I4R: Promoting Deep Reinforcement Learning by the Indicator for Expressive Representations

Xufang Luo[1*] , **Qi Meng**[2] , **Di He**[3] , **Wei Chen**[2] and **Yunhong Wang**[1]
[1]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing
[2]Microsoft Research, Beijing, China
[3]Key Laboratory of Machine Perception (MOE), School of EECS, Peking University
{luoxufang, yhwang}@buaa.edu.cn, {meq, wche}@microsoft.com, di_he@pku.edu.cn

Abstract

Learning expressive representations is always crucial for well-performed policies in deep reinforcement learning (DRL). Different from supervised learning, in DRL, accurate targets are not always available, and some inputs with different actions only have tiny differences, which stimulates the demand for learning expressive representations. In this paper, firstly, we empirically compare the representations of DRL models with different performances. We observe that the representations of a better state extractor (SE) are more scattered than a worse one when they are visualized. Thus, we investigate the singular values of representation matrix, and find that, better SEs always correspond to smaller differences among these singular values. Next, based on such observations, we define an indicator of the representations for DRL model, which is the *Number of Significant Singular Values* (*NSSV*) of a representation matrix. Then, we propose I4R algorithm, to improve DRL algorithms by adding the corresponding regularization term to enhance the NSSV. Finally, we apply I4R to both policy gradient and value based algorithms on Atari games, and the results show the superiority of our proposed method.[1]

1 Introduction

Along with the developmet of deep learning techniques, deep reinforcement learning (DRL) models have been more widely used in decision making tasks and automatic control tasks [Mnih *et al.*, 2015; Silver *et al.*, 2016; Schulman *et al.*, 2017]. A DRL model consists of two parts. One is a deep neural network (DNN) which is for learning representations of the state, via extracting features from raw inputs (i.e., raw signals). Here, such DNN can be considered as a state extractor (SE). For example, convolutional neural network (CNN) and recurrent neural network (RNN), are two kinds of typical SEs for images-based games [Mnih *et al.*, 2015] and natural language-based games [Zhao and Eskenazi, 2016], respectively. The other is to take actions according to the representations and the current learned policies, and the whole DRL model can be optimized by the gradient/value based DRL algorithms.

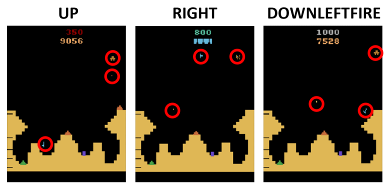


Figure 1: Raw frames with different actions above in the Atari game Gravitar. Fine-grained local differences are labeled in red circles.

In this paper, we aim to improve the SE in DRL models. As we all know, well-performed SEs can extract high-quality representations, which retains the essential information and is crucial for good policies. However, training SEs end-to-end in DRL models, always faces the challenge that, direct and accurate targets are not available. Instead, although Bellman Error or estimated return can be used for updating signals, they cannot serve as the efficient guidance for SEs to extract expressive representations.

Furthermore, different from the case in supervised learning where the inputs are largely diverse (e.g. ImageNet classification), there is another big challenge that, in DRL, SEs need to extract expressive and fine-grained local features. Specifically, when the consecutive raw inputs are similar, the tiny differences among inputs may probably lead to different actions. For instance, in Atari game Gravitar, as shown in Fig. 1, the input frames are extremely similar to others which contain the same objects, and they only differ in few local details which are essential to take the quite different actions (e.g., Up, Right, and Downleftfire).

Therefore, we need to design a method for training better SEs, to extract expressive representations for better policies in DRL models. Referring to the indicators [Tran *et al.*, 2017; Hjelm *et al.*, 2019], widely used in unsupervised learning tasks which are also lack of direct and accurate targets for learning representations, in this paper, we quantitatively define a novel indicator for expressive representations based on the experimental observations, and utilize it to improve DRL algorithms. ***First***, we investigate the relationship between the representations and performance in DRL via visualizing the representations generated by SEs under different performances. We observe that for a better SE, the representations are more

[*]This work was conducted at Microsoft Research.
[1]Released code: https://github.com/XufangLuo/I4R

