critic) [38]. Both ICM and AR4E are built on A3C.

For a fair comparison, we follow the same architecture for the state transition network, except for AR4E, which has the three convolutional layers with ELUs for the Action Representation Module. This additional module only slightly increases the number of model parameters (+0.5%). Table I shows the details of the network configuration.

All agents are trained using visual inputs (images) converted to grey scale and resized to 42×42 pixels [38]. To train the networks, sixteen workers are used to perform RL following the asynchronous training protocol in A3C [38]. The feature from the Forward Model, $\phi(s_t)$, is fed into a long short term memory (LSTM) network with 256 units. Two separate FC layers are exploited to estimate the value function and the action a_t from the LSTM networks. We use the ADAM solver [42] with 10^{-4} as learning rate.

B. Scenario

We compare the performance of the networks and validate AR4E with a first-person view navigation task in the *VizDoom MyWayHome* environment (see Fig. 5). The goal here is to reach an armour, which gives an extrinsic reward [43], [44].

TABLE I: Details of the configuration of the networks. The Action Representation Module decodes the responses by setting stride to 2 [41]. 'Concat' denotes a concatenation that fuses two responses.

Forward Model				Inverse Model	
State Transition Network		Action Representation Module		Action Classification Network	
Layer	**Filter** (stride)	**Layer**	**Filter** (stride)	**Layer**	**Filter** (stride)
Conv1+ELU	$32 \times 1 \times 3 \times 3$ (1)	Conv*1+ELU	$4 \times 4 \times 3 \times 3$ (2)	Concat	-
Conv2+ELU	$32 \times 32 \times 3 \times 3$ (1)	Conv*2+ELU	$8 \times 4 \times 3 \times 3$ (2)	FC*1+ELU	$256 \times 512 \times 1 \times 1$ (1)
Concat	-	Conv*3+ELU	$8 \times 8 \times 3 \times 3$ (1)	FC*2	$4 \times 256 \times 1 \times 1$ (1)
Conv3+ELU	$40 \times 40 \times 3 \times 3$ (1)				
Conv4+ELU	$32 \times 40 \times 3 \times 3$ (1)				
FC1+ELU	$288 \times 256 \times 1 \times 1$ (1)				
FC2	$256 \times 256 \times 1 \times 1$ (1)				

We consider three settings, namely dense, sparse, and extremely sparse extrinsic rewards [17]. With the dense setting, the agent is randomly spawned in 17 locations (some of which are near the goal). In the sparse and extremely sparse settings, the agent takes at least 270 and 350 steps (actions) to reach the goal state, respectively. Episodes are terminated when the agent reaches the armour or when 2100 time steps are completed.

The agent can perform four discrete actions: *move forward*, $a_t = (1, 0, 0, 0)$; *turn right*, $a_t = (0, 1, 0, 0)$; *turn left*, $a_t = (0, 0, 1, 0)$; and *no action*, $a_t = (0, 0, 0, 1)$. For efficient training, we set the action to be repeated four times [38]. Unless otherwise stated, the total number of steps taken by all workers is 20M, and the values of the hyper-parameters are $\beta = 0.4$, $\gamma = 1.0$, $\eta = 10$, and $m = 3 \times 10^{-5}$.

C. Navigation with Sparse Extrinsic Rewards

Fig. 6 shows that with dense setting, A3C, ICM, and AR4E have good performance, whereas with sparser rewards the proposed model has good performance while the performance of the other models degrades. In settings with sparse and extremely sparse extrinsic rewards, the A3C agent fails to perform navigation as it has insufficient feedback to improve itself during training and the policy cannot be trained efficiently. The ICM agent has good performance with sparse rewards, but it has slow convergence in the environment containing extremely sparse rewards. As the sparsity of the extrinsic rewards increases, AR4E outperforms the other models, indicating that our model effectively learns the features for exploration.

