SAC+GAARA: Generate Augmented And Reconstructable latent states with Ae

Jizheng Chen, Junru Gong
Shanghai Jiao Tong University
humihuadechengzhi@sjtu.edu.cn
gongjunru@sjtu.edu.cn

Abstract

Nowadays the camera is a convenient and inexpensive way to acquire information, especially in complex, non-stationary and unstructured environments, which made image-based reinforcement learning(RL) receive widespread attention. But training an agent to solve control tasks directly from high-dimensional images has proven difficult, especially in model-free case. Prior work has shown that learning a latent representation together with control policy and applying model-free RL algorithms such as SAC and DDPG on the latent representation helps the learning process. However, the gap between state learning and latent representation learning is still huge and we observe some unstability and inefficiency in encoder training, making the converage slow. In this final project, we combine current popular method SAC+AE with contrastive leaning to train a more stable and robust encoder, and an innovative annealing on contrastive loss is proposed to further improve the encoder. Experiments on MuJoCo control tasks [8] verifies the correctness and efficiency of our method. Code, results and videos are available at https://github.com/Otsuts/SAC-GAARA

1 Introduction

Cameras are a convenient and inexpensive way to acquire state information, especially in complex, unstructured environments, where effective control requires access to the proprioceptive state of the underlying dynamics. Thus, having effective RL approaches that can utilize pixels as input would potentially enable solutions for a wide range of real world applications, for example robotics. However, acquiring a robust and proper representation from raw pixels is a great challenge, and due to the lack of training data, the encoder may be unstable and the training efficiency is limited even though deep convolutional encoders can learn good representations upon which a policy can be trained. What's more, in a model free algorithm(SAC in our case), the agent is fragile to the original pixels, and the encoder may be affected by the changing background, and may not be able to recognise certain gestures of the agent. All of above made reinforcement leaning from images difficult.

Concerning on the challenges, some solutions are proposed to improve sample efficiency and the stability of encoders in order to learning a more reliable representation of latent space, [10] tried the variational autoencoders to acquire a latent state, and explored the underlying reasons and identify variational autoencoders as the cause of the divergence, before proposing SAC+RAE structure to tackle with vulnerability and observasional noise, matching state-of-the-art model-free and model-based algorithms on MuJoCo control tasks.[7] applied the anchor-positive approach, training a visual representation encoder by ensuring the embeddings of data-augmented versions o_q and o_k of observation o match using a contrastive loss.

We revisit the concept of adding an autoencoder to model-free RL approaches focused on off_policy algorithms and process of data augmentation. By runnning them as baselines we confirm that apart from using a decoder to minimize the pixel reconstruction loss, using contrastive learning and data augmentation to further enhance the encoder's capability is vital. Also, in order to dynamically balance contrastive learning and reconstruction, we proposed an annealing way on the data augmentation loss, and ablation study on following parts verifies its accesibility. We name the work GAARA, indicating the process of generating augmented and reconstructable latent states with Ae.(The name GAARA also pays tribute to the beloved charactor Gaara in Cartoon Naruto Shippuden.)

2 Related Work

2.1 Self-Supervised Learning

Self-Supervised Learning is aimed at learning rich representations of high dimensional unlabeled data to be useful for a wide variety of tasks.The fields of natural language processing and computer vision have seen dramatic advances in self-supervised methods such as BERT(Devlin et al., 2018)[2], CPC, MoCo, SimCLR.

2.2 World Models for Sample-efficiency

The key idea of world models are: Learn model of the envirenment from experience and use learned model to improve value/policy optmization.
There are some works focusing on world model these years.Such as PILCO:A model-based and data-efficient approach to policy search.ICML 2011[1] and Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning.RSS 2011.[3]Besides, Dreamer[5], DayDreamer and Iso-Dream[6] model also gain excellent experinmental results.

2.3 Sample-efficient RL for Image-based Control

With the development of DMControl environment, some sample-efficient model for image based continuous tasks could be measured.We mainly focus on methods such as SAC+AE, CURL and so on.And we compared our method with SAC+AE.

3 Background

3.1 AE

An autoencoder is defined by the following components:
Two sets:
the space of decoded messages $\mathcal{X}$, the space of encoded messages $\mathcal{Z}$.
Two parametrized families of functions:
the encoder family $\mathcal{E}_\phi : \mathcal{X} \to \mathcal{Z}$, parametrized by ϕ; the decoder family $\mathcal{D}_\theta$:$\mathcal{Z} \to \mathcal{X}$, parametrized by θ.
For any $x \in \mathcal{X}$, we usually write $z = E_\phi(x)$, and refer to it as the latent variable.Conversely, for any $z \in \mathcal{Z}$, we usually write $x^{'} = D_\theta(z)$, and refer to it as the decoded message.
Usually, both the encoder and the decoder are defined as multilayer preceptrons.For example, a one-layer-MLP encoder $\mathcal{E}_\phi$ is:

$$E_\phi(x) = \sigma(\mathcal{W}x + b) \tag{1}$$

where σ is an element-wise activaion function, W is weight mateix, and b is bias vector.

