Object-Oriented State Abstraction in Reinforcement Learning for Video Games

1st Yu Chen
Guanghua School of Management
Peking University
Beijing, China
yu.chen@pku.edu.cn

2nd Huizhuo Yuan
School of Mathematical Sciences
Peking University
Beijing, China
yuanhz@pku.edu.cn

3rd Yujun Li
Department of Computer Science and Engineering
Shanghai Jiao Tong Univeristy
Shanghai, China
liyujun145@gmail.com

***Abstract*—We present a novel method to obtain object-oriented state representations for video games. Inspired by the mechanism of attention to objects in human vision, we try to make the agents automatically detect the important objects during the learning process. The detection is directed by the Q value based on the abstract state representations. The process does not require human prior knowledge and provides a faster and lighter way for AI playing games. We present empirical results on the *Battle City* game to validate our method. In comparison with raw images input and other preprocessing methods, our approach achieves better final results and uses smaller state space.**

***Index Terms*—reinforcement learning, video games, state abstraction, object detection**

I. Introduction

Many studies have shown that in the real world, human eyes automatically focus on objects and filter out some background information [1]–[3]. At the same time, some objects are more crucial than others, such as obstacles and boundaries. Inspired by the above facts, we try to design a mechanism that enables reinforcement learning (RL) agents to detect objects in image-input tasks automatically.

Image is one of the most common input forms for RL tasks in many games [4]–[6]. Image-input data are usually high-dimensional and can precisely capture game details. However, they also lead to a large state space of the MDP and add extraneous noise. An excessive number of states require large samples and may lead to unstable training results for RL. A natural solution is to reduce the state space, leaving only features that are critical to the learning process. The procedure of obtaining an efficient state space is called *state abstraction*.

Our main contribution is to enable RL agents to automatically detect new objects, abstract the state space based on the detected objects and learn policy in the object-oriented state space. Our approach makes it easier to train RL game agents, and the procedure is more robust to interfering objects and noise.

II. Preliminaries

A *ground* MDP is defined as a 5-tuple $M \equiv (\mathcal{S}, \mathcal{A}, \mathbb{P}, R, \gamma)$, where $\mathcal{S}$ is a set of states that use image representation, and $\mathcal{A}$ is a set of actions. For $\forall s, s' \in \mathcal{S}, a \in \mathcal{A}$, $\mathbb{P}(s' \mid s, a)$ is the state transition probability from s to s' by taking action a. Let $R_t = R(s, a)$ be the reward function which is normalized to $[-1, 1]$. The objective is to find a *policy* $\pi : \mathcal{S} \mapsto \mathcal{A}$ by maximizing the discounted future reward $\sum_{k\geq 0} \gamma^k R_{t+k}$ with $\gamma \in [0, 1)$.

Let $\phi : \mathcal{S} \mapsto \overline{\mathcal{S}}$ be a mapping function from a ground state space to an abstracted one. Given ϕ and s, we define the inverse image set $\mathcal{I}_\phi(\overline{s}) = \{s \mid \phi(s) = \overline{s}, s \in \mathcal{S}\}$ where $\overline{s} := \phi(s)$. An *abstract* MDP of M is defined as $\overline{M} \equiv (\overline{\mathcal{S}}, \mathcal{A}, \overline{\mathbb{P}}, \overline{R}, \gamma)$, where $\overline{\mathcal{S}} = \{\overline{s} \mid \overline{s} = \phi(s), s \in \mathcal{S}\}$ is the abstracted state space. $\mathcal{A}$ and γ are the same as in the *ground* MDP.

Given a policy π for a ground MDP, the Q-function is defined as $Q(s, a) = \mathbb{E}(\sum_{k\geq 0} \gamma^k R_{t+k} \mid s_t = s, a_t = a)$. The Q-function decomposes into the *Bellman equation* and $Q^{t+1}(s, a)$ can be updated by $R(s, a) + \gamma \max_{a'} Q^t(s', a')$.

