Visual Radial Basis Q-Network

Julien Hautot, Céline Teulière, Nourddine Azzaoui

Visual Radial Basis Q-Network

Julien Hautot[1], Céline Teuliere[1], and Nourddine Azzaoui[2]

[1] Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France {julien.hautot,celine.teuliere}@uca.fr
[2] Laboratoire de Mathématiques Blaise Pascal, Clermont-Ferrand 63100, France azzaoui.nourddine@uca.fr

Abstract. While reinforcement learning (RL) from raw images has been largely investigated in the last decade, existing approaches still suffer from a number of constraints. The high input dimension is often handled using either expert knowledge to extract handcrafted features or environment encoding through convolutional networks. Both solutions require numerous parameters to be optimized. In contrast, we propose a generic method to extract sparse features from raw images with few trainable parameters. We achieved this using a Radial Basis Function Network (RBFN) directly on raw image. We evaluate the performance of the proposed approach for visual extraction in Q-learning tasks in the Vizdoom environment. Then, we compare our results with two Deep Q-Network, one trained directly on images and another one trained on feature extracted by a pretrained auto-encoder. We show that the proposed approach provides similar or, in some cases, even better performances with fewer trainable parameters while being conceptually simpler.

Keywords: Reinforcement Learning · State Representation · Radial Basis Function Network · Computer Vision

This paper has been accepted for publication at the 3rd International Conference on Pattern Recognition and Artificial Intelligence, ICPRAI 2022.
DOI: https://doi.org/10.1007/978-3-031-09282-4_27

1 Introduction

Reinforcement learning (RL) has made significant progress in recent years, allowing the use of policy and value-based algorithms on images to perform tasks in fully observable contexts [8,23,29]. However, when the state space is more complex or partially observable, learning remains harder. Solutions are introduced by implementing additional learning modules like Long Short Term Memory (LSTM) [12] or enlarging the networks by adding convolutional layers [14].
Another strategy to facilitate the learning of an agent in a complex visual environment is to reduce the dimension of the state space [20]. This can be accomplished by extracting relevant features from the images using learning based methods such as auto-encoder [18]. The dimension of the state space has a large impact on how these solutions are implemented. Indeed, an auto-encoder needs lots of parameters to encode large states and hence its training is time and energy consuming.
Furthermore using sparse representation in RL has been proven to be efficient [22], this can be done with Radial Basis Function Networks (RBFN) [3,11,24,26] a well know function approximator used for classification and regression tasks [32]. Since its success

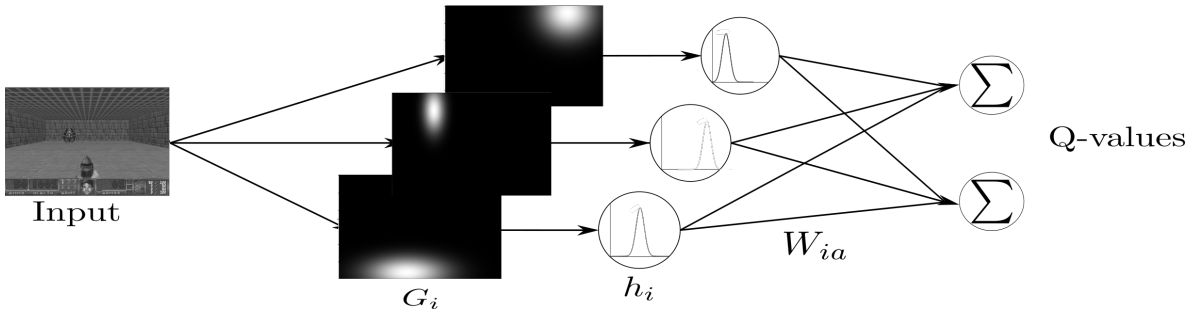

Fig. 1. Visual Radial Basis Q-Network architecture. Input raw image are filtered by spatial Gaussian filters G_i then Gaussian activation h_i are applied on all the filtered pixels and Q-values are calculated by a linear combination of h_i and $W_i a$ the weight between activation h_i and output $Q(S, a)$

in deep learning, the idea of using RBFN in RL has emerged and encountered a great interest especially in non-visual tasks applications [2,5,6]. In all these works, RBFN are however not applied directly on raw pixel, but rather on features extracted from the environment that involve pre-training steps.