3.2 Contrastive Learning

Contrastive learning can be understood as learning a differentiable dictionary look-up task. Given a query q and keys $\mathbb{K} = k_0, k_1, ...$ and an explicitly known partition of $\mathbb{K}$(with respect to q) $P(\mathbb{K}) = (k_+, \mathbb{K} \backslash k_+)$, the goal of contrastive learning is to ensure that q matches with k_+ relatively more than any of the keys in $\mathbb{K} \backslash k_+ . q, \mathbb{K}, k_+$,and $\mathbb{K} \backslash k_+$ are also referred to as anthor, targets, positive, negatives respectively in the parlance of contrastive learning.Similarities between the anchor and targets are best modeled with dot products($q^T k$) or bilinear prodects ($q^T W k$) though other forms like euclidean distances are also common. To learn embeddings that respect these similarity relations, van den Oord et al.(2018)[9] propose the InfoNCE loss:

$$\mathcal{L}_q = log\frac{exp(q^TWk_+)}{exp(q^TWk_+) + \sum_{i=0}^{K-1} exp(q^TWk_i)} \tag{2}$$

3.3 SAC

SAC[4] is an off-policy RL algorithm that optimizes a stochastic policy for maximizing the expected trajectory returns. It is effective when solving tasks from state observations but fails to learn efficient policies from pixels. SAC is an actor-critic method that learns a policy π_ψ and critics $\mathcal{Q}_{\phi_1}$ and $\mathcal{Q}_{\phi_2}$. The parameters ϕ_i are learned by minimizing the Bellman error:

$$\mathcal{L}(\phi_i, \mathcal{B}) = \mathbb{E}_{t\sim\mathcal{B}}[(\mathcal{Q}_{\phi_i}(o,a) - (r + \gamma(1-d)\mathcal{T}))^2] \tag{3}$$

where $t = (o, a, o^{'}, r, d)$ is a tuple with observation o, action a, reward r and done signal d, $\mathcal{B}$ is the replay buffer, and $\mathcal{T}$ is the target, defined as:

$$\mathcal{T} = (\underset{i=1,2}{min}\, \mathcal{Q}^*_{\phi_i}(o^{'}, a^{'}) - \alpha \log \pi_\psi(a^{'}|o^{'})) \tag{4}$$

In the target equation (2), $\mathcal{Q}^*_{\phi_i}$denotes the exponential moving average(EMA) of the parameters of $\mathcal{Q}_{\phi_i}$. Using the EMA has empirically shown to improve training stability in off-policy RL algorithms. The parameter α is a positive entropy coefficient that determines the priority of the entropy maximization over value function optimization.
While the critic is given by $\mathcal{Q}_{\phi_i}$, the actor samples actions from policy π_ϕ and is trained by maximizing the expected return of its actions as in:

$$\mathcal{L}(\phi) = \mathbb{E}_{a\sim\pi}[\mathcal{Q}^\pi(o,a) - \alpha \log \pi_\phi(a|o)] \tag{5}$$

where actions are sampled stochastically from the policy $a_\phi(o, \xi) \sim \tanh(\mu_\phi(o) + \sigma_\phi(o)) \odot \xi$ and $\xi \sim \mathcal{N}(0, I)$ is a standard normalized noise vector.

4 Propose Method: SAC+GAARA

This section describes the structure of our SAC+GAARA method, within which the SAC part isn't changed much from [10], where an actor, a critic, and a critic target network is maintained. The whole framework of SAC+GARRA is demostrated in Figure 1. We will first illustrate the data augmentation module in our GAARA network.

