Training Agents with Long-range Information in Deep Reinforcement Learning

Hanhua Zhu[1,a)] Tomoyuki Kaneko[1,b)]

Abstract: Capturing the relational information of crucial objects from the input observation has a great influence on the performance of deep reinforcement learning algorithms. However, some related objects may be distributed in a wide area of space or time which makes the computation of relations difficult with traditional local recurrent or convolutional operations. In this paper, we apply non-local operation into the network structure of Importance Weighted Actor-Learner Architecture (IMPALA) to help the model capturing the long-rang dependencies in space, time or even spacetime. The results in five Atari games demonstrate that more information is extracted by computing the relation between important objects with the non-local operation.

Keywords: Deep reinforcement learning, Long-range dependencies, Non-local operation, Atari games.

1. Introduction

Reinforcement learning algorithms have recently been firmly established as feasible approaches in several complex tasks such as Go and Atari games. Combining traditional reinforcement learning algorithms with deep neural networks sheds light on the viability of training directly from high-dimensional game screen inputs [1] [2] [3]. However, the observation, which represents the information from the game environment, will be insufficient and inaccurate if we treat only one game screen data as input. For example, in the catching game such as Pong and Breakout in Atari games [4], a single game frame is unable to reveal the velocity and the direction of the ball which is the most significant information in this kind of game. Attributed to the partial observability, the training will suffer from the problem of incomplete and noisy state information [5] [6] which has a negative impact on the performance of traditional reinforcement learning algorithms.

There are mainly two approaches to alleviate the negative influence caused by this kind of partial observability in current reinforcement learning algorithms. One is using lager receptive fields formed by deep stacks of convolutional neural networks (CNNs) to process stacked consecutive frames [1] [2] [3]. The other one is capturing sequential information with recurrent operations [5]. Both two approaches make use of long-range information to complete the partial observation but their operations are processed in a local range, either in space or time. In order to capture the long-range dependencies, these local operations need to be applied repeatedly which causes several problems such as computationally inefficient and optimization difficulties [7]. Unlike the local convolutional and recurrent operations, non-local operations capture the long-range dependencies efficiently by computing the relations between two positions in space, time or spacetime which induced us to investigate the possible benefits from applying non-local operations into the reinforcement learning algorithms.

In this work, we incorporate non-local neural networks [7] into the structure of a famous deep reinforcement learning model called IMPALA [3] to help the model captures the long-range dependencies. Intending to conduct the non-local operation, we propose two new network structures based on 2D CNNs and 3D CNNs respectively and investigate the effect of non-local operation in Atari games. The results demonstrate that the non-local operation helps the agents extract more information from former frames by capturing long-range dependencies between important objects and thus improve the performance in some of the games. On the other hand, overvalued outdated information also has a negative influence on some games.

2. Related Work

Capturing long-range dependencies is important in deep neural networks as the crucial information may distribute over a wide area of space, time or even spacetime when the dimension of the input increases. Intending to obtain the velocity and the direction of the objects in the game, Deep Q-Networks [1] receives four consecutive frames instead of a single game frame as input and computes the interactions between the same object's positions in four frames with local convolutional operations. The first convolutional operation captures local spacetime dependencies and with the CNNs becoming deeper, more information distributed in long-range space will be covered in the increased recep-

[1] Graduate School of Interdisciplinary Information Studies, The University of Tokyo

[a)] zhu-hanhua@g.ecc.c.u-tokyo.ac.jp

[b)] kaneko@acm.org

tive fields. However, for some games, the observation from four frames is still insufficient and thus long-range sequential information should be taken into account. Deep Recurrent Q-Networks [5] incorporates LSTM [8] into the network structure of DQN to help the model capture the dependencies in time. Although DRQN processes both convolutional and recurrent operations, space and time dependencies are computed separately and the relations between some positions may be forgotten during the training.

Relational reinforcement learning [9] combines self-attention [10] with IMPALA for computing non-local interactions. The model computes interactions between any two positions regardless of their positional distance. However, it only takes spatial information into account. For some more complex tasks that need long-range time information such as StarCraft II, an additional LSTM is needed.