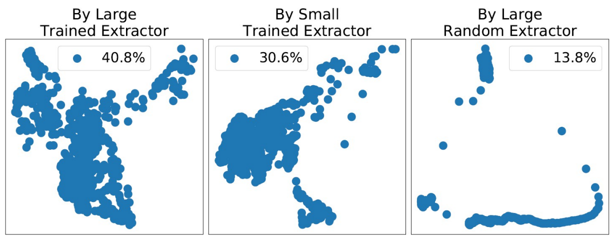
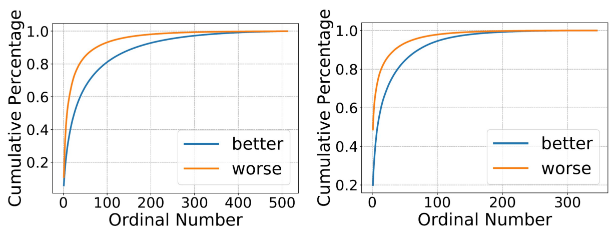


Figure 2: Two-dimensional embedding of the representations assigned by different SEs. The percentage of the covered area is labeled.



(a) High rewards trajectory. (b) Low rewards trajectory

Figure 3: Cumulative percentage of singular values.

scattered, and the singular values of the representation matrix will have smaller differences. Here, the representation matrix is composed of stacked features obtained by the SE applied to each sample in the batch. ***Next***, based on the above observation, we define an indicator using the *Number of Significant Singular Values* (*NSSV*) in the representation matrix for representations. Here, NSSV can indicate the divergence of singular values, i.e., the smaller differences the singular values have, the larger NSSV of SE will be. Thus, we conduct exploratory experiments on Atari games, to verify the relationship between NSSV and the performance of DRL algorithms, and the result demonstrates the positive correlation between NSSV and the reward. ***Then***, we propose a novel method, called I4R (i.e., **I**ndicator **for R**epresentations), to promote the performances for DRL algorithms by adding NSSV as a regularization term to the loss function. In details, because it is difficult for us to optimize an argmin operator in NSSV, here we design two regularization terms, i.e., *Max-Min* and *Max/Min*, as the surrogates of NSSV. Both two regularization terms are computationally efficient, and can be applied to various DRL algorithms. ***Finally***, we apply our proposed method on two popular DRL algorithms, A3C [Mnih *et al.*, 2016] and DQN [Mnih *et al.*, 2015], for playing Atari games, and results show that I4R outperforms the baselines on most games.

2 Related Work

In recent years, there are many studies on learning representations. On the one hand, just in DRL area, generally speaking, there are two main typical categories for learning representations. One is based on auxiliary models, such as auto-encoders [Mattner *et al.*, 2012; Higgins *et al.*, 2017], generative adversarial networks [Donahue *et al.*, 2017; Shelhamer *et al.*, 2016] and some other models [Oh *et al.*, 2017; Racanière *et al.*, 2017]. Such auxiliary models can help to improve learning representations via completing some certain tasks, such as reconstructing the current observation or state [Watter *et al.*, 2015], predicting the future observations or states [François-Lavet *et al.*, 2019], and recovering the actions given transitions [Zhang *et al.*, 2018]. Here, some essential information for taking actions can be retained by completing the above auxiliary tasks. The other is based on prior task-specific knowledge/information. For example, in [Goel *et al.*, 2018], the authors proposed to use the detection of moving objects for learning video games better. In [Jonschkowski and Brock, 2015], the authors proposed to use robotic prior knowledge for robot learning. In [Zhao and Eskenazi, 2016], task-related information was utilized in chatting systems. Different from these methods, in this paper, we simply add a regularization term without any task-specific knowledge/information, and do not need to build or train any extra models.

On the other hand, in unsupervised learning area which has the similar challenge to RL, that both of them are lack of direct and accurate targets for model-training, using principles to help to learn representations is one of the most widely used methods. In [Tran *et al.*, 2017; Hjelm *et al.*, 2019], the authors designed disentanglement or Infomax, for learning representations in some stochastic models (e.g., GAN and VAE). And there are also some principles which are proposed for other purposes, such as denoising [Vincent *et al.*, 2010] for robustness. All these above methods are different from our proposed method which is motivated by the observations in DRL, and applied into the deterministic RL models.

3 Methodology

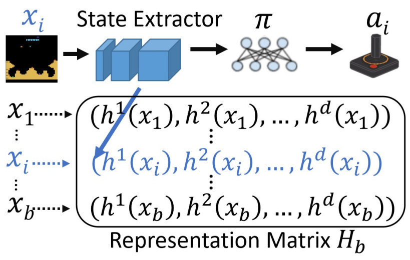
In this section, we show the proposed method I4R based on exploratory experiments, including 3 parts, i.e., observations, the proposed indicator NSSV, and the novel algorithm I4R.