Contributions. In this paper, we aim to train an agent in a partially observable environment, with only raw pixel inputs and without prior knowledge, i.e., without pre-training or additional information. We extend the RBFN to extract random sparse features from visual images. Our major contribution is the design and the analysis of a generic features extraction method based on RBFN for visual inputs. We evaluate our extracted features performance in a Q-learning setting and compare the result with state-of-the-art methods. We used Vizdoom as the virtual environment [15] where visual tasks are hard to solve with classic RL algorithms [16].

Paper Organisation. In section 2, we present a brief review of related approaches, then we introduce the theoretical background in section 3. In section 4 we present our RBFN method and analyse the extracted features in section 5. Finally section 6 presents an evaluation of the method on two different visual RL tasks and compare the performances against state-of-the-art approaches.

2 Related Works

State representation learning methods are popular approaches to reduce the state dimension used in visual RL. In the survey [20], Lesort et al. distinguished several categories of approaches for state representation learning. Auto-encoders can be used to extract features by reconstructing input images [18,25,13], or by forward model learning to predict the next state given the actual state and the corresponding action [19]. Learning an inverse model to predict the action knowing states can also be used to extract features that are unaffected by uncontrollable aspect of the environment [33]. Prior knowledge can also be used to constrain the state space and extract conditional features [10]. These learning representation methods typically require to train a network

upstream of an RL algorithm, which can be time-consuming due to the large number of parameters or the environment dimension. Even when the state representation is trained parallel with the agent as in [27] or when unsupervised learning is used as in [1,30], the training is still time-consuming since multiple convolutional networks are trained. Unlike the current state-of-the-art, our method does not rely on pre-training or convolutional network; instead the features are extracted using a combination of Radial Basis Function (RBF) without training nor prior knowledge.

RBFNs have been recently used as feature extractors for 3-D point cloud object recognition [7]. They achieved similar or even better performances compared to the state-of-the-art with a faster training speed. Earlier, in [9] RBF-based auto-encoder showed the efficiency of RBFN as feature extractors prior to classification tasks. One key property of RBFN activation is their sparsity; for each input, only a few neurons will be activated. This is advantageous for RL as it imposes locality of activation avoiding catastrophic interference and keeping stable value which allows bootstrapping [22]. Applying directly RBFN on high dimensional images is challenging and has not been yet much explored in the literature. Other sparse representations, such as extreme machine learning or sparse coding, have been investigated but none have been extended to visual RL as their computing cost tends grows exponentially with the input dimension. One solution in the literature is to reduce the inputs dimension with convolution [21,28].

3 Background

3.1 Reinforcement Learning

In RL we consider an agent interacting with an environment through actions a, states s and rewards r. The environment is modelled as a Markov decision process (MDP). Such processes are defined by the tuple $\langle S, A, T, R\rangle$, where S and A are the states and action spaces, $T : S \times A \to S$ is the transition probability to go from a state $s \in S$ to another state $s' \in S$ by performing an action $a \in A$. $R : S \times A \to \mathbb{R}$ is the reward function. At each time step, the agent will face a state $s \in S$ and choose an action $a \in A$ according to its policy $\pi : S \to A$. The environment will give back a reward $r = R(s, a)$ and the probability of reaching s' as the next state is given by $T(s, a, s')$. The goal of the agent is to find a policy which maximizes its expected return, which we define as the cumulative discounted reward along time-step t, i.e., $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where γ is the discounted rate determining the actual value of future reward.

We chose to evaluate our state representation method in a Q-learning [31] setting. To optimize the policy, Q-learning estimates the state-action value function (Q-value) which represents the quality of taking an action, $a \in A$, in a state, $s \in S$, following a policy π. $Q^\pi(s, a)$ is defined by

$$Q^\pi(s, a) := \mathbb{E}_\pi[R_t | s_t = s, a_t = a]. \tag{1}$$

The optimal policy can then be estimated by approximating the optimal Q-value given by the Bellman equation such that