3. Method

3.1 Deep Reinforcement Learning

Among a family of state-of-the-art reinforcement learning algorithms, we chose a computation efficient algorithm IMPALA to train the agents. IMPALA uses resources more efficiently in a single machine and also is suitable for large scale distributed training. Instead of communicating the gradients of parameters between workers and a central parameter server, the actors in IMPALA communicate trajectories which consist of states, actions and rewards with a centralized learner. With these trajectories, the learner is able to access the information necessary for the training and therefore the parameter updating and the trajectory generation can be conducted in parallel. For the purpose of reducing the harmful discrepancy between the latest policy updated by the learner and the out-of-date policy used in trajectory generation, IMPALA uses a V-trace off-policy actor-critic algorithm.

The goal of IMPALA is to learn a policy π and a value function V^{π}. As the learning is off-policy, the algorithm need to learn the V^{π} of policy π which is usually called target policy by using trajectories generated by a different policy μ called behaviour policy. When given a trajectory $(x_t, a_t, r_t)_{t=s}^{t=s+n}$ which consists of sequences of state x, the reward r and the action a selected by policy π starting from time s to time $s+n$, the n-steps V-trace target for $V(x_s)$ at state x_s is defined as:

$$v_s = V(x_s) + \delta_s V + \gamma c_s (v_{s+1} - V(x_{s+1})), \qquad (1)$$

where $\delta_s V = \rho_s(r_s + \gamma V(x_{s+1}) - V(x_s))$ represents temporal difference for V and and γ is the discount factor. Truncated importance sampling weights $\rho_s = \min(\overline{\rho}, \dfrac{\pi(a_s|x_s)}{\mu(a_s|x_s)})$, $c_s = \min(\overline{c}, \dfrac{\pi(a_s|x_s)}{\mu(a_s|x_s)})$. ρ_s defines the fixed point of the update and $\overline{\rho}$ directly influences the value function the algorithm converge to. In details, the algorithm will converge to the value function of the target policy if $\overline{\rho}$ is infinite while close to the value function of the behavior policy when $\overline{\rho}$ is close to zero. $c_s ... c_{t-1}$ measure how much a temporal difference at time t will influence the update at a previous time s and $\overline{c}$ influences the speed of convergence.

At training time s, the parameters of value function θ are updated by gradient descent in the direction of

$$(v_s - V_\theta(x_s))\nabla_\theta V_\theta(x_s), \qquad (2)$$

which adjusts the parameters θ to reduce the difference between the output $V_\theta(x_s)$ and the V-trace target v_s. The policy parameters ω are updated by policy gradient in the direction of

$$\rho_s \nabla_\omega \log \pi_\omega(a_s|x_s)(r_s + \gamma v_{s+1} - V_\theta(x_s)), \qquad (3)$$

which adjusts the parameters ω to increase the log-probability of chosen action which lead to a higher state-action value and decrease the action which have a low state-action value. $V_\theta(x_s)$ is used as a baseline to reduce the variance of the policy gradient estimate. Furthermore, an additional entropy bonus is added to prevent premature convergence:

$$-\nabla_\omega \sum_a \pi_\omega(a|x_s) log \pi_\omega(a|x_s). \qquad (4)$$

The overall update is conducted by summing these three gradient with appropriate coefficients.

3.2 Non-Local Neural Networks

Non-local operations capture the long-range dependencies by computing the response at a position as a weighted sum of all positions in space, time or even spacetime in the input feature maps. A generic non-local operation is defined as:

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j), \qquad (5)$$

where i is the index of a position whose response to be computed and j is the index which represents all possible positions. $\mathbf{x}$ is the input feature and $\mathbf{y}$ is the output feature of the same size as $\mathbf{x}$. Function f computes the relation between position i and position j while function g computes the representation of the position j. The output is normalized by a factor $C(\mathbf{x})$.