4.1 Data Augmentation Method

When a batch of data is sampled from the buffer, the original picture is cropped in two ways. Center crop is applied to create the anchor, and random crop for the positive samples to do the data augmentation across the batch. Reasons behind cropping method is that using central crop to create anchor ensures data consistency between training and evaluating, and random crop can make our encoder more alert and can recognize different states. The augmentation procedure is shown in Figure 2

4.2 Comparison Module

This module mainly serves to measure the similarity of key and query, where key is generated by encoder from center cropped observation(anchor), and query is generated by target encoder from a random cropped observation(positive sample). We employ the bi-linear inner product to link between query and key, formulated as:

$$sim(q, k) = q^TWk \tag{6}$$

where$q = f_\theta(x_q)$and $k = f_{\overline{\theta}}(x_k)$, θ is the parameter of encoder and $\overline{\theta}$is the parameter of target encoder. We will then use Equation 2 to update encoder with gradient decent. Note that parameters of target encoder shall not be updated here, instead it's updated using exponentially moving average version by:

$$\theta = m\theta + (1-m)\overline{\theta} \tag{7}$$

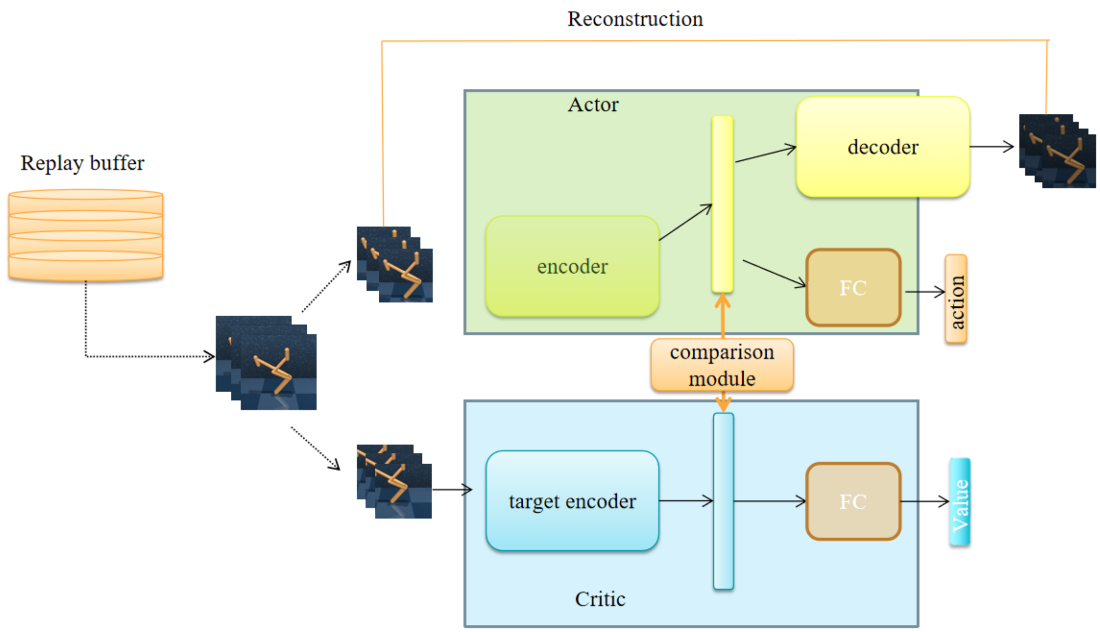

Figure 1: This is a vitualization of our framework, where the encoder is a convolution network transferring an observed image into a laten representation, and the decoder is a trans-convolution network that restructs an image from the vector generated by encoder. Our comparison module serves to learn a matrix W that meatures similarity between query and key.

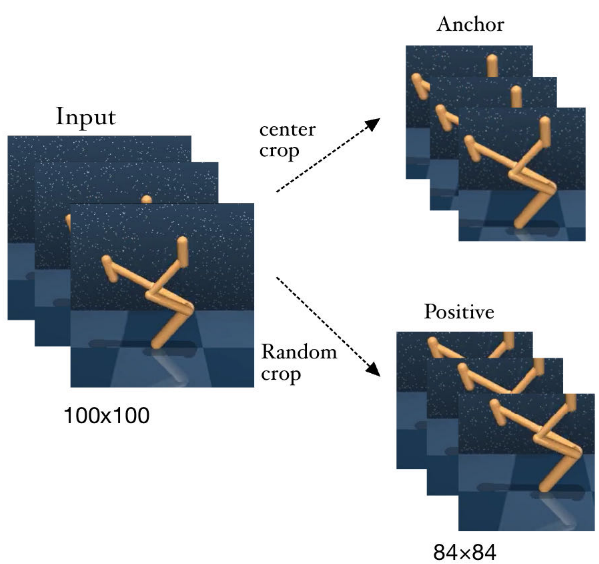

Figure 2: Visually illustrating the process of generating an anchor and its positive using stochastic random crops. Our aspect ratio for cropping is 0.84, i.e, we crop a 84 × 84 image from a 100 × 100 simulation-rendered image. We apply central crop to create the anchor and random crop to create the positive sample. Applying the same crop coordinates across all frames in the stack ensures time-consistent spatial jittering.