3.1 Observations on Exploratory Experiments

In this section, all exploratory experiments are implemented on a popular decision making task, Atari game, with high-dimensional video frame inputs, and we aim to find out the relationship between the representations extracted by SE and the performance of DRL algorithms.

We select two DRL models with the same number of last hidden layer units but different model sizes, to play the Atari game Gravitar for 200M frames, respectively. After the training process, the large model scores 3050, while the small model scored 600. Besides, for excluding the bias of model size, we also illustrate another model with the representations which are generated by the former large model with its randomly initialized weights, and this model only scores 0. Thus, we name these above three models as *large trained* model, *small trained* model and *large random* model, respectively.

First, we use the raw frames in the trajectory played by the large trained model, as the input of three SEs, and plot the two-dimensional embedding of the representation matrices in the last hidden layer assigned by such three models in Fig. 2. These plots are generated by using SVD dimension reduction on the representation matrices. From Fig. 2, we observe that the embedding of representations generated by the model with



x_i
State Extractor
π
a_i
x_1
$(h^1(x_1), h^2(x_1), \ldots, h^d(x_1))$
x_i
$(h^1(x_i), h^2(x_i), \ldots, h^d(x_i))$
x_b
$(h^1(x_b), h^2(x_b), \ldots, h^d(x_b))$
Representation Matrix H_b

Figure 4: Architecture of a DRL model and its representation matrix H_b. d is the number of hidden units, and b is the mini-batch size.

higher final score, is more scattered, which indicates that this high-dimensional RL task may prefer to more scattered representations.

Next, we give a more informative description about the visualization above. we plot the cumulative percentage of singular values, i.e., $\frac{\sum \text{first k largest singular values}}{\sum \text{all singular values}} \times 100\%$ (k is an ordinal number) in Fig. 3(a). Here, high rewards trajectory generated by the large trained models are sent to large trained and small trained models to generate representation matrices. We can conclude that, compared with a worse model, the curve of a better model is closer to a linear one. Besides, we also observe the same results if we change the input trajectories to those generated by the small trained model in Fig. 3(b). And the same behavior is also exhibited in other games, such as SpaceInvader[2]. Regarding relatively small singular values as noise, the curve regarding the cumulative percentage of singular values, can reflect the intrinsic dimension of the representation [Houle, 2017], and if the curve is closer to a linear one, the intrinsic dimension of the representation will be higher. The above results indicate that, regardless of the input trajectories, the intrinsic dimension of the representation matrix, will be higher when they are generated by a better DRL model, which **motivates** us that, by defining an indicator to evaluate the intrinsic dimension in the representation matrix, we can evaluate the corresponding DRL model.

3.2 NSSV: An Indicator of Representations

According to the observations on exploratory experiments, we use NSSV, i.e., the smallest number of significant singular values in the representation matrix, which is then verified to be highly related to the performance of DRL models, especially in high-dimensional RL tasks.

Definition of NSSV

Assume that $\{x_1, \cdots, x_b\}$ is a mini-batch of raw inputs. We use operator $h(x) : \mathcal{X} \to \mathcal{S}$ to denote an SE, where $\mathcal{X}$ and $\mathcal{S}$ denote the raw input space and the state space, respectively. For given observation x_i, we obtain its the representation $(s_{i1}, \cdots, s_{id}) = h(x_i) = (h^1(x_i), \cdots, h^d(x_i))$. Here, as shown in Fig. 4, the representations for a mini-batch of observations, consist of a matrix which can be called *representation matrix* and denoted as $H_b = (h(x_1), \cdots, h(x_b))^T$.

Therefore, we have the definition of NSSV as follows.

[2]https://github.com/XufangLuo/I4R/releases/download/0.1/SupplementaryMaterials.pdf

Episode Data	ϵ	Average NSSV over 39 games		
		LT Model	ST Model	LR Model
LT Model	20%	164.9	137.8	103.8
	10%	247.7	213.8	197.4
ST Model	20%	135.4	116.3	83.7
	10%	210.8	187.6	166.1
LR Model	20%	72.1	64.7	50.9
	10%	120.6	112.0	105.5

Table 1: Relationship between model performance and NSSV. LT = Large Trained; ST = Small Trained; LR = Large Random.