For an abstract MDP, we use $\overline{Q}(\overline{s}, a), \overline{\pi}, \overline{Q}^*(\overline{s}, a)$ and $\overline{\pi}^*$ to denote its Q function, policy, and optimums. Similarly, the abstracted $\overline{Q}^{t+1}(\overline{s}, a)$ can be updated by $R(s, a) + \gamma \max_{a'} \overline{Q}^t(\overline{s'}, a')$.

Object detection is used to identify a specific pixel area over images and classify them into different objects. We learn from the practice of R-CNN [7] to perform detection in RL. The basic idea is to propose a bunch of boxes for each image, and check whether each box corresponds to an object.

We enable RL agents automatically recognize objects under the guidance of Q function. Specifically, improvements in object detection accuracy may result in a better approximation function and an increase in future rewards.

III. Methodology

A. Object oriented state abstraction

Throughout this subsection, we assume the coordinates of a total of m objects in K classes are known and demonstrate how to construct ϕ. For each s, let $\mathfrak{B}_1(s), \cdots, \mathfrak{B}_m(s)$ be the bounding boxes for the m objects. We first divide image s into an $L \times L$ grid, and then represent the K classes by K matrices of size $L \times L$. For the k-th matrix, element (i, j) is 1 if there is an object of class k at the grid (i, j) in s, and otherwise 0. As a result, ϕ is $\mathcal{S} \mapsto \{0, 1\}^{L\times L\times K}$. This abstract mapping function can significantly reduce the state space size.

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

B. Automatic Object Detection

We present an iterative algorithm to detect objects automatically for image-input games, and we call it *automatic object detection reinforcement learning* (AOD-RL). In each iteration, the algorithm performs the following steps. The procedure is described in Algorithm 1, and see (a) to (f) in Fig. 1 for an intuitive explanation.

Algorithm 1 AOD-RL

Input: Number of boxes N, dimensions of the autoencoder p, number of trajectories T and number of clusters K.

procedure AUTOMATIC OBJECT DETECTION RL
  $t \leftarrow 0$, $\mathcal{C}^0 \leftarrow$ randomly initialized k-means centroids, randomly initialize Q approximator.
  repeat
    $t \leftarrow t+1$
    for trajectory in $\{1, \cdots, T\}$ **do** ▷
      Interact according to $\overline{Q}_{t-1}(\phi_{t-1}(s), a)$.
      Update $\overline{Q}_{t-1}(\phi_{t-1}(s), a)$ by (1)
    Sample a set of states $\mathcal{M}_s$ from the replay memory.
    $\mathcal{O}^t \leftarrow \{\mathfrak{B}_i(s) \mid 1 \le i \le N, s \in \mathcal{M}_s\}$
    $\mathcal{X}^t \leftarrow \{\text{Autoencoder}(o) \in \mathbb{R}^p \mid o \in \mathcal{O}^t\}$.
    $\mathcal{C}^t \leftarrow$ CENTROIDS UPDATE($\mathcal{X}^t$, $\mathcal{C}^{t-1}$, $\overline{Q}_{t-1}$)
    $\phi_t(s) \leftarrow \Phi(s, \mathcal{C}^t)$. Copy the parameters of $\overline{Q}_{t-1}(\phi_{t-1}(s), a)$ to $\overline{Q}_t(\phi_t(s), a)$
  until terminated

Clustering object candidates. First, the agent collects image states $\{s \in \mathcal{S}\}$ from the environment according to $\overline{Q}$ in the previous iteration. For each s, it proposes N bounding boxes $\mathfrak{B}(s)$ of candidate objects. Second, an autoencoder is used to project the bounding boxes onto a lower dimensional space, and we collect the encoded data in $\mathcal{X}_t$. Then AOD-RL executes a centroids update procedure to get clustering centroids of objects, which is demonstrated in the next subsection. Given the new obtained centroids $\mathcal{C}_t$, a bounding box $\mathfrak{B}(s)$ is assigned to the i-th cluster if $\forall j \neq i, \|x - c_i\| \le \|x - c_j\|$, where x is the encoded $\mathfrak{B}(s)$ and $c_i, c_j \in \mathcal{C}_t$.

Update $\overline{Q}$ function. Once the clustering step assigns each bounding box a label, AOD-RL can interact with the environment and updates a new $\overline{Q}(\phi(s), a)$ according to the newly obtained ϕ.