4.3 Reconstruction Module

The structure of our encoder and decoder follows auto-encoder(AE), we further imposes a L_2 penalty on the learned representation z_t and weight-decay on the decoder parameters, imitating RAE in [10]:

$$J(RAE) = \mathbb{E}_{\mathbf{o}_t \sim \mathcal{D}}[\log p_\theta(\mathbf{o}_t|\mathbf{z}_t) + \lambda_\mathbf{z}||\mathbf{z}_t||^2 + \lambda_\theta||\theta||^2] \tag{8}$$

where $\mathbf{z}_t = f_\theta(\mathbf{o}_t)$ is the latent representation from encoder and λ_θ, $\lambda_\mathbf{z}$ are hyper parameters

The comparison module is updated right after reconstruction module, indicating that we'll first update encoder along with our decoder using gradient from reconstructive loss, then then the encoder is updated with the comparison module to make it capable of recongnizing augmented data and original data. The whole updating process is illustrated in **??**

4.4 Annealing on Comparison Loss

In the beginning of training process, comparison module can increase encoder's knowledge to the environment and different postures from our agent, which boosts training efficiency and stability. But after some time of encoder training, it is capable to recognize the pixel and extract the latent representation to some extent, and the comparison loss may somehow interference further improvement for both encoder and decoder. That's because the decoder may receive a badly-represented latent space and can't perform well as encoder is still occupied in distinguishing anchor and positive samples.

Driven by this intuition, we proposed the annealing way on comparison loss. Idea behind this method is that the balance between comparison and reconstruction can be dynamically adjusted, and to realize it we add an annealing parameter to manually adjust comparison loss. It is formulated as:

$$loss_{com} = \alpha_{ann} \times log \frac{exp(q^T W k_+)}{exp(q^T W k_+) + \sum_{i=0}^{K-1} exp(q^T W k_i)} \tag{9}$$

where $loss_{com}$is the final comparison loss and α_{ann} is the annealing parameter. Specifically, it's initialized to 1.0, and decreased by a decay factor(10^-6 in our case) every time the encoder and comparison matrix W is updated. Finally it will stay at a lower bound(0.5 in our finally case)and does not change along time. The updating algorithm is shown as algorithm **??**

5 Technical Details

In this section we will give some details of our work, including its history versions and an elaborated process toward the final version. Section 5.1is about our first idea and naive trial of a smart decoder, a first combination of comparison and construction is at section 5.2, where no central crop is applied to create the anchor, and finally we give our failed experiment of letting decoder plays comparison role at section 5.3 for future works.

5.1 Towards a Discriminative Decoder

Our first intuition is to add comparison loss as a part of reconstruction loss and use the combined loss to update encoder and decoder altogether. In this case the total loss will yields:

$$L = \log p_\theta(\mathbf{o}_t|\mathbf{z}_t) + \lambda_\mathbf{z}||\mathbf{z}_t||^2 + \lambda_\theta||\theta||^2 + \lambda_c L_c \tag{10}$$

where L_c can be derived from Formula 2

Results in section 6 shows that this can't get a promising result, and the reason may lies in the essence of decoder as the training process. Decoder's duty is to reconstruct an image from a latent representation, but in the beginning when the encoder is not so well-learned, the two representation may be different, and requiring decoder to follow encoder to train is unrealistic. So not updating decoder using comparison loss may give a better result, and that's what we do in section 5.2

5.2 Separate Comparison Loss

Our next trial lies on separating comparison loss from reconstruction loss. In this case decoder's parameters are updated by the gradient from Formula 8 ,which prevent the decoder from the 'looking

Algorithm 1: Update Decoder And Comparison Module

Input :
Input observation pixels obs;
Target observation pixels $target\_obs$;
Observation anchor (center cropped) obs_anchor;
Observation positive sample (random cropped) obs_pos

Output :
Updated encoder parameters f_θ;
Updated decoder parameters g_ψ;
Updated comparison parameter W