Among a family of versions of function f, we chose embedded gaussian function to compute the relation between two positions. The function f is defined as:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}, \qquad (6)$$

where two embedings $\theta(\mathbf{x}_i) = W_\theta \mathbf{x}_i$ and $\phi(\mathbf{x}_j) = W_\phi \mathbf{x}_j$. The normalized factor is set as $C(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$ which means for a given i, $\frac{1}{C(\mathbf{x})} f(\mathbf{x}_i, \mathbf{x}_j)$ becomes the softmax computation along the dimension j. In our experiments, function g is set as a linear embedding $g(\mathbf{x}_j) = W_g \mathbf{x}_j$ for simplicity. Here W_θ, W_ϕ and W_g are the weight matrices to be learned.

The non-local operation is wrapped into a non-local block which is defined as:

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i, \qquad (7)$$

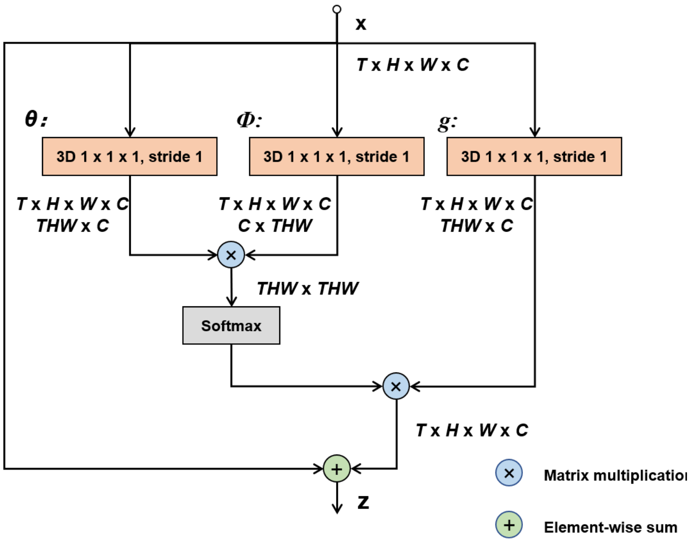


Fig. 1: Non-local block. T, H, W, C represent the number of frames, the height of a single frame, the width of a single frame and the number of channels respectively.

where $\mathbf{y}_i$ is given in Eq. 5. Similar to the ideas in residual learning [11], this equation ensures that the performance will be no worse than the network without the non-local operation. A non-local block is illustrated in Fig. 1. As shown in Fig. 1, the embeding functions θ, ϕ and g are implemented by 3D CNNs [12] with kernel $1 \times 1 \times 1$ and stride 1. Different to the structure used in research [7], the number of channels in θ, ϕ and g are same the to the number of channels in input x.

3.3 Visual Rationalization

In order to explain how the non-local operations help capture long-range dependencies, we imply sensitivity maps which is widely used to highlight the important regions of input images in image classification tasks [13]. A sensitivity map $M_c(x)$ is computed by

$$M_c(x) = \frac{\partial S_c(x)}{\partial x}, \tag{8}$$

where $S_c(x)$ represents the activation function for class c with the input image x. ∂S_c is the derivative of S_c which shows the importance of each pixel of x to the classification score for class c. In our work, we compute $\partial S_a(x)$ instead of $\partial S_c(x)$ where $S_a(x)$ represents the score of chosen action a before the last softmax layer. After we get the $\frac{\partial S_a(x)}{\partial x}$, a ReLU [14] operation is applied to filter the features that have a negative influence on the action of interest. So the sensitivity map used in our experiments is finally defined as:

$$M_a(x) = ReLU(\frac{\partial S_a(x)}{\partial x}). \tag{9}$$

4. Experiments

We compared different network structures with original IMPALA in several Atari games produced by OpenAI Gym [15]. We also conducted visual explanations to show how different network structures capture the long-range dependencies.