Definition 1. *(NSSV) Suppose h is an SE in DRL, and $H_b \in \mathbb{R}^{b\times d}$ is its representation matrix over observations $\{x_1, \cdots, x_b\}$. We define the NSSV of SE h in the following equation:*

$$NSSV(H_b; \epsilon) = \mathbf{argmin}_j \left\{ j : \frac{\sum_{i=1}^{j} \sigma_i(H_b)}{\sum_{i=1}^{\min\{b,d\}} \sigma_i(H_b)} > 1 - \epsilon \right\}, \tag{1}$$

where $\sigma_i(H_b)$ is the i-th largest singular value of H_b.

NSSV is the least number of singular values whose sum's percentage passes a threshold over to all singular values. Notably, NSSV is related to the number of the sampled observations, given precision ϵ, and extractor model h.

Relationship between NSSV and Rewards

In order to verify the correlation between NSSV and the performance (i.e., the reward) of DRL models, we compare NSSV of different performed SEs, and observe the change of NSSV during the training process.

In details, for each 55 Atari game, we compare the NSSV of a large trained model, a small trained model, and a large random model, respectively, and select the Atari games in which the large trained model performs best, and the large random model performs worst. In this way, we finally pick 39 games among 55 ones. Then, for each 39 game, we use three models to generate three trajectories with different final rewards, and all the observations in these three trajectories, are sent to three SE models, respectively. Hence, we can obtain 9 matrices composed of the generated representations. As a result, the average NSSV over 39 games with ϵ of 10% and 20%, are listed in Table 1. The results demonstrate that, given different trajectories and model sizes, the average NSSV is always positively relative to the model performance, i.e., the NSSV of a better model is always larger than a worse one.

Furthermore, we also study the change of NSSV during the training process. Here, we train a model to play the Atari game WizardOfWor for 200M frames. Each time the model is updated in a mini-batch of transitions, the representation vectors in the last hidden layer will compose a representation matrix, and singular values of the representation matrix will be recorded. Then, we calculate NSSV with ϵ of 20%. Fig. 5 shows that the curve trends of the testing rewards and NSSV, are highly consistent, i.e., when NSSV increases/decreases, the test rewards increase/decrease, concomitantly. And the same phenomena on more games, can been seen in Fig. 7 which is the following section.

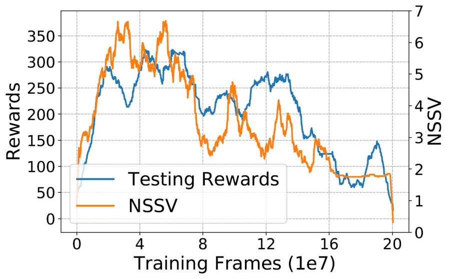


Figure 5: Curves tracking testing rewards and NSSV during the training process.

Notably, we notice that during the training process, NSSV, together with the testing rewards, is undulant, which indicates that, the representations are not expressive enough during the training process and cause troubles for decision making. Therefore, we conclude that the DRL model should be further improved during the training process, and with the help of NSSV, we will next propose I4R algorithm.

3.3 I4R: An Algorithm for Promoting DRL

In this subsection, we propose our method, so-called I4R, to improve the power of SE, aiming at promoting the performance of DRL models.

As aforementioned, an SE is encouraged to have a larger NSSV. In order to minimize the computation cost as much as possible, here we adopt a simple way that, a regularization term $R(H)$ of representation matrix H, is added to the loss function. Thus, the total loss function is denoted as follows:

$$\mathcal{L} = Loss + \alpha * R(H), \tag{2}$$

where α is the coefficient.

On the one hand, for policy gradient algorithm, the loss function is a policy loss which is defined as $\mathcal{L}_{policy}(\theta) = \sum f(s,a)\log \pi(a|s,\theta)$, where $\pi(a|s,\theta)$ is the output of the policy network with parameter θ. Here, $f(s,a)$ is a criterion function, e.g., the advantage in Actor-Critic algorithm [Munk *et al.*, 2016]. On the other hand, for value based algorithms, the loss function is a squared temporal difference which is defined as $\mathcal{L}_{value}(\theta^t) = \mathbb{E}_{s,a,s'}[(r+\gamma \max_{a'} Q(s',a';\theta^{t-1}) - Q(s,a;\theta^t))^2]$, where $Q(s,a;\theta)$ is the learned Q-value of a DNN [Van Hasselt *et al.*, 2016].