Get latent representation $\mathbf{z}_t$ with $\mathbf{z}_t = f_\theta(obs)$;
Get reconstructed observation rec$_{o}bswith$rec_obs = g$_\psi(\mathbf{z}_t)$
Calculate loss with $L = \log p_\theta(\mathbf{o}_t|\mathbf{z}_t) + \lambda_{\mathbf{z}}||\mathbf{z}_t||^2 + \lambda_\theta||\theta||^2$
Back propagation $loss$ to update f_θ and g_ψ with:

$$\theta \leftarrow \theta - \eta\frac{\partial L}{\partial \theta}$$

and

$$\psi \leftarrow \psi - \eta\frac{\partial L}{\partial \psi}$$

Get latent representation for anchor and pos with $\mathbf{a}_t = f_{\overline{\theta}}(obs\_anchor)$ and $\mathbf{p}_t = f_{\overline{\theta}}(obs\_pos)$
Calculate L_{com} with $loss_{com} = \alpha_{ann} \times log\frac{exp(q^TWk_+)}{exp(q^TWk_+)+\sum_{i=0}^{K-1}exp(q^TWk_i)}$
if $\alpha_{ann} > 0.5$ **then**
  $\alpha_{ann} -= 10^{-6}$
Back propagation $loss$ to update f_θ and W with:

$$\theta \leftarrow \theta - \eta\frac{\partial L}{\partial \theta}$$

and

$$W \leftarrow W - \eta\frac{\partial L}{\partial W}$$

left while turning right' dilemma. When updating, encoder first get gradient together with decoder following Formula 8 , after which it receives another gradient from comparison loss, and updated twice. Note that in this case augmented data(pos) and original data(anchor) are both generated from random crop, which turned out to give sub-optimal results. The reason for this is the inconsistency of testing and training, in the training process encoder always receives a random cropped picture, while in the evaluating process the data fed in the network is the raw data. To deal with that we apply central crop method to generate the anchor data, which in essence is similar to raw pictures, letting along the fact that central crop helps the model to focus on the key part of the picture where the agents acts and give different posture, filtering our unnecessary information

Replacing random crop with central crop is proofed helpful, result of encoder learning from random cropped picture is at Figure 8, and central cropped result is at Figure 3

5.3 Trial to Simplify Comparison Module

After doing the work above, we tried to do some simplification work. Our intuition for this part is to simplify the comparison part, or in other words, to merge it in other modules. Now the comparison part use bi-linear inner product to calculate the similarity between query and key, and we want to harness an existing neural network to replace it, so the decoder agent becomes our target.

The detail behind the idea is as follows: after updating encoder along with decoder using reconstruction loss 8, we fix the decoder, and let it play the role as a discriminator. We use a reconstruction-compare loss to measure the difference between the latent representation of anchor and pos generated by encoder:

$$L = ||o_a - o_p||^2 \tag{11}$$

where o_a is the reconstructed image from latent representation $\mathbf{z}_a$, and o_p is the reconstructed image from latent representation $\mathbf{z}_p$

Sadly this method totally crashed to whole model Figure 9, we'll leave this part for future works to find out the reason.

6 Experimental Results

6.1 Evaluation

We measure the performance of our method and baselines at about 400k environment steps on DMControl. Generally, the DMControl benchmark was set at 500k environment steps, but due to the limitation of computing resources, our time and energy, we just gain the results of at least 360k steps.But that is enough to prove the superiority of our method.
Actually we have four versions in total. The followings are our third release. For another three versions, we also train them on cheetah run task.

6.2 Environments

We benchmark the performance of SAC+GAARA for continuous control environments.We focus on three tasks of DMControl suite: cheetah run, walker walk and finger spin.We first try several weights for GAARA loss on cheetah run task and choose 0.5 as our final model argument.Then we train our model on three following tasks and compare our model performance with SAC+AE baseline.
Our final model arguments are shown hereA.1 in Appendix.

6.3 Result

The result of different arguments on cheetah run task were shown in Figure 3.
Obviously, the choice of comparison argument means a lot in our model.Finally we choose annealing to 0.5, which means it will decrease from 1 to 0.5 when training and then keep constant.
The results of our model on three tasks compared with SAC+AE baseline were shown in Figure 4, Figure5 and Figure6.
In cheetah run task, SAC+GAARA shows amazing performance.Our model converges faster and gain better reward than SAC+AE baseline. In walker walk task, SAC+GAARA learns better than baseline in 360k steps.In finger spin task,SAC+GAARA also reach the standard of baseline.
As a result, SAC+GAARA is talented and powerful on current experimental environment.

The results of our other three versions were shown in Figure 7, Figure8 and Figure9.
In first version and second version, our model was beaten by SAC+AE baseline. It converges early and keep stable. In our last attempt, our model was ruined.The agent refused to learn from the environment.

7 Conclusion

From the process of improving and experimenting with the original model, we can see that combining the method of contrastive learning with the AE architecture can indeed improve the speed of training convergence and the model effect. And in this process, the two students in our group did not simply pursue the reproduction of the paper, but actively explored and sought ways to improve. In the process of reading the original paper and code over and over again, I not only deepened our understanding and comprehension of the original algorithm, but also experienced the fun of exploration, and was able to propose innovative methods and skills to further improve the effect of the model. The final results also confirm our hard work and efforts, proving that GAARA's method really has a lot of usefulness and research space. In the future, if there is time, we will consider further improving the network architecture, enhancing the expression ability of the network, and further improving the effect of the model.