AOD-RL iteratively runs the above two steps. The choice of K in k-means is not critical, as long as it is greater than the number of object classes. Irrelevant classes will be filtered out by Algorithm 2. We define the abstraction mapping as $\phi_t(s) = \Phi(s, \mathcal{C}_t)$, where $\Phi(s, \mathcal{C})$ refers to the grid mapping given class centroids $\mathcal{C}$. The abstracted $\overline{Q}$ function as $\overline{Q}(\phi_t(s), a)$. Within step t, the corresponding Bellman update of $\overline{Q}(\phi_t(s), a)$ is defined as

$$(1-\alpha)\overline{Q}(\phi_t(s), a) + \alpha\big(R(s,a) + \gamma \max_{a'} \overline{Q}(\phi_t(s'), a')\big). \quad (1)$$

As we can see, in AOD-RL the key to automatic objects detection is the clustering centroids obtained by the k-means algorithm. The k-means procedure may miss important objects due to random factors, and may also obtain some classes that are irrelevant to the learning task. The following subsection addresses these issues.

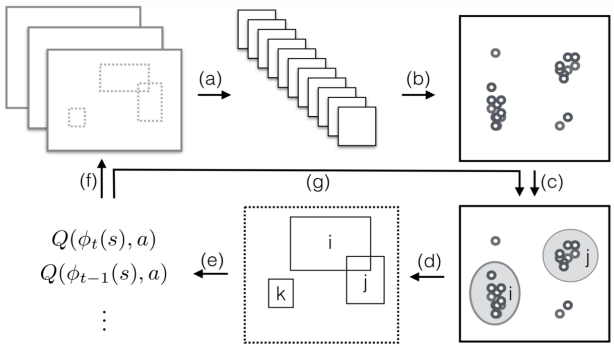


Fig. 1: An intuitive explanation of AOD-RL. (a) Randomly propose bounding boxes, (b) autoencoder, (c) k-means, (d) new abstraction ϕ, (e)learn abstracted Q-function, (f) collect new experience states and (g) update centroids and filter out irrelevant objects (Algorithm 2).

C. Centroids Update Mechanism

AOD-RL obtains a state abstraction based on detected objects. In this subsection, we present a novel algorithm that can filter out irrelevant object classes and add new classes to explore based on the environmental feedback.

We give a measurement of the relevance of each class of object. A relevant object class implies that the presence of objects can influence the state value. For each cluster centroid $c_i \in \mathcal{C}$, we examine the difference between $Q(\phi(s), a)$ and $Q(\phi_{-i}(s), a)$, where $\phi_{-i}(s)$ represents the abstract mapping without detecting class i. Large difference means that class i has a significant influence on $\overline{Q}$. Based on this motivation, we propose Algorithm 2, which describes the (g) step in Fig. 1.

Algorithm 2 Centroids Update Algorithm

Input: New observations in memory $\mathcal{X}_t$, previous centroids $\mathcal{C}_{t-1}$ and $\overline{Q}_{t-1}$.
Output: A new set $\mathcal{C}_t$ containing relevant centroids and exploring centroids.

procedure CENTROIDS UPDATE($\mathcal{X}_t$, $\mathcal{C}_{t-1}$, $\overline{Q}_{t-1}$)
  Perform k-means algorithm on $\mathcal{X}_t$, with initial centroids $\mathcal{C}_{t-1}$, denote the new centroids as $\mathcal{C}$.
  $K \leftarrow |\mathcal{C}|$, $\phi(s) \leftarrow \Phi(s, \mathcal{C})$
  Sample a set of states and the corresponding actions from the replay memory for calculating Score(i).
  for Each $c_i \in \mathcal{C}$ **do** ▷ Calculate relevance of i
    $\mathcal{C}_{-i} \leftarrow \mathcal{C} \setminus \{c_i\} \cup \{\text{random}(\mathbb{R}^p)\}$,
    $\phi_{-i}(s) \leftarrow \Phi(s, \mathcal{C}_{-i})$
    Score(i) $\leftarrow \sum_{s,a} |\overline{Q}_{t-1}(\phi(s), a) - \overline{Q}_{t-1}(\phi_{-i}(s), a)|$
  $\mathcal{C}_{\text{revelant}} \leftarrow \{c_i \in \mathcal{C} \mid \text{Score}(i) \ge \text{threshold}, 1 \le i \le K\}$
  $\mathcal{C}_{\text{explore}} \leftarrow \{\text{random}(\mathbb{R}^p) \mid \text{Score}(i) < \text{threshold}, 1 \le i \le K\}$
  $\mathcal{C}_t = \mathcal{C}_{\text{relevant}} \cup \mathcal{C}_{\text{explore}}$
  return $\mathcal{C}_t$