Because $R(H)$ should be equal to $-NSSV(H_b;\epsilon)$, and optimizing $-NSSV(H_b;\epsilon)$ directly is not easy under an argmin operator, the next proposition will show that constraining the difference between the largest singular value and the lowest singular value, can make $-NSSV(H_b;\epsilon)$ to be larger, which can take the place of the argmin operator.

Proposition 1. *Assume that the singular values in the representation matrix of SE h_1, satisfy $\sigma_1 > \sigma_2 > \cdots > \sigma_d$ and the singular values in representation matrix of SE h_2, satisfy $\tilde{\sigma}_1 > \tilde{\sigma}_2 > \cdots > \tilde{\sigma}_d$, and $\tilde{\sigma}_1 < \sigma_1, \tilde{\sigma}_d > \sigma_d$, $\sigma_i = \tilde{\sigma}_i$ for $i = 2, \cdots, d-1$, then we have $NSSV(H_b^1;\epsilon) < NSSV(H_b^2;\epsilon)$.*

According to Proposition 1, if the gap between the largest and the smallest singular value becomes more narrow, NSSV will be correspondingly improved. Therefore, we propose two surrogates of $-NSSV(H_b;\epsilon)$ as follows.

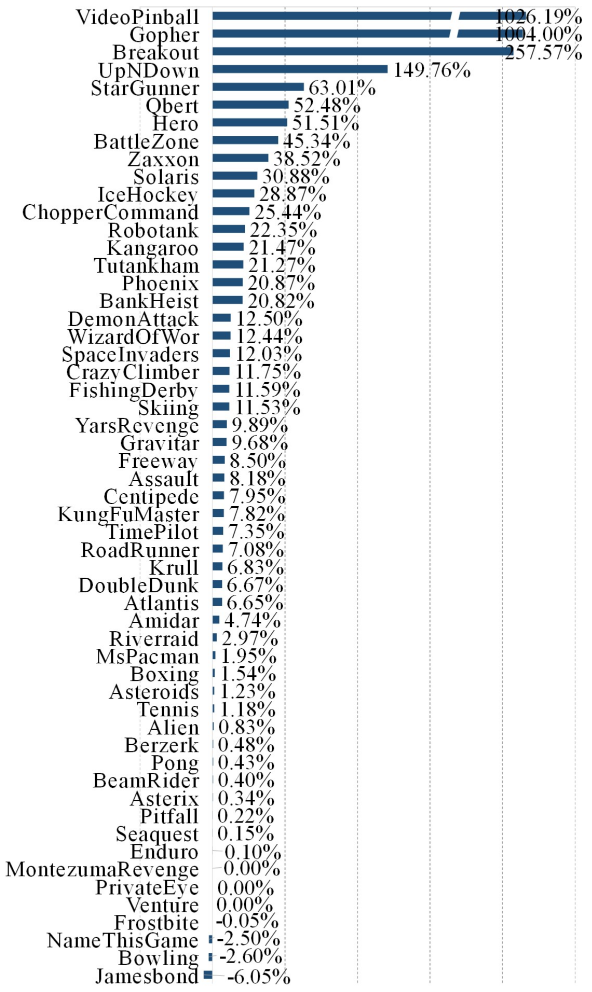


Figure 6: Improvements of our method compared to A3C, using the metric given in Eq. 5.

The first one, **Max-Min**, is defined as follows:

$$Max - Min = \sigma_{max}(H_h) - \sigma_{min}(H_h), \tag{3}$$

where $\sigma_{max}(H_h)$ and $\sigma_{min}(H_h)$ are the largest singular value and the smallest singular value of H_h, respectively.

The second one, **Max/Min**, is defined as follows:

$$Max/Min = \frac{\sigma_{max}(H_h)}{\sigma_{min}(H_h)}. \tag{4}$$

Similar to the Max-Min term, Max/Min should be minimized during the training process.

Notably, that I4R can be applied to many kinds of DRL algorithms, and it can improve DRL algorithms just by adding a regularization term when the DRL model is updated. Apart from the transitions used for updating the DRL model, there are no extra data to be introduced for encouraging highly expressive representations. Thus, the proposed method can be widely applied to many current DRL algorithms with little extra computational cost, and in the following section, we will next further test the performance of I4R by experiments.