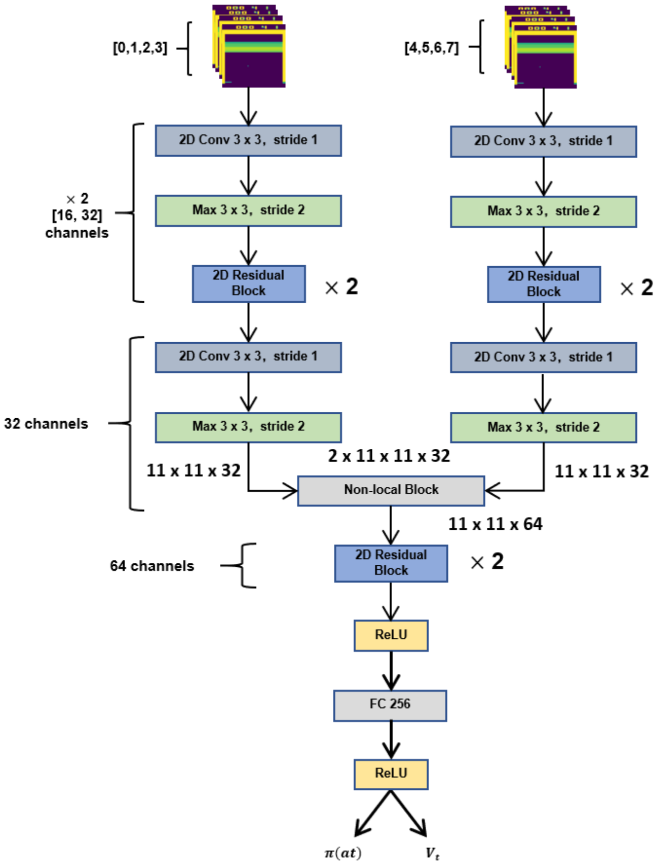


Fig. 2: 2D network structure with a non-local block.

4.1 Details of Training

In this work, we used the residual network [11] as the basic unit to construct our model. A residual block consisted of two convolution layers and each of them was followed by a ReLU [14] unit. We conducted our experiments with six different network structures.

- The original IMPALA with the same setting as the one proposed in research [3].
- The IMPALA-8 which has the same structure as the original IMPALA except for the number of input frames, treated 8 stacked consecutive frames as input while the input of original IMPALA was 4 stacked consecutive frames.
- The 2D network structure which used two independent networks to collect the information from the first four frames and the last four frames separately.
- The 2D with a non-local block which applied non-local operation to compute the relations between the information from the first four frames and the last four frames. Its network structure is shown in Fig 2.
- The 3D network structure where 2D CNNs were replaced by 3D CNNs for capturing more information in time positions.
- The 3D with a non-local block whose network structure

Table 1: The network structure of 3D CNNs with non-local block.

Layer			output size
3D conv	2×3×3, stride 2,1,1		4×84×84×16
max pool	1×3×3, stride 1,2,2		4×42×42×16
res	$\left[\begin{array}{c} 1 \times 3 \times 3, stride1 \\ relu \end{array}\right]$	× 2	4×42×42×16
res	$\left[\begin{array}{c} 1 \times 3 \times 3, stride1 \\ relu \end{array}\right]$	× 2	4×42×42×16
3D conv	2×3×3, stride 2,1,1		2×42×42×32
max pool	1×3×3, stride 1,2,2		2×21×21×32
non-local block			2×21×21×32
res	$\left[\begin{array}{c} 1 \times 3 \times 3, stride1 \\ relu \end{array}\right]$	× 2	2×21×21×32
res	$\left[\begin{array}{c} 1 \times 3 \times 3, stride1 \\ relu \end{array}\right]$	× 2	2×21×21×32
3D conv	2×3×3, stride 2,1,1		1×21×21×32
max pool	1×3×3, stride 1,2,2		1×11×11×32
res	$\left[\begin{array}{c} 1 \times 3 \times 3, stride1 \\ relu \end{array}\right]$	× 2	1×11×11×32
res	$\left[\begin{array}{c} 1 \times 3 \times 3, stride1 \\ relu \end{array}\right]$	× 2	1×11×11×32
relu			1×11×11×32
fc	256		256
relu			256
fc	action number		action number

is shown in Table 1. The non-local block computed the relations between the information captured by former 3D CNNs.