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a')]. \tag{2}$$

In Deep Q-learning [23], the optimal Q-value is approximated by a neural network parametrized by θ. At each time-step Q-values are predicted by the network, then the agent chooses the best action according to an exploration strategy. The agent experiences $\langle s_t, a_t, s_{t+1}, F\rangle$ are stored into a replay memory buffer D where F is a Boolean indicating if the state is final or not. Parameters of the Q-Network θ are optimized at each iteration i to minimize the loss $L_i(\theta_i)$ for a batch of state uniformly chosen in the replay buffer D, defined by

$$L_i(\theta_i) = \begin{cases} \mathbb{E}_{(s,a,r,s')\sim U(D)}[(r - Q(s,a;\theta_i))^2] & \text{if} \quad \text{F} \\ \mathbb{E}_{(s,a,r,s')\sim Rb}[\left(r + \gamma \max_{a'} Q(s',a';\bar{\theta}_i) - Q(s,a;\theta_i)\right)^2] & \text{otherwise.} \end{cases} \tag{3}$$

To stabilize the training a target network is used with parameters $\bar{\theta}$ which is periodically copied from the reference network with parameters θ.

3.2 Radial Basis Function Network (RBFN)

The network is composed of three layers: the inputs $\mathbf{X}$, the hidden layers ϕ compose of RBFs and the output layers $\mathbf{Y}$. A RBF is a function ϕ defined by its center μ and its width σ as

$$\phi(\mathbf{X}) = \phi(||\mathbf{X} - \mu)||, \sigma), \quad \text{where } ||.|| \text{ is a norm.} \tag{4}$$

In this paper we will consider Gaussian RBF with the Euclidean norm defined as

$$\phi_g(\mathbf{X}) = \exp\left(\frac{||\mathbf{X} - \mu||^2}{2\sigma^2}\right). \tag{5}$$

A RBFN computes a linear combination of N RBF:

$$y_i(\mathbf{X}) = \sum_{j=1}^{N} w_{ij} \cdot \phi(||\boldsymbol{X} - \mu_j||, \sigma_j), \tag{6}$$

where, w_{ij} is a learnable weight between output y_i and hidden RBF layer ϕ_j.

RBFN are fast to train due to the well-localized receptive fields that allow activation of few neurons for one input. The farther the input value is from the receptive field of the neuron, the closer output value is to zero.

4 Method

Our method focuses on the projection of high dimensional input state spaces into a lower dimensional space by combining Gaussian receptive filters and RBF Gaussian activations. Our network architecture is shown in Fig 1. Each receptive field can be

seen as the attention area of the corresponding neuron. The attention area of a neuron i, for a pixel p in a state S (of shape $w \times h$) with coordinates (p_x, p_y) is defined as follows:

$$G_{i,p_x,p_y} = \exp\left(-\left(\frac{(p_x/w - \mu_{x,i})^2}{2\sigma_{x,i}^2} + \frac{(p_y/h - \mu_{y,i})^2}{2\sigma_{y,i}^2}\right)\right), \tag{7}$$

where, $\mu_{x,i}, \mu_{y,i} \in [0, 1]$ define the center of the Gaussian function along spatial dimension and $\sigma_{x,i}, \sigma_{y,i} \in [0, 1]$ are the standard deviations. $\mathbf{G_i} \in \mathcal{M}_{w \times h}$ is the full matrix that defines the spatial attention of a neuron. Given the attention area, the activation of the hidden neuron i is computed using a Gaussian RBF activation function weighted by $\mathbf{G_i}$:

$$\mathbf{h}_i(\mathbf{S}) = \exp\left(-\frac{\sum((\mathbf{S} - \mu_{z,i}) \odot \mathbf{G}_i)^2}{2\sigma_{z,i}^2}\right), \tag{8}$$

where, $\mu_{z,i} \in [0, 1]$ is the center and $\sigma_{z,i} \in [0, 1]$ the standard deviation of the RBF intensity Gaussian activation function. Symbol $\odot$ is the Hadamard product, i.e., the element-wise product. Parameters $\mu_{z,i}$ and $\sigma_{z,i}$ have the same size as the input channel.

To test the efficiency of our extraction we use the extracted features as the state in a Q-learning algorithms where the Q-value will be approximated by a linear combination of N_g Gaussian neurons activations.

$$Q(s, a) = \sum_{i=0}^{N_g} w_{ai} \times h_i(s), \tag{9}$$

Where, w_{ai} is the weight between the action a and the neuron i. On each step the input image is passed to the RBF layer and the computed features are saved in the replay buffer. Then during each training iteration a batch of random features is chosen from the replay memory buffer and the weights are updated using a gradient descent step (Adam [17]) to minimize equation (3).