Hyper-parameter	Mean	Median
A3C	595.19%	75.75%
Max-Min α=0.01	765.75%	84.27%
Tune the Regularizer	772.40%	96.62%
Tune the Regularizer and α	788.16%	96.62%

Table 2: Results on Atari games with different hyper-parameters.

Term	A3C	+ Max/Mean	+ Max-Mean
Mean	453.70%	622.04%	930.77%
Median	226.75%	231.23%	272.41%

Table 3: Comparison between two regularization terms via human normalized scores in Eq. 6.

4 Experiments

In this section, we conduct experiments to test the effectiveness of our proposed method for both gradient and value based DRL algorithms, and we evaluate our proposed method on 55 Atari games [Bellemare *et al.*, 2013]. Atari games are challenging and one of the most popular RL tasks. Although all Atari games are video frames, their intrinsic mechanism varies tremendously, which makes it difficult to find a single algorithm and a hyper-parameter setting for all games.

Thus, we will next mainly answer the following questions: (1) Can the performances of DRL model be improved by applying I4R method to the baseline DRL algorithm (e.g., A3C)? (2) How to choose the two proposed regularization terms and how to set coefficient α? (3) Will the NSSV indicator of SE be promoted after using I4R? (4) Can I4R improve the performance of other kinds of DRL algorithms? These above questions will be answered in the following subsections.

4.1 Experiments for Promoting A3C

Basic Settings

In this subsection, we use A3C [Mnih *et al.*, 2016], one of the widely used policy based DRL algorithms, as our baseline. Here, each agent is trained for 200M game frames, and the reward is averaged over 5 runs for different random seeds. As for I4R, the outputs of LSTM are taken as the representations generated by SE, and each time the A3C model is updated using a mini-batch of transitions, the representations of observations in these transitions will form matrix H. Besides, except for coefficient α, other hyper-parameters and settings are set as the same as the baseline, and we will discuss the selection of the regularization terms and α in the next subsection.

Overall Performances

Referring to [Wang *et al.*, 2016], we use the measurement as shown in Eq. 5, to compare the performance between I4R and the baseline method. Here, human scores and random scores are also taken from [Wang *et al.*, 2016].

$$\frac{\mathrm{Score_{Agent}} - \mathrm{Score_{Baseline}}}{\max\{\mathrm{Score_{Human}}, \mathrm{Score_{Baseline}}\} - \mathrm{Score_{Random}}} \times 100\%. \quad (5)$$

Then, we summarize the improvement of our method over the baseline in Fig. 6. Among 55 games, we can observe that, our proposed I4R outperforms A3C over 48 games, which demonstrates that utilizing I4R, can obtain more expressive representations and improve the original DRL algorithm.

NSSV Indicator Analysis

In this subsection, we analyze the NSSV indicator by comparing the baseline and our proposed I4R. Here, the curves tracking testing rewards and the NSSV are plotted in Fig. 7. We can see that, I4R outperforms A3C on these games, which shows that improving the NSSV indicator can significantly benefit policy learning for DRL. Besides, we can observe that, the trends of the reward curve and the NSSV curve, are highly consistent, and a larger reward and a larger NSSV always come out together, which implies that it is necessary to hold expressive representations for learning good policies.

4.2 Hyper-parameters Analysis

Choosing Hyper-parameters

To choose proper hyper-parameters, we test Max-Min and Max/Min with α in $\{0.1, 0.01, 0.001\}$, respectively, on three randomly selected games. We find that, the Max-Min term with $\alpha = 0.01$ performs the best, and outperforms the baseline on all three games. Thus, we apply such setting to all other games, and in Table 2, we list the human normalized score calculated by Eq. 6. The result shows that with the same hyper-parameter settings, our proposed I4R performs much better than the baseline, which demonstrates the superiority of I4R.

$$\frac{\mathrm{Score_{Agent}} - \mathrm{Score_{Random}}}{\mathrm{Score_{Human}} - \mathrm{Score_{Random}}} \times 100\%. \quad (6)$$