Algorithm 2 retains centroids with relevance higher than a threshold, and replaces others with random centroids for exploration. The threshold is determined by a rule of thumb based on t-test for Q values.

We compare each class with all the classes that are smaller than its relevance and test if there is a significant difference. Let Score(i) be sorted in a descending order, and let $x_i =$

Score(i) and $\bar{x}_{i+} = 1/(K-i)\sum_{j=i+1}^{K}$ Score(j), where $\bar{x}_{i+}$ represents the average of the smallest $K-i$ scores. For each i, we perform a t-test on $\mathcal{H}_0 : x_i = \bar{x}_{i+}$, and denote the t-value as t_i. The test statistic t_i should be close to 0 for irrelevant classes and significant positive for relevant ones. Then the threshold is chosen to be Score$(\arg\max_i t_i/t_{i+1})$. See Fig. 2 (c) for an intuitive understanding.

Algorithm 2 can help remove irrelevant classes and it enables exploration of new classes in the next iteration. Algorithm 2 is inserted in line 11 of Algorithm 1, and makes automatic object detection feasible.

IV. Experiments

A. Environment and experimental settings

We use a modified *BattleCity* video game environment [8], where the agent controls a tank to destroy enemy tanks by moving and firing. This game is simple and contains obvious object features, thus we use it to evaluate the performance of AOD-RL. The ground state space $\mathcal{S}$ consists of RGB images of size $(160, 160, 3)$ (600 Kb per frame). The action space $\mathcal{A}$ is of size 9, including movements in four directions, fire in four directions and also standby. The reward $R(s_t, a_t)$ is 1 if an enemy is destroyed; otherwise, it is set to be -0.01.

AOD-RL uses the grid abstraction $\phi(s)$ described in III-A with $L = 10$, K in algorithm 1 is set to be 20. As a result, each abstracted state uses $2,000$ bytes. Compared to the original image, the abstracted state $\bar{s}$ is only 0.3% the size of $s \in \mathcal{S}$. We use the same convolutional neural networks (CNN) as the Q-function approximator for all cases, with 4 layers of 64 filters and 1 dense layer of 128 filters. In AOD-RL, the length of bounding boxes randomly ranges in $[20, 40]$ and then resize to $20\times20\times3$. The autoencoder encodes 300 boxes to $7\times7\times3$ by a 3-layer CNN. The k-means are executed in $\mathbb{R}^{147}$. γ is set to be 0.9 and learning rate is 10^{-4} along with the Adam optimizer. The number of trajectories T in Algorithm 1 is $2,000$, and the average number of steps is about 500.

We provide a video demo of the AI player at the link[1].