4.1 Selection of RBF Parameters

There are 6 hyper-parameters for each Gaussian unit. Centers of Gaussian filters $\mu_{x,y}$ and centers of Gaussian activation μ_z are chosen uniformly between 0 and 1 as we need to cover all the state space. The standard deviations $\sigma_{x,y,z}$ influence the precision of the activation of a neuron, i.e., the proximity between pixel intensities weighted by attention area and intensity center of the neuron. In that way RBF layer allows activation of few neurons for a particular image.

The hyper-parameters are chosen at the beginning and never changed during training. After empirical experiments and based on the work of [4] which used also Gaussian attention area, for all the study we choose 2001 neurons for gray and 667 neurons for rgb inputs as each neuron has 3 intensity center, one for each canal. The choice of standard deviations is as follows: $\sigma_z = 1$ and $\sigma_{x,y}$ uniformly chosen in $[0.02 - 0.2]$.

Table 1. Distribution of the active neurons on 1000 states with 20 different RBF parameters initialisation. Per cent of the total number of neurons (2001).

	Basic		Health gathering	
	gray	RGB	gray	RGB
active neurons (aN)	$32 \pm 1\%$	$31 \pm 1\%$	$29 \pm 6\%$	$32 \pm 5\%$

4.2 Vizdoom Scenarios

We evaluate our method on two Vizdoom scenarios to show the robustness of our network to different state dimensions and partially observable tasks. Scenarios are defined as follows:

Basic Scenario. In this task the agent is in a square room, its goal is to shoot a monster placed on a line in front of him. At each episode the monster has a different position while the agent spawns all the time in the same place. In this scenario the agent has the choice between 8 possible actions: move right, move left, shoot, turn left, turn right, move forward, move backward and do nothing. In this configuration the agent can be in a state where the monster is not visible. The agent gets a reward of 101 when the shoot hits the monster, -5 when the monster is missed and -1 for each iteration. The episode is finished when the agent shoots the monster or when it reaches the timeout of 300 steps.

Health Gathering Scenario. In this task the agent is still in a square room but the goal is different, the agent has to collect health packs to survive. It has 100 life points at the beginning of the episode, each iteration the agent looses some life points, if the agent reaches a health pack it gains some life point. The episode ends when the agent dies or when it reaches 2100 steps. Reward is designed as follows: $r_t = \text{life}_{t+1} - \text{life}_t$. Possible actions are move forward, move backward, turn left, turn right, do nothing. All the health packs spawn randomly and the agent spawns in the middle of the room.

5 Analysis of Pattern Activations

In this section we put in evidence the sparsity of our network, and analyse the activation patterns of our neurons and their differences. We then present an example of feature targeted by a RBF neuron.
We generated 1 000 states from both scenarios in gray and rgb with a random policy and a random "skip frame" between 0 and 12 in order to cover all the state space, i.e., each random action is repeated a certain number of time. In each scenario we can classify the neurons into two categories: inactive neurons (iN) which have an activation always inferior to 0.01 and active neurons (aN) which have an activation superior to 0.01 in