It should be noted that the 2D and the 3D network structure was the same as the structure shown in Fig 2 and table 2 respectively but without the non-local block so that we could make a comparison between these paired structures and observe the effect of the non-local block.

All the image date in the training was converted from RGB color space to Gray color space and resized into 84×84 at first. The last four frames were stacked together as the observation of current state in original IMPALA while the last eight frames were stacked as the observation for the other five network structures. The hyperparameters used in the training of the four networks were totally same to the setting in research [3]. We trained our agents in the architecture of 32 workers and 1 centre learner on a single machine with two NVIDIA 1080TiGPUs and one 16-core 32-processor AMD CPU.

The results of original IMPALA were the average test scores of three different agents trained independently by the same model with 200 million environment steps while the results of the other five network structures were the score of a single training in the same setting. The testing environment was similar to the training environment, except that agents only had a single life during training while games over when agents lost the standard number of lives in the testing environment. The average score over 200 plays was recorded as the final score of the agents in the test environment.

4.2 Atari games

The Atari games have been used to evaluate most recent deep reinforcement learning algorithms. We chose five Atari games as our experiment environments and the results are shown in Table 2. The eighth column of the table shows the results in research [3] which are different from the results of our IMPALA shown in the second column. We investigate that the differences are introduced by some parameters such as the number of workers that are not written in the paper [3]. Because our experiments are based on our implementation, we only concentrate on the results of our six network structures. From Table 2, we can find that observing from more frames has a positive influence on the training of agents in four of five games. Collecting information from two independent networks improves the performance in three games but also affects the training of two games, Centipede and Krull, in a negative way. The 3D network structure has a similar performance, better in Breakout and Seaquest, a little worse in Amidar, Centipede and Krull compared to the IMPALA-8. As the 2D and 3D network structure process the input frames in totally different ways compared to the original IMPALA, we think the difference of performances in five different games is reasonable. When we concentrate on the effect of the non-local operation, we can find that in Centipede, Krull and Seaquest, the non-local operation brings an improvement in the final score but decreases the scores in Amidar and Breakout.

The training procedure is shown in the Fig 3. The scores shown in the figures are computed by the exponential moving average [16] whose decrease coefficient is set to 0.9 and then be smoothed with the Savitzky-Golay filter [17]. In Centipede and Seaquest, the IMPALA showed unstable performance and the results fluctuated in a non negligible range. In Centipede, the results of three independent training were 5691.00, 7873.71 and 6586.15. And in Seaquest the results were 1433.40, 1903.90 and 1557.40. We think the fluctuation also happened during the training of the other five network structures and we will increase the number of experiments to get more accurate results in the future.

Table 2: The score of agents trained by different network structures.

Game	IMPALA	IMPALA-8	2D	2D with non-local	3D	3D with non-local	IMPALA [3]
Amidar	757.45	791.26	**1106.30**	951.96	592.28	480.24	1554.79
Breakout	473.00	473.14	491.39	470.07	**500.67**	449.46	787.34
Centipede	**6716.95**	5317.63	1920.87	4645.31	4209.75	5522.83	11049.75
Krull	5966.99	**7755.75**	5798.38	7609.27	6755.00	7282.88	8147.70
Seaquest	1631.57	1984.10	2562.35	2612.40	2130.60	**2706.10**	1753.20

4.3 Visual Rationalization

To explain how the additional non-local block influences the training, we visualized the important regions where the agents base their decisions in the game screen by sensitivity map. If the trained agents are able to capture the long-range dependencies, the regions should cover the crucial objects in all input frames. The results are shown in Fig 5. As the original IMPALA treated four frames as input, the images of IMPALA shown in Fig 5 were taken from two observations separated from eight consecutive frames which were the one observation of the other network structures. As shown in Fig 5, the IMPALA concentrates on the important objects, the dangerous fish and the people who need to be rescued, successfully when observes from four frames but loses the objects when the input is increased to eight frames. In the 2D CNNs based network structures, both two structure covers the important objects successfully which corresponds to the similar results shown in Table 2. In contrast to the failure of capturing the important objects in former frames with baseline 3D network structure, the long-range dependencies are captured by computing relations between important objects in different frames with the non-local operation in the 3D network with non-local structure, which helps the agent concentrate on the objects in each frame successfully.