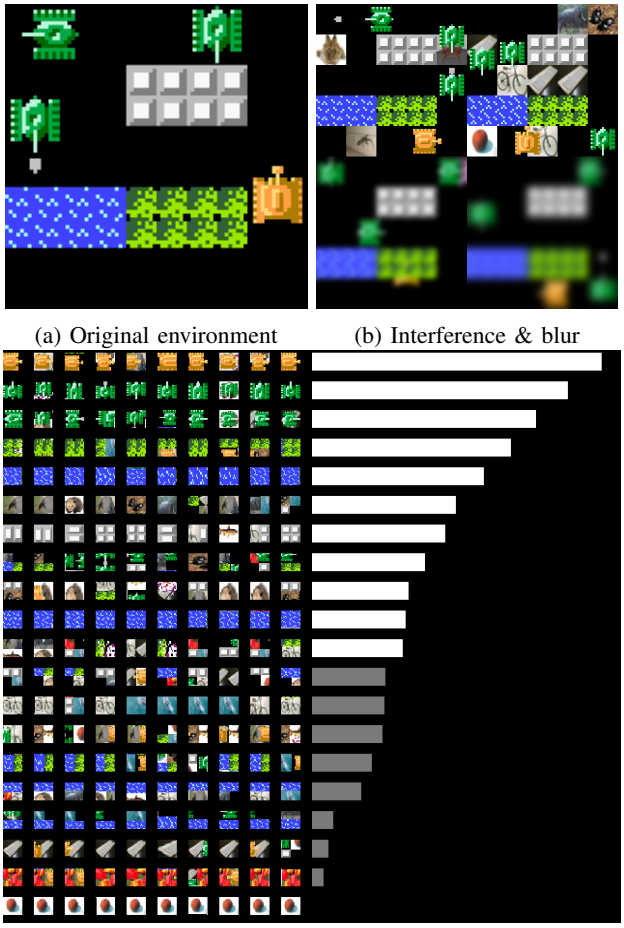
B. Evaluations

We consider the following aspects of the proposed methods.

Validation. We compare it with the original image-input version Q learning, which is denoted as *IMAGE*. We also compare the AOD-RL with the image autoencoder method [9] and denote the latter as *IMAGE-AE*. To validate the correctness of AOD-RL, we evaluate the distance between $\overline{Q}^*(\phi(s), a)$ and $Q^*(s, a)$. We treat the optimal model (*OPT*) as the one based on ground truth coordinates information from game memory. We examine the performance of AOD-RL, IMAGE and IMAGE-AE in the environment and record the average reward divided by the value of OPT. Denoting the original environment as Origin, we also introduce the following challenges to examine the robustness.

Robustness. In complex games, many factors may interfere with RL agents, for example, visual effect and environmental objects. We add foreign objects on time-varying positions of the map, and the levels of interference are named as Interference-n, where n represents the number of interfering objects. We also add Gaussian blur to the environment to test robustness, and different degree of blur is named Blur-n, where n represents the kernel size of Gaussian blur.

[1]Watch AOD-RL playing *Battle City* at https://youtu.be/yTJImqxO7Z0.



(a) Original environment (b) Interference & blur

(c) Detected objects and relevance score

Fig. 2: (a) the original environment; (b) interference and blur, (c) clusters and their relevance scores. Gray bars indicate irrelevant objects. AOD-RL filters out 9 classes out of 20.

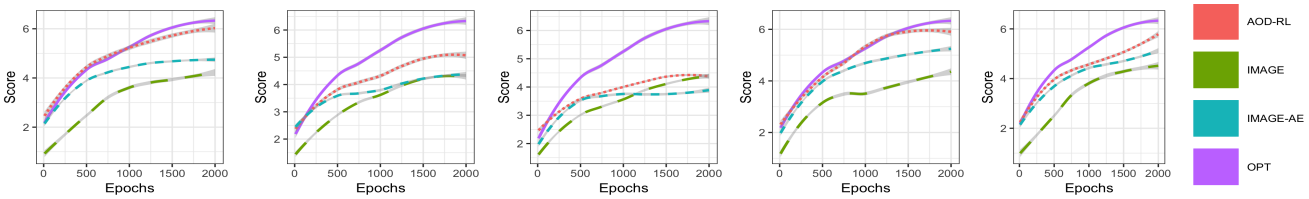
C. Results

We conduct experiments on 8 different game maps and report the results. Each map corresponds to a different MDP, and we artificially generate four associated environments with disturbances, namely, Interference-3, Interference-10 (e.g., see two of the top in Fig. 2 (b)), Blur-2 and Blur-3 (e.g., see two of the bottom in Fig. 2 (b)).

The k-means in Algorithm 1 is executed when $5,000$ new (s, a) are collected, and Algorithm 2 is applied in the first 10 iterations. In Fig. 2 (c), we exhibit some objects that the agent recognizes. On the right side of each class, we use a bar plot to show their relevance score, and objects with gray bar are considered to be irrelevant according to the t-test rule.