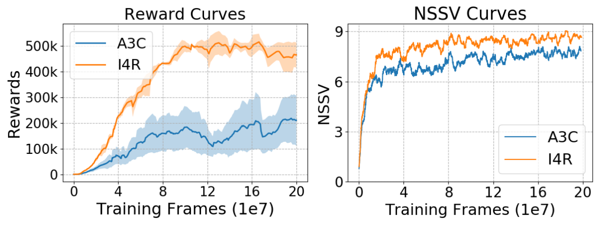
And then, we change the regularization term from Max-Min into Max/Min on some games which did not perform well, and also change α, accordingly. Finally, we use Max-Min with $\alpha = 0.01$ on 39 games, and $\alpha \neq 0.01$ only on three games, and the chosen regularization term and α for all 55 games in final results in Fig. 6 are listed in Table 2 of Supplementary Materials. In Table 2, we also list the score results, and we can observe that improvements can be obtained further by slightly tuning the regularization term and α.

Two Regularization Terms

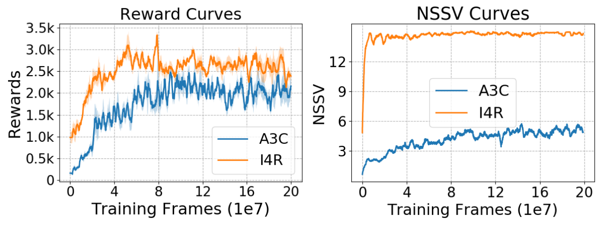
To find the difference between the two regularization terms, we test each term on 9 randomly selected Atari games with $\alpha = 0.01$. Then, we list mean and median scores among these 10 games in Table 3, and plot the performances of the game UpNDown and Qbert in Fig. 8. Here, we can see that, generally, the algorithm performance can be promoted largely by respectively applying both two regularization terms, and larger improvements can be obtained by adding the Max-Min term rather than the Max/Min term. We consider that such phenomenon is reasonable, because the Max/Min term will become very large when the minimum singular value is small, while from such perspective, the Max-Min term can be relatively soft for scale.

Ablation Study on α

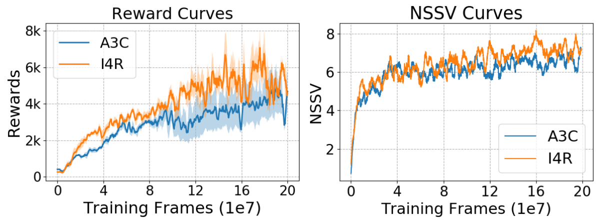
We conduct an ablation study on α to investigate the effectiveness of I4R. Here, we select 9 games and fix their rank regularization term types. Then, we run I4R with different α from 10^{-1} to 10^{-6}, and the performances are evaluated using the metric in Eq. 5, In Table 4, we show that our proposed I4R outperforms the baseline on average scores over games with most α. Besides, the rank regularization term plays a major role on improvements, instead, α is just used for tuning.



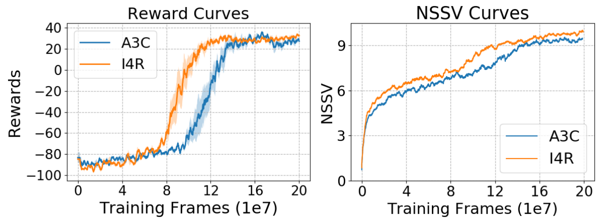
(a) UpNDown.



(b) Asteroids.



(c) Asterix.



(d) FishingDerby.

Figure 7: Testing rewards curves (left) and NSSV curves (right) on 4 Atari games for I4R (yellow) and the baseline (blue).

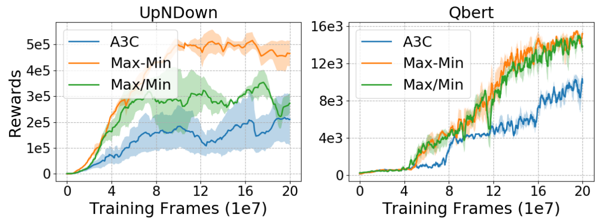


Figure 8: Rewards of I4R with two regularization terms. The shaded area depicts variances.

4.3 Experiments for Promoting DQN

Since that the proposed I4R does not contain any algorithm-related operations, we investigate whether it can promote the performance of other DRL algorithms, e.g., value based DRL algorithms. Here, we choose DQN [Mnih *et al.*, 2015] as the baseline, and use the same network architecture and hyper-parameters as [Mnih *et al.*, 2015]. We run each method on 30 Atari games (i.e., the first 30 ones in the alphabetical order) for 200M frames. As a result, we find that our proposed method outperforms the baseline on 20 games, and compared with the baseline, it also obtains more than 2 times human normalized score for the median of scores on 30 games (i.e., 15% and 38%), which demonstrates the superiority of I4R over DQN, and supports our claim that encouraging highly expressive SE promotes performances of kinds of DRL algorithms.