References

[1] M. P. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. 2011.

[2] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[3] H. Durrant-Whyte, N. Roy, and P. Abbeel. Learning to control a low-cost manipulator using data-efficient reinforcement learning. 2011.

[4] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, and P. Abbeel. Soft actor-critic algorithms and applications. 2018.

[5] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. 2020.

[6] Minting Pan, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Iso-dream: Isolating noncontrollable visual dynamics in world models.

[7] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. 2020.

[8] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, Ddl Casas, D. Budden, A. Abdolmaleki, J. Merel, and A. Lefrancq. Deepmind control suite. 2018.

[9] P. Vincent, H. Larochelle, Y. Bengio, and Pierre Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. 2008.

[10] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. 2019.

A Appendix

A.1 hyper parameters of our model

Hyperparameter	Value
action_repeat	4
actor_beta	0.9
actor_log_std_max	2
actor_log_std_min	-10
actor_lr	0.001
actor_update_freq	2
alpha_beta	0.5
alpha_lr	0.0001
batch_size	128
comparison_lambda	1.0
critic_beta	0.9
critic_lr:	0.001
critic_target_update_freq	2
critic_tau	0.01
curl_latent_dim	128
decoder_latent_lambda	1e-06
decoder_lr	0.001
decoder_type	pixel
decoder_update_freq	1
decoder_weight_lambda	1e-07
discount	0.99
encoder_feature_dim	50
encoder_lr	0.001
encoder_tau	0.05
encoder_type	pixel
eval_freq	10000
frame_stack	3
hidden_dim	1024
image_size	84
init_steps	1000
init_temperature	0.1
num_eval_episodes	10
num_filters	32
num_layers	4
num_train_steps	1000000
pre_transform_image_size	100
replay_buffer_capacity	700000
save_buffer	false
save_model	true
save_tb	true
save_video	true
seed	1
work_dir	./log