These results provide evidence that the non-local operation helps the agent capture more information from the input frames, especially from the former frames, which may be also harmful to some games. We can find that the agents trained with the non-local operation consider the information from former frames as important as the newest frame in Fig 4, which may confuse the agent and result in a wrong action decided based on outdated information. We think this is the reason for the decrease in game Amidar and Breakout.

5. Discussion

In this paper, we investigate the effectiveness of applying the non-local operation into the network structures of IMPALA to help the model capture the long-range dependencies. We also propose two network structures based on 2D and 3D CNNs respectively and test their performance in Atari games. The results show that the non-local operation helps the agents capture the long-range dependencies between different frames and thus influence the performance in a positive way. On the other hand, the negative influence is also observed which induced us to conduct more experiments in the future. As the increased number of parameters makes this method computational expensive, finding a more effective network structure is necessary for future work. We choose five totally different Atari games in this paper, which makes it difficult to demonstrate the effect of capturing long-range dependencies. So we are also planning to conduct experiments in more suitable environments.

References

[1] Mnih et al.: Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, p. 529 (2015).

[2] Mnih et al.: Asynchronous methods for deep reinforcement learning, *International conference on machine learning*, pp. 1928–1937 (2016).

[3] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I. et al.: Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, *arXiv preprint arXiv:1802.01561* (2018).

[4] Bellemare, M. G., Naddaf, Y., Veness, J. and Bowling, M.: The arcade learning environment: An evaluation platform for general agents, *Journal of Artificial Intelligence Research*, Vol. 47, pp. 253–279 (2013).

[5] Hausknecht, M. and Stone, P.: Deep recurrent q-learning for partially observable mdps, *2015 AAAI Fall Symposium Series* (2015).

[6] Wierstra, D., Foerster, A., Peters, J. and Schmidhuber, J.: Solving deep memory POMDPs with recurrent policy gradients, *International Conference on Artificial Neural Networks*, Springer, pp. 697–706 (2007).

[7] Wang, X., Girshick, R., Gupta, A. and He, K.: Non-local Neural Networks, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 7794–7803 (2018).

[8] Hochreiter, S. and Schmidhuber, J.: Long short-term memory, *Neural computation*, Vol. 9, No. 8, pp. 1735–1780 (1997).