D. Navigation without Extrinsic Rewards

To quantify the benefits of using intrinsic rewards, we perform transfer learning to evaluate if the model learned by self-supervised training encodes meaningful information for navigation. First, we train a model learning features from intrinsic rewards only (i.e. in a self-supervised manner). Then we fine-tune this pre-trained model in *VizDoom* with sparse extrinsic rewards. In the fine-tuning stage, we set $\eta = 1$ to decrease the influence of the intrinsic rewards and to force the network to focus on the extrinsic rewards.

Table II shows the results by changing the total number of steps in pre-training. Compared to training from scratch, the performance decreases when the steps for pre-training

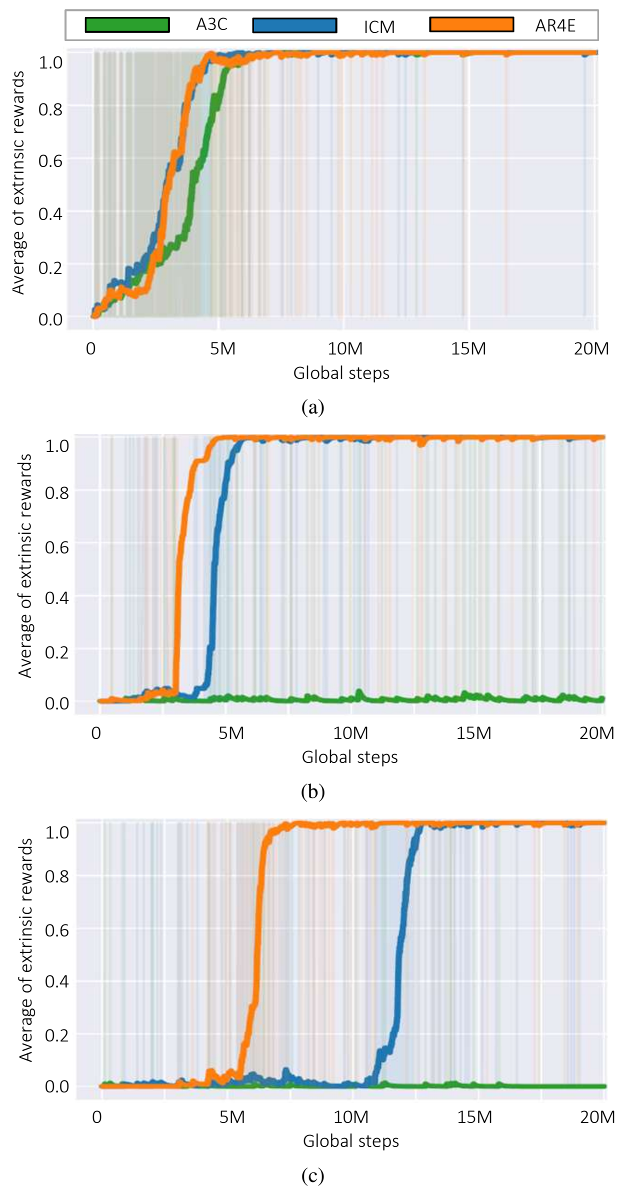


Fig. 6: Extrinsic rewards for an agent trained with AR4E, A3C, and ICM in an environment with (a) dense, (b) sparse, and (c) extremely sparse extrinsic rewards. As the sparsity of the extrinsic rewards grows, the relative improvement of AR4E with respect to the other models increases.

TABLE II: Success ratios when fine-tuning pre-trained models in *VizDoom* with sparse setting. Success refers to the agent achieving the goal during trials within 20M steps.

Pre-training total steps	**Success ratio (%)**
0 (from scratch)	94.45 ± 22.87
0.5M	91.89 ± 27.28
2M	92.01 ± 26.08
10M	$\mathbf{96.32 \pm 18.89}$

TABLE III: Success ratios with different loss functions and their combinations in the Forward (Fwd) and Inverse (Inv) models. Success refers to the agent achieving the goal during trials within 20M steps.