A.2 Experiment Results

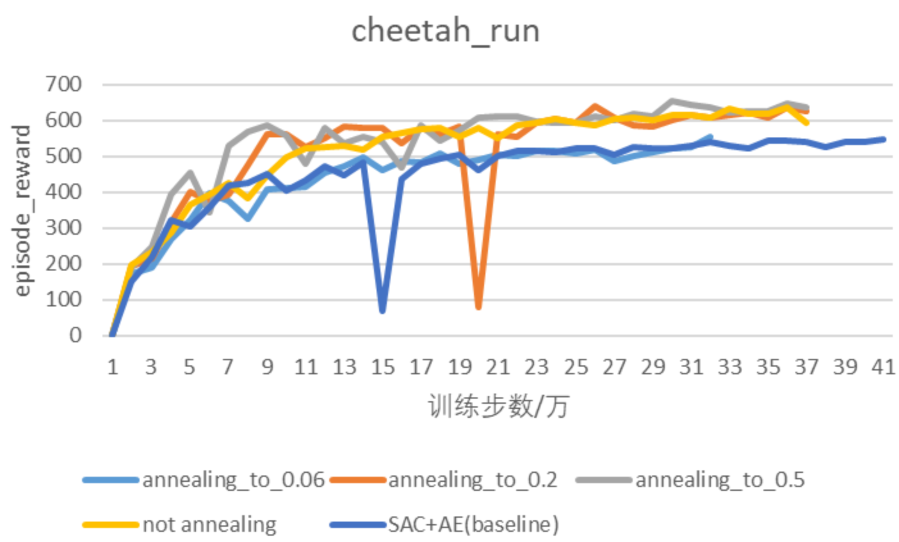

Figure 3: an attempt to adjust the parameters

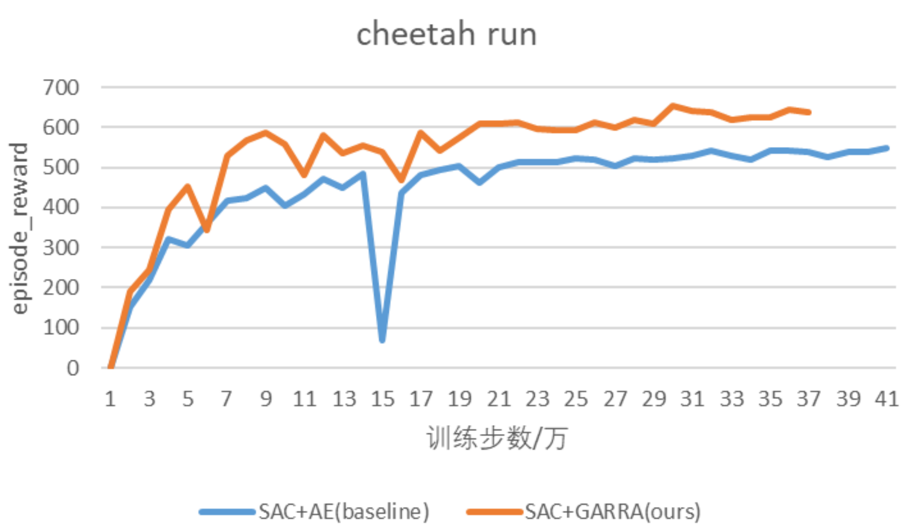

Figure 4: result of cheetah run task

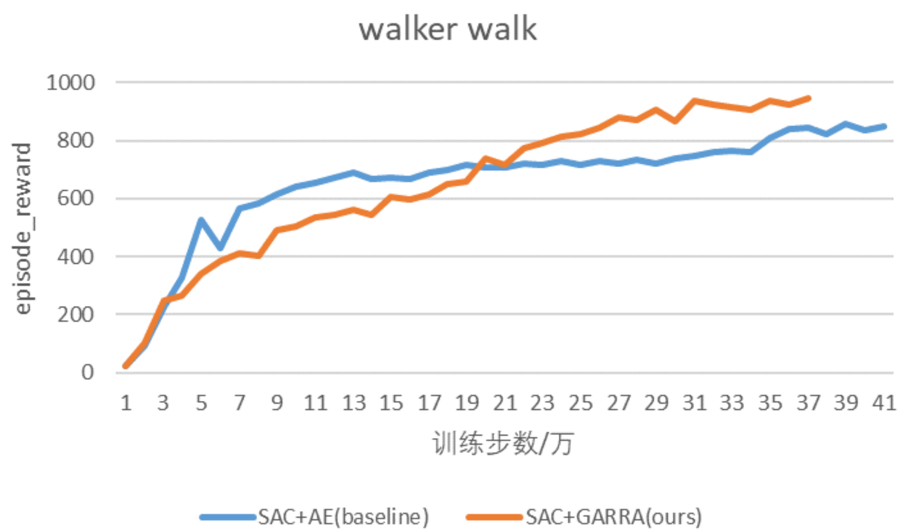

Figure 5: result of walker walk task