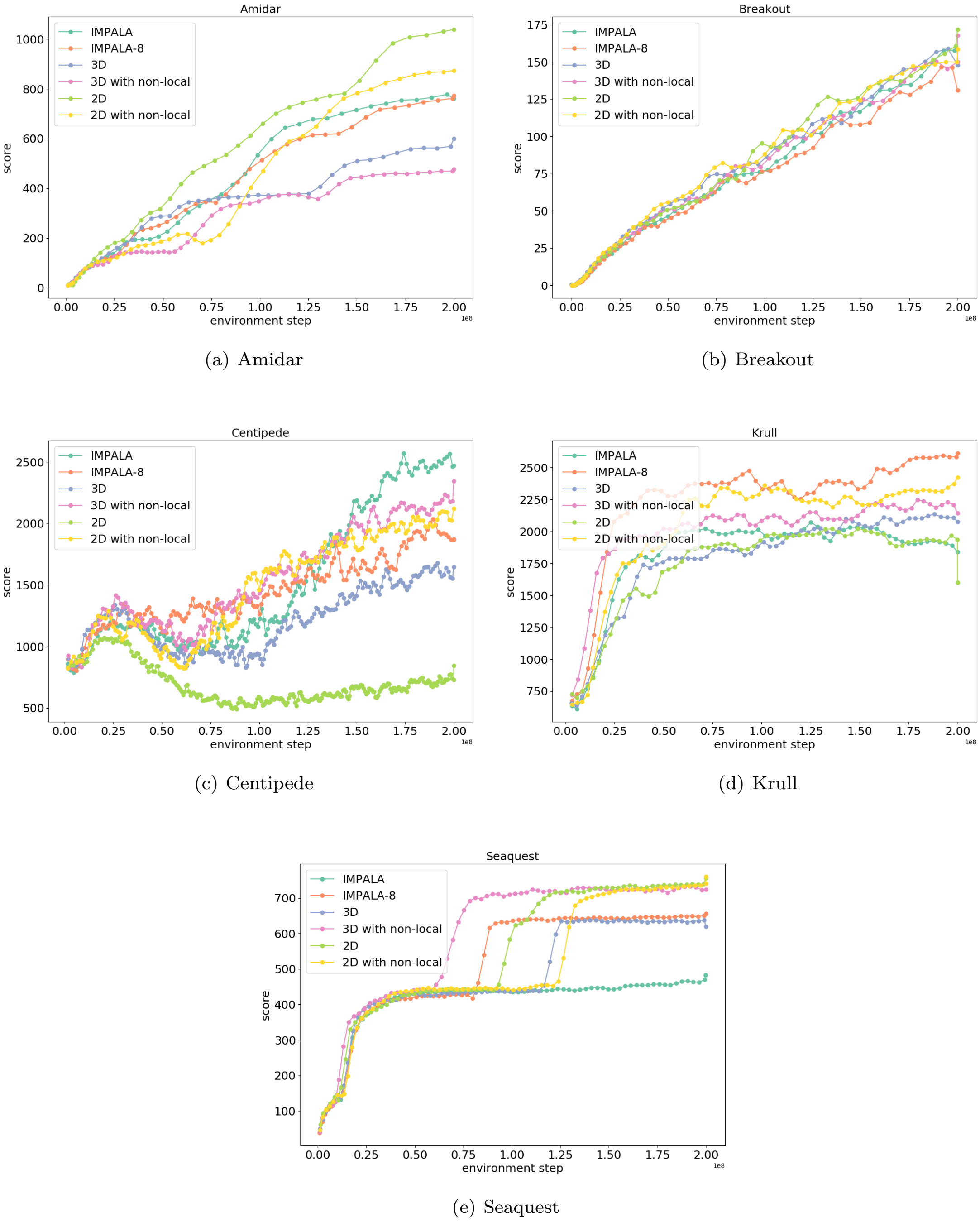
[9] Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E. et al.: Relational deep reinforcement learning, *arXiv preprint arXiv:1806.01830* (2018).

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I.: Attention is all you need, *Advances in neural information processing systems*, pp. 5998–6008 (2017).

[11] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (2016).

[12] Ji, S., Xu, W., Yang, M. and Yu, K.: 3D convolutional neural networks for human action recognition, *IEEE transactions on pattern analysis and machine intelligence*, Vol. 35, No. 1, pp. 221–231 (2012).

[13] Smilkov, D., Thorat, N., Kim, B., Viégas, F. and Wattenberg, M.: Smoothgrad: removing noise by adding noise, *arXiv preprint arXiv:1706.03825* (2017).