4.4 Simulation Environment and Time Costs

We use Gym package [Brockman *et al.*, 2016] as the Atari simulation environment, and use the pytorch-a3c package [Kostrikov, 2018] which is adopted in [Peysakhovich and Lerer, 2018; Lerer and Peysakhovich, 2018] to implement all algorithms and methods. We do not change the network architecture (shown in Supplementary Materials) embedded in the code. An environment wrapper is utilized to simplify original visual screens which are pre-processed to 42×42 gray-scale images. Here, we use a frame-skip of 4, and the number of process is set to be 16. Besides, all experiments are conducted at Azure Virtual Machines with 24 cores, and it takes the baseline and our proposed method about 8 hours to finish the training process.

α	0.1	0.01	0.001	0.0001	0.00001	0.000001
Average	41.36%	76.91%	56.58%	53.38%	25.16%	35.76%

Table 4: Ablation Study on α.

5 Conclusions

In this paper, we mainly study the relationship between representations and performance of the DRL agents. First, we observe that, when DRL agents achieves high rewards, its representations are more expressive, i.e., the representations are more scattered and the singular values of the representation matrix are more uniformly distributed. Next, we define the NSSV indicator, i.e, the smallest number of significant singular values, as a measurement for learning representations, Then, we verify the positive correlation between NSSV and the rewards, and further propose a novel method called I4R, to improve DRL algthims via adding the corresponding regularization term to enhance NSSV. Finally, we conduct experiments for promoting A3C and DQN, on 55 and 30 Atari games, respectively, which demonstrates that I4R can promote the DRL performance significantly. Besides, in our future work, we will further take the study on if I4R also has influence on supervised/unsupervised learning process.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61421003).

References

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

[Donahue *et al.*, 2017] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017.

[François-Lavet *et al.*, 2019] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *AAAI*, 2019.

[Goel *et al.*, 2018] Vikash Goel, Jameson Weng, and Pascal Poupart. Unsupervised video object segmentation for deep reinforcement learning. In *NeurIPS*, pages 5683–5694, 2018.

[Higgins *et al.*, 2017] Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv*, 2017.

[Hjelm *et al.*, 2019] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[Houle, 2017] Michael E Houle. Local intrinsic dimensionality i: An extreme-value-theoretic foundation for similarity applications. In *SISAP*, pages 64–79, 2017.

[Jonschkowski and Brock, 2015] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[Kostrikov, 2018] Ilya Kostrikov. PyTorch implementations of A3C. https://github.com/ikostrikov/pytorch-a3c, 2018. Accessed on Feb 28, 2018.

[Lerer and Peysakhovich, 2018] Adam Lerer and Alexander Peysakhovich. Learning social conventions in markov games. *arXiv*, 2018.

[Mattner *et al.*, 2012] Jan Mattner, Sascha Lange, and Martin Riedmiller. Learn to swing up and balance a real pole based on raw visual input data. In *ICONIP*, pages 126–133, 2012.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937, 2016.

[Munk *et al.*, 2016] Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. In *CDC*, pages 4667–4673, 2016.

[Oh *et al.*, 2017] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *NeurIPS*, pages 6118–6128, 2017.

[Peysakhovich and Lerer, 2018] Alexander Peysakhovich and Adam Lerer. Consequentialist conditional cooperation in social dilemmas with imperfect information. In *ICLR*, 2018.

[Racanière *et al.*, 2017] Sébastien Racanière, Théophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *NeurIPS*, pages 5690–5701, 2017.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

[Shelhamer *et al.*, 2016] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv*, 2016.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[Tran *et al.*, 2017] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning GAN for pose-invariant face recognition. In *CVPR*, pages 1415–1424, 2017.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11(Dec.):3371–3408, 2010.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, pages 1995–2003, 2016.

[Watter *et al.*, 2015] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NeurIPS*, pages 2746–2754, 2015.

[Zhang *et al.*, 2018] Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *arXiv*, 2018.

[Zhao and Eskenazi, 2016] Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv*, 2016.