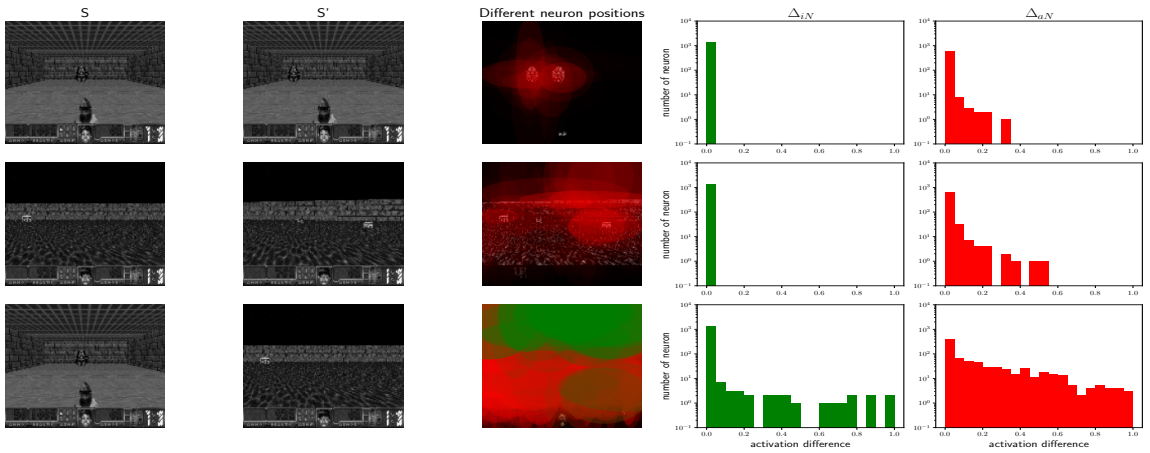

Fig. 2. Comparison of neurons activity on different states for one seed. Histograms represent $\Delta_N = |N(s) - N(s')|$ with N the neurons activation (iN,aN). Neuron positions represent the position of the different neurons with $|S - S'|$ in the background. *Top:* comparison for two close states in basic scenario. *Middle:* comparison of two close states in the health gathering scenario. *Bottom:* comparison between both scenarios

some states. Table 1 gives the percentage of aN on the total number of neurons for 20 different seeds in each configuration. We can see that even with a random generation of Gaussian neurons we have stable distribution, around 30% of activated neurons, across different random seeds, state spaces and scenarios.

Fig 2 highlights the difference in activation for iN and aN between two frames. The first observation is that iN can be used to differentiate two scenarios as some iN of basic scenario become aN in health gathering scenario, those neurons describe the scenario rather than the state. The active neurons, have also more differences between two scenarios than two states, but they have a non negligible number of neurons that differ between the states. These neurons describe particular features as shown in the neuron position column. Indeed, in the basic scenario, the position of different aNs are around the monster, for health gathering there is more differences between both frames so more different neurons. However, we identified 3 main areas, two on both health packs and one on the upper side of the right wall. Between both scenarios, the different iNs and aNs are distributed over the entire image with more differences on the top for the iN as the ceiling is characteristic of the scenario.

In figure 3 we study two different neuron activations, one for each scenario, on 5000 images, and we print the position and orientation of the agent (black arrow) when the neuron is activated or inactivated under a certain threshold. In the basic scenario the neuron fires only when the agent has a wall at its left. Whereas, in the health gathering the neuron is activated when the agent is not facing a wall but rather when the floor is in the neuron attention area. Each aN gives a direct or indirect information about the environment, the monster can be identified by a neuron that gets activated on the wall; indeed, when this neuron will face the monster it will be deactivated indicating that there is something in front of the agent but it is not necessarily the monster. This is why a combination of different activations is needed to get a complete information.