(a) Amidar

(b) Breakout

(c) Centipede

(d) Krull

(e) Seaquest

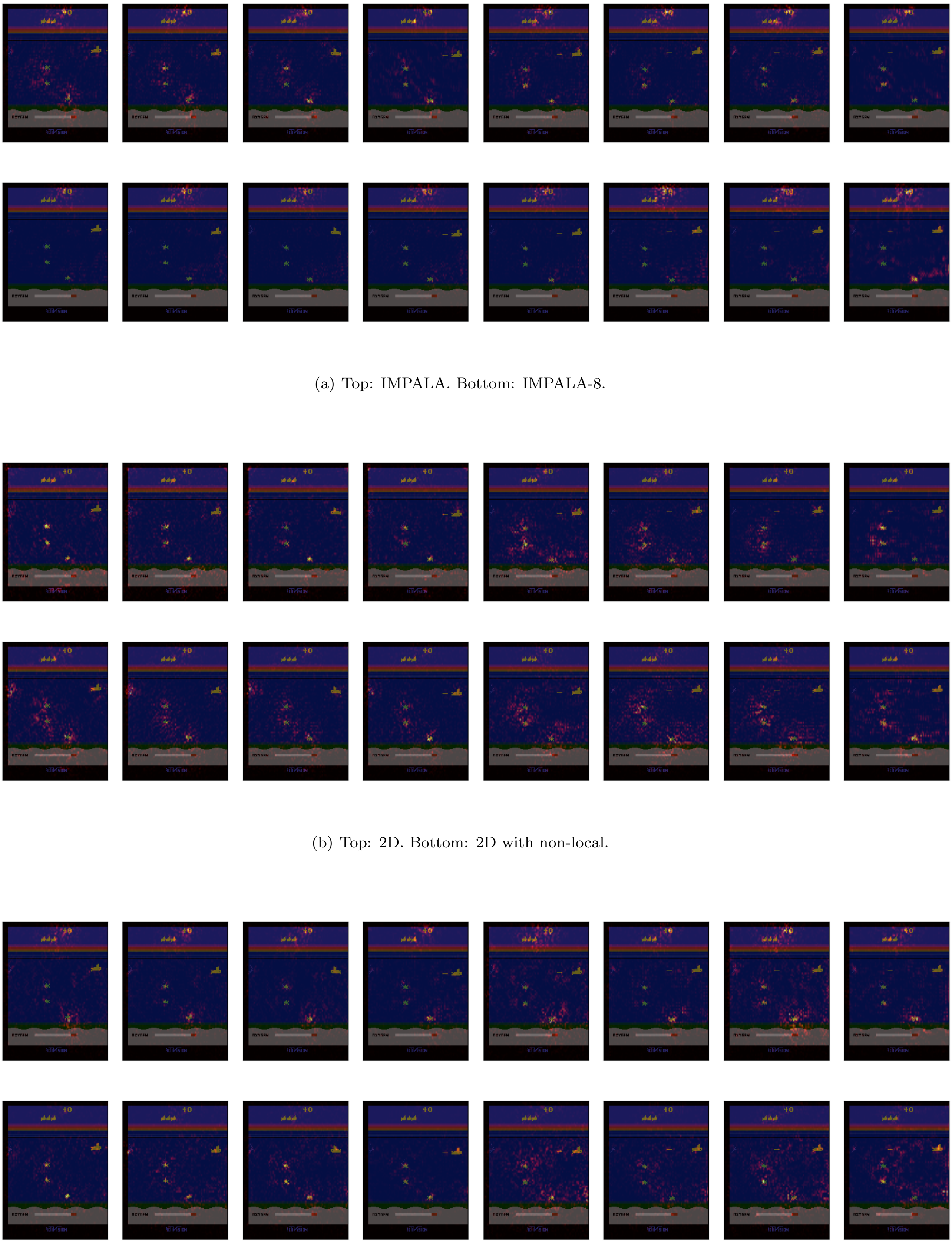
Fig. 3: The training procedure of five games with six different network structures.

[14] Glorot, X., Bordes, A. and Bengio, Y.: Deep sparse rectifier neural networks, *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323 (2011).

[15] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W.: OpenAI Gym (2016).

[16] Lawrance, A. and Lewis, P.: An exponential moving-average sequence and point process (EMA1), *Journal of Applied Probability*, Vol. 14, No. 1, pp. 98–113 (1977).

[17] Press, W. H. and Teukolsky, S. A.: Savitzky-Golay smoothing filters, *Computers in Physics*, Vol. 4, No. 6, pp. 669–672 (1990).



(a) Top: IMPALA. Bottom: IMPALA-8.

(b) Top: 2D. Bottom: 2D with non-local.

(c) Top: 3D. Bottom: 3D with non-local.

Fig. 4: The important regions in eight consecutive environment frames of game Seaquest from agents trained by six different network structures.