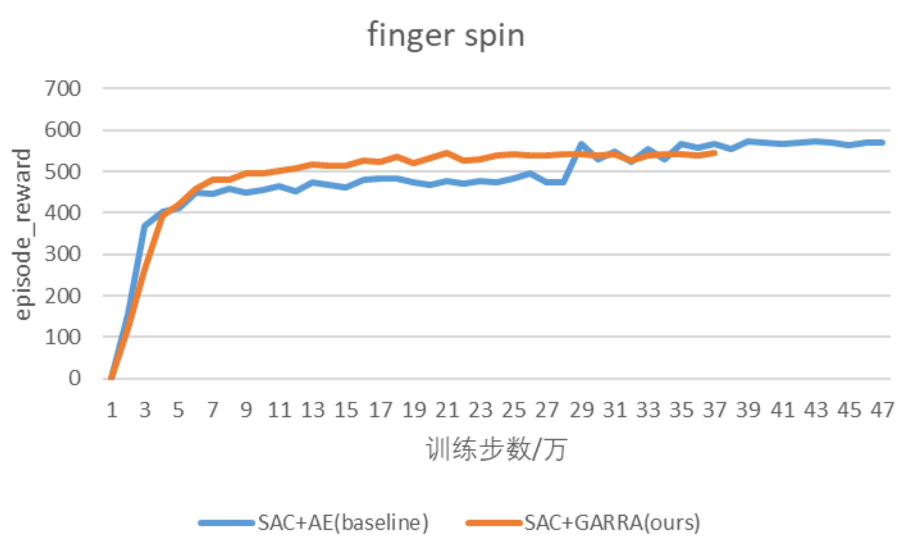

Figure 6: result of finger spin task

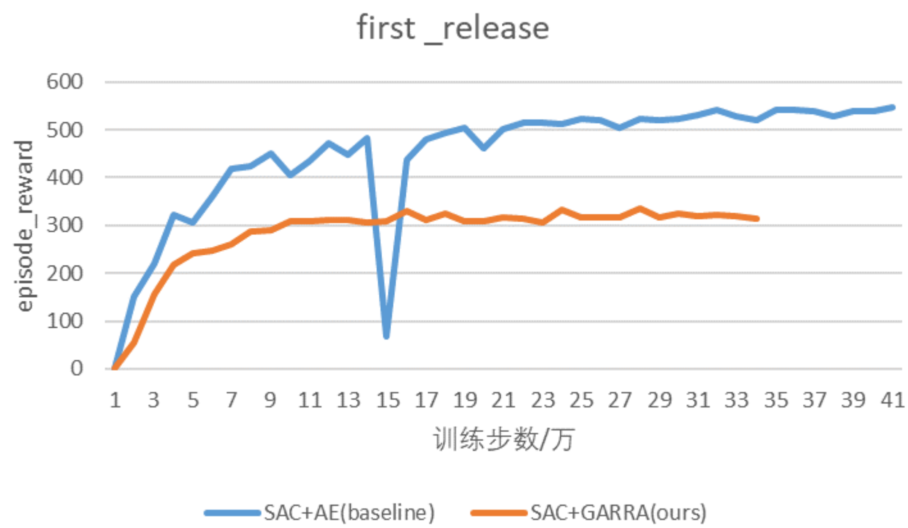

Figure 7: result of first release

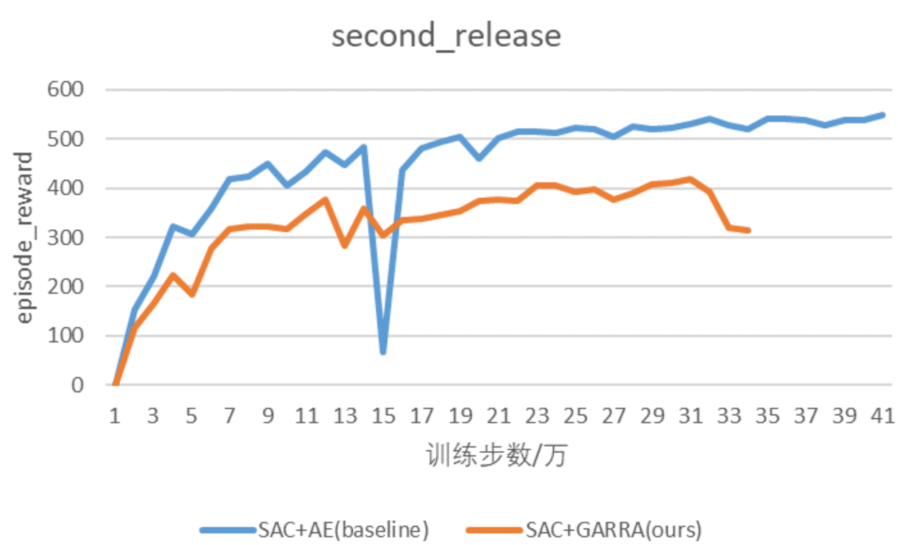

Figure 8: result of second release

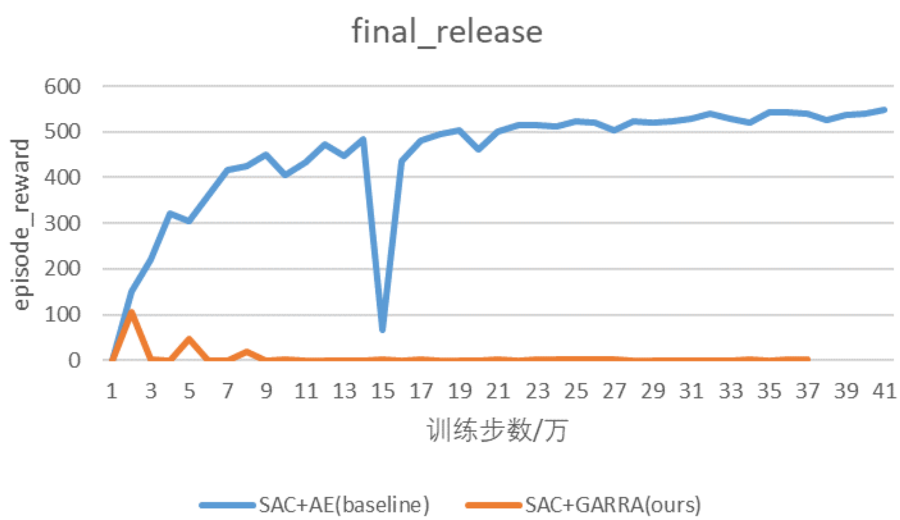

Figure 9: result of final release