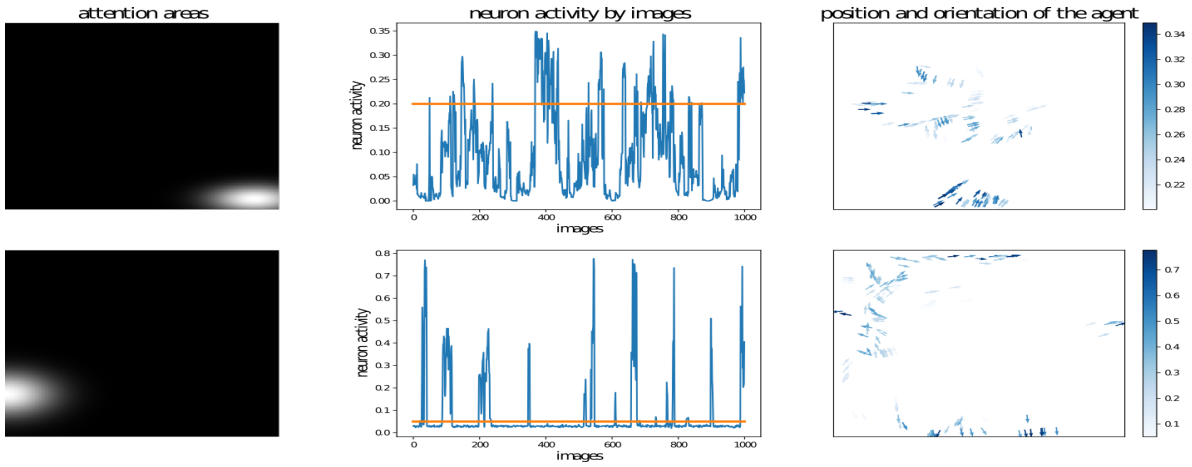

Fig. 3. Study of the activity of two neurons, in both scenarios. Health gathering scenario on top and basic scenario at the bottom. The arrow on the last column represents the agent's position when the neuron activity is below the threshold (orange line), the color represents the amplitude of the activation.

All the neurons formed a pattern activation specific for each scenario, and inside this pattern there is small variation helping to differentiate states in the scenario. We show in the next section that this combination can provide enough information about the environment for Q-learning to predict relevant Q-values.

6 Reinforcement Learning Application

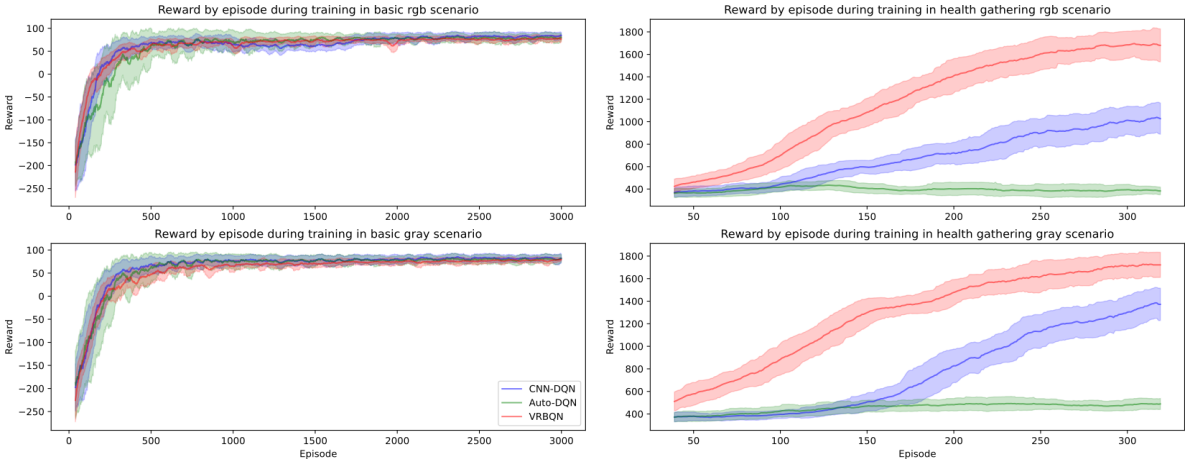

Fig. 4. Comparison of 3 different approaches on basic (left) and health gathering (right) task on 20 different seeds. Rgb inputs for the top and gray for the bottom. Bold line represents the mean and transparent area the standard deviation. The plotted rewards for health gathering scenario is the number of alive step

In the following section we compare our algorithm on two different Vizdoom tasks, basic and health gathering scenarios, with a CNN-DQN, a convolutional network trained

directly on pixels, and an autoencoder-DQN where the RL algorithm is applied on features extracted from a pre-trained CNN auto-encoder. Both CNN are similar to the one used in [23] with one more convolutional layer as the inputs image are bigger, 120×160. Both scenarios have different characteristics, basic scenario has a sparse reward while health gathering reward is dense. Training is done with gray and RGB input state to check the robustness of tested networks to different input spaces. The dimension of the used images is 120×160. We stack two consecutive frames to form a state providing information about movement and we used a "skip frame" of 6 to train faster.
Agents are trained during 100k training steps. The RL part of CNN-DQN and autoencoder-DQN used target networks updated every 1000 iterations. An epsilon-greedy policy is used during training with epsilon starting to decay after 1000 training steps: it goes from 1 to 0.1 in 10 000 training step. A learning rate of 0.00025 and a batch size of 64 are used.
Our method, V-RBQN, does not require a target network as the features are sparse and the approximation of the Q-value is stable and can efficiently bootstrap. In addition, we did not use any exploration strategy and we used a learning rate of 0.01 and a batch size of 256.