TABLE I: Average final reward for 4 candidate methods, divided by the reward of OPT.

	Origin			Interference-3			Interference-10			Blur-2			Blur-3		
case	IM	AE	AOD	IM	AE	AOD	IM	AE	AOD	IM	AE	AOD	IM	AE	AOD
1	.68	.76	**.94**	.68	.70	**.78**	**.69**	.61	.67	.70	.82	**.96**	.73	.78	**.87**
2	.74	.78	**.97**	.71	.67	**.84**	.65	.62	**.74**	.64	.76	**.90**	.69	.74	**.86**
3	.68	.65	**.87**	.62	.63	**.76**	.62	.53	**.72**	.66	.58	**.69**	.62	.70	**.85**
4	.70	.64	**.88**	**.73**	.66	**.73**	.67	.66	**.77**	**.70**	.66	.67	.64	.68	**.71**
5	.69	.87	**.88**	.69	.68	**.77**	.50	.43	**.81**	**.85**	.79	.81	**.86**	.60	.83
6	.90	.87	**.96**	.72	.68	**.90**	.73	.72	**.86**	.88	.80	**.99**	.89	.90	**.93**
7	.77	.64	**.88**	.71	.58	**.83**	**.69**	.55	.66	.79	.64	**.85**	**.85**	.62	.83
8	.82	.74	**.96**	.78	.71	**.91**	.82	.71	**.85**	.82	.69	**.96**	**.95**	.72	.94



(a) Origin (b) Interference-3 (c) Interference-10 (d) Blur-2 (e) Blur-3

Fig. 3: Smoothed training curves in case 1 of the 4 models.

Table I shows the smoothed reward curves of each experiment. Fig. 3 shows the accumulated reward as a function of training epochs in case 1.

We draw the following empirical conclusions. In the Origin environment, AOD-RL behaves as well as OPT from the perspective of training speed and final score (see Fig. 3 (a)). When there is interference, the performance of all three methods has declined. Under moderate disturbance conditions, AOD-RL is significantly better than the other two (see Fig. 3 (b)). In blurred environments, AOD-RL is close to OPT in Fig. 3 (d) and the performance becomes worse in Fig. 3 (e) but it is still better than the other two.

From Table I, AOD-RL performs the best in most cases, and the average final reward of AOD-RL reaches 84% of OPT, while IMAGE is 73.2% and IMAGE-AE is 68.9%. These numbers show that AOD-RL can still achieve a good level of performance in the presence of interference and blurring.

V. Conclusions

Human-designed and rule-based state abstractions are widely used in games [10]–[12]. These methods have empirically proven to be useful, but often require the involvement of human experts. Our method is reward-driven and does not require the participation of human experts.

The game AI trained by our method is more robust and has anti-interference characteristics. For better user experience, modern games use numerous decorative items and visual effects, which increases the difficulty of AI players. Our approach provides a robust solution for this circumstance.

In complex games such as StarCraft [13], object coordinates are necessary for AI. This requires additional development costs and may lead to security issues when AI is accessing game memory. Our method provides a safe and lightweight way to teach AI to obtain coordinates automatically.

References

[1] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive Psychology*, vol. 12, no. 1, pp. 97–136, 1980.

[2] M. I. Posner and Y. Cohen, "Components of visual orienting," *Attention and Performance, X, Hillsdale*, 1984.

[3] J. P. Gottlieb, M. Kusunoki, and M. E. Goldberg, "The representation of visual salience in monkey parietal cortex." *Nature*, vol. 391, no. 6666, pp. 481–484, 1998.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[5] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.

[6] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[8] H. Sethy, A. Patel, and V. Padmanabhan, "Real time strategy games: a reinforcement learning approach," *Procedia Computer Science*, vol. 54, pp. 257–264, 2015.

[9] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *The 2012 International Joint Conference on Neural Networks*, 2012.

[10] N. Jiang, A. Kulesza, and S. Singh, "Abstraction selection in model-based reinforcement learning," in *International Conference on Machine Learning*, 2015, pp. 179–188.

[11] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[12] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara, "Transparency and explanation in deep reinforcement learning neural networks," 2018.

[13] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser *et al.*, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.