Model		**Success ratio** (%)		
Fwd	Inv	Dense	Sparse	Extremely Sparse
-	L_I	7.87±26.93	0.01±1.12	0.06±2.50
L_F	-	9.12 ± 28.79	0.44 ± 6.60	0.09±2.96
L_{F_1}	L_I	$\mathbf{96.78 \pm 17.63}$	91.33±28.13	87.10±33.51
L_{F_2}	L_I	8.13 ± 27.33	0.0006 ± 0.00	11.25 ± 3.35
L_F	L_I	96.06 ± 19.43	$\mathbf{94.45 \pm 22.87}$	$\mathbf{87.13 \pm 33.48}$

are 0.5M and 2M as self-supervision is insufficient in these cases. The model trained from scratch is outperformed by the pre-trained model with 10M steps as self-supervision produces meaningful features for navigation.

E. Ablation Study

To investigate the contribution of the components within the proposed network, we conducted an ablation analysis by training the agent with different loss functions and computed the success rate during 20M total steps.

Table III shows a good performance when both L_{F_1} and L_I are used and that the performance decreases when extrinsic rewards become sparse. When the network is trained with L_{F_2}, the agent is unable to perform the navigation task, because the triplet ranking loss can be influenced not only by the direction of the feature vector but also by its magnitude. A common approach to avoid this problem is to normalise the feature vector [40], [45].

VI. Conclusion

We proposed an Action Representation Module that expands the dimensions of one-hot codes of input actions, which are then fused with the state-transition network. The network can predict the future state efficiently in an exploration task. Moreover we train the Forward Model with joint regression and triplet ranking loss functions, enabling the network to discriminate predictions from different actions.

As the sparsity of the extrinsic rewards increases, the training of the proposed network converges faster than previous networks [17], [38]. However, similarly to these other networks, our model may repeatedly turn left and right during navigation: in our future work we will investigate loss functions to handle this problem and we will transfer our model to a mobile robot for visual exploration in the real world.

Acknowledgements

We thank Simon Butcher for his support in setting up the environment for simulations in QMUL's Apocrita HPC.

References

[1] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017.

[2] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[3] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," in *ICRA 2018 Workshop on Machine Learning in Planning and Control of Robot Motion*, 2018.

[4] Y. Chebotar, K. Hausman, Z. Su, G. S. Sukhatme, and S. Schaal, "Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2016.

[5] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, "Time-contrastive networks: Self-supervised learning from video," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018.

[6] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2016.

[7] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2000.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Systems (RAS)*, vol. 57, no. 5, pp. 469–483, 2009.

[9] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Comput.*, vol. 3, no. 1, pp. 88–97, 1991.

[10] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011.

[11] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.

[12] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.

[13] J. Ho, J. Gupta, and S. Ermon, "Model-free imitation learning with policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.

[14] S. Mohamed and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2015.

[15] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.

[16] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.

[17] D. Pathak, A. E. Pulkit Agrawal, and T. Darrell, "Curiosity-driven exporation by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017.

[18] S. M. A. Eslami, D. Jimenez Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis, "Neural scene representation and rendering," *Science*, vol. 360, no. 6394, pp. 1204–1210, 2018.

[19] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.

[20] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *Proc. Int. Conf. Learn. Rep. (ICLR)*, 2018.

[21] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[22] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2008.

[23] A. Giusti, J. Guzzi, D. C. Ciresan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots." *IEEE Robot. Autom. Lett. (RA-L)*, vol. 1, no. 2, pp. 661–667, 2016.

[24] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2004.

[25] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2009.

[26] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The Int. J. Robot. Res. (IJRR)*, vol. 34, no. 2, pp. 131–157, 2015.

[27] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015.

[28] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.

[29] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[30] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[31] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018.

[32] P.-Y. Oudeyer, F. Kaplan, and V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, 2007.

[33] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. Int. Conf. Simulation of Adaptive Behavior: From animals to animats*, 1991.

[34] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2015.

[35] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2017.

[36] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv:1507.00814*, 2015.

[37] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, 1992.

[38] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.

[39] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv:1511.07289*, 2015.

[40] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015.

[41] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Machine Intell. (TPAMI)*, vol. 39, no. 4, pp. 640–651, 2016.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv:1606.01540*, 2016.

[44] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, 2016.

[45] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.