For the basic scenario, the left curves in figure 4 show that all the methods converge to the maximum reward. Both methods that trained the agent on extracted features have a larger variance during training than when training directly on images with convolution. This high variance is due to the imprecision of the features which can be too close for 2 different states that require 2 different actions to maximize the Q-value. For example when the monster is not perfectly in front of the weapon, the agent can be in a situation where the features are similar and so the Q-value for shoot, turn left and right are very close, the agent will oscillate between those three actions until it shoots the monster. In the health gathering scenario, which is more difficult to solve with Q-learning [16], the auto-encoder method does not find an optimal solution and V-RBQN performs better that the CNN-DQN. Both state-of-the-art methods converge to a behavior where the agent can get stuck in walls, whereas our method alleviates this problem by having special local and sparse neurons activations. Test results in table 2 demonstrate the robustness of our method to different state spaces. Indeed, for the same number of training steps, our method gets similar results with different inputs size, i.e., corresponding to gray and rgb images, whereas CNN-DQN and autoencoder-DQN have a bigger variance and a smaller reward for rgb input when comparing with gray input for a given time-step.
Finally, we realized another test to highlight the sparsity of our network. We tested the learned weights only on the aNs for each scenario without the contribution of the iNs. Results in table 2 show that the iNs are not required to solve a task. This allows faster forward passes as the quantity of neurons is divided by a factor of three when dealing with a specific scenario.

Table 2. Reward on 1000 episode of basic and health gathering scenario for 3 different algorithms. Score is calculated with mean± standard deviation on 20 different seeds. V-RBQN (only aN) represents the reward when testing without inactive neurons.

Scenario	basic		health gathering	
	gray	rgb	gray	rgb
basicDQN	$\mathbf{86 \pm 6}$	$\mathbf{86 \pm 6}$	1519 ± 751	1092 ± 716
Autoencoder-DQN	86 ± 7	85 ± 7	601 ± 357	438 ± 384
V-RBQN	85 ± 8	85 ± 9	$\mathbf{1809 \pm 543}$	$\mathbf{1778 \pm 573}$
V-RBQN (only aN)	85 ± 8	84 ± 9	1799 ± 571	1690 ± 635

7 Conclusion

In this paper we have extended the Radial Basis Function Network to handle feature extraction in images. We put into light the sparsity and locality properties of our network by analyzing the extracted features. We then show the usefulness of these extracted features by using them as inputs of partially observable visual RL tasks. Despite its simplicity our Visual Radial Basis Q-Network (V-RBQN) gives promising results on two scenarios, i.e., the basic and the health gathering tasks, even with different input channel sizes. Thanks to the sparsity of the extracted features and their locality, the proposed approach outperforms the evaluated baseline while having less trainable parameters, without any exploration strategy nor target network. In addition, the inactive neurons can be omited without hindering the efficiency which allows to reduce the size of the network by a factor of three. One of the research direction will be to find generic RBF parameters, by training them on different types of images to be optimal regardless the input images. Future work will also consider the extension of this approach to continuous action spaces and other RL algorithms.

References

1. Anand, A., et al.: Unsupervised State Representation Learning in Atari. CoRR **abs/1906.08226** (2019), arXiv: 1906.08226
2. Asadi, K., Parikh, N., Parr, R.E., Konidaris, G.D., Littman, M.L.: Deep radial-basis value functions for continuous control. In: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence. pp. 6696–6704 (2021)
3. Broomhead, D.S., Lowe, D.: Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. Tech. rep., ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM) (1988)
4. Buessler, J.L., et al.: Image receptive fields for artificial neural networks. Neurocomputing **144**, 258–270 (2014). https://doi.org/10.1016/j.neucom.2014.04.045
5. Capel, N., Zhang, N.: Extended Radial Basis Function Controller for Reinforcement Learning. CoRR **abs/2009.05866** (2020), arXiv: 2009.05866
6. Cetina, V.U.: Multilayer Perceptrons with Radial Basis Functions as Value Functions in Reinforcement Learning. In: ESANN (2008)
7. Chen, W., et al.: Deep RBFNet: Point Cloud Feature Learning using Radial Basis Functions. CoRR **abs/1812.04302** (2018), arXiv: 1812.04302

