Deep Reinforced Accident Anticipation with Visual Explanation Supplementary Materials

Wentao Bao, Qi Yu, Yu Kong
Golisano College of Computing and Information Sciences, Rochester Institute of Technology
{wb6219, qi.yu, yu.kong}@rit.edu

Abstract

This document provides further details about the training algorithms of SAC [4], and the implementation details.

1. Training Algorithm

1.1. Soft Actor Critic

As introduced in Section 3.4 in the main paper, to adapt the soft actor critic (SAC) [4] algorithm to our DRIVE model training, original SAC algorithm needs to be substantially adapted. The Algorithm 1 summarizes the training steps of the improved SAC algorithm.

At first, the transitions including the current state $\mathbf{s}_t$, action $\mathbf{a}_t$, immediate reward r_t, next state $\mathbf{s}_{t+1}$, and the hidden states of LSTM layer $\mathbf{h}_t$ are gathered into the replay buffer $\mathcal{D}$. For each gradient step, a mini-batch of transitions are uniformly sampled from $\mathcal{D}$ to update different model components, including the policy networks (**actor**), Q-networks (**critic**), and **RAE**. As the actor update, automatic entropy tuning, and RAE update are elaborated clearly in the main paper, here we only present more details about how the critic networks are learned during SAC training.

To update the critic, in practice, the Clipped Double Q-learning [2] is used that two identical Q-networks θ_i ($i \in \{1, 2\}$) are maintained. The loss function takes the sum of the losses from the two outputs, i.e., $J(\theta) = \sum_i J(\theta_i)$, where each of them $J(\theta_i)$ is defined as the expectation of mean-squared error:

$$J(\theta_i) = \mathbb{E}\left[\left(Q_{\theta_i}(\mathbf{s}, \mathbf{a}) - y(r, \mathbf{s}', \mathbf{a})\right)^2\right], \tag{15}$$

Here, the optimization target $y(r, \mathbf{s}', \mathbf{a})$ is defined as

$$y(r, \mathbf{s}', \mathbf{a}) = r + \gamma(1-d)\left(\min_{j=1,2} Q_{\bar{\theta}_j}(\mathbf{s}', \hat{\mathbf{a}}') - \alpha \log \pi_\theta(\hat{\mathbf{a}}'|\mathbf{s}')\right) \tag{16}$$

where r is the reward batch, γ is the discounting factor, and d labels whether the sampled transitions are at the last step T. Note that the sate $\mathbf{s}'$ is the batch of next state from replay buffer, while the action $\hat{\mathbf{a}}'$ is sampled from the output of pre-updated policy network π_θ, i.e., $\hat{\mathbf{a}}' \sim \pi_\theta(\cdot|\mathbf{s}')$ which enables SAC to be an off-policy method. The entropy term $\log \pi_\theta(\hat{\mathbf{a}}'|\mathbf{s}')$ is obtained by the Eq. (8) in our main paper.

In this paper, the critic network parameters θ are updated more frequently than other parameters by the gradients of $J(\theta)$ to achieve more stable training. The Table 1 summarizes the hyperparameter setting in experiments. Note that the major hyperparameters are following existing literature [4]. For different datasets, we used the same set of hyperparameters and do not tune them specifically.

Algorithm 1 Improved SAC for the DRIVE Model Training

Require: $\theta_1, \theta_2, \phi, \beta$ ▷ Initial parameters
$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target networks
$\mathcal{D} \leftarrow \emptyset, \mathbf{h}_