8. Christodoulou, P.: Soft Actor-Critic for Discrete Action Settings. CoRR **abs/1910.07207** (2019), arXiv: 1910.07207
9. Daoud, M., et al.: RBFA: Radial Basis Function Autoencoders. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 2966–2973 (2019). https://doi.org/10.1109/CEC.2019.8790041
10. Finn, C., et al.: Deep spatial autoencoders for visuomotor learning. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 512–519 (2016). https://doi.org/10.1109/ICRA.2016.7487173
11. Hartman, E.J., et al.: Layered Neural Networks with Gaussian Hidden Units as Universal Approximations. Neural Computation **2**(2), 210–215 (1990). https://doi.org/10.1162/neco.1990.2.2.210
12. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
13. van Hoof, H., et al.: Stable reinforcement learning with autoencoders for tactile and visual data. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3928–3934 (2016). https://doi.org/10.1109/IROS.2016.7759578
14. Justesen, N., et al.: Deep Learning for Video Game Playing. IEEE Transactions on Games **12**(1), 1–20 (2020). https://doi.org/10.1109/TG.2019.2896986
15. Kempka, M., et al.: ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG). pp. 1–8 (2016). https://doi.org/10.1109/CIG.2016.7860433
16. Khan, A., et al.: Playing first-person shooter games with machine learning techniques and methods using the VizDoom Game-AI research platform. Entertainment Computing **34**, 100357 (2020). https://doi.org/10.1016/j.entcom.2020.100357
17. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
18. Lange, S., Riedmiller, M.: Deep auto-encoder neural networks in reinforcement learning. Proceedings of the International Joint Conference on Neural Networks (2010). https://doi.org/10.1109/IJCNN.2010.5596468
19. Leibfried, F., et al.: Model-Based Stabilisation of Deep Reinforcement Learning. CoRR **abs/1809.01906** (2018), arXiv: 1809.01906
20. Lesort, T., et al.: State representation learning for control: An overview. Neural Networks **108**, 379–392 (2018). https://doi.org/10.1016/j.neunet.2018.07.006
21. Liu, H., et al.: Active object recognition using hierarchical local-receptive-field-based extreme learning machine. Memetic Computing **10**(2), 233–241 (2018). https://doi.org/10.1007/s12293-017-0229-2
22. Liu, V., et al.: The Utility of Sparse Representations for Control in Reinforcement Learning. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 4384–4391 (2019). https://doi.org/10.1609/aaai.v33i01.33014384
23. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
24. Moody, J., Darken, C.: Learning with localized receptive fields. Yale Univ., Department of Computer Science (1988)
25. Nair, A., et al.: Visual Reinforcement Learning with Imagined Goals. CoRR **abs/1807.04742** (2018), arXiv: 1807.04742
26. Park, J., Sandberg, I.W.: Approximation and Radial-Basis-Function Networks. Neural Computation **5**(2), 305–316 (1993). https://doi.org/10.1162/neco.1993.5.2.305
27. Pathak, D., et al.: Curiosity-Driven Exploration by Self-Supervised Prediction. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 488–489. IEEE (2017). https://doi.org/10.1109/CVPRW.2017.70
28. Rodrigues, I.R., et al.: Convolutional Extreme Learning Machines: A Systematic Review. Informatics **8**(2), 33 (2021). https://doi.org/10.3390/informatics8020033
29. Schulman, J., et al.: Proximal Policy Optimization Algorithms. CoRR **abs/1707.06347** (2017), arXiv: 1707.06347
30. Srinivas, A., et al.: CURL: Contrastive Unsupervised Representations for Reinforcement Learning. CoRR **abs/2004.04136** (2020), arXiv: 2004.04136
31. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. Adaptive computation and machine learning series, The MIT Press, second edition edn. (2018)
32. Wu, Y., et al.: Using Radial Basis Function Networks for Function Approximation and Classification. ISRN Applied Mathematics **2012**, 1–34 (2012). https://doi.org/10.5402/2012/324194

33. Zhong, Y., et al.: Disentangling Controllable Object Through Video Prediction Improves Visual Reinforcement Learning. In: 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3672–3676 (2020). https://doi.org/10.1109/ICASSP40776.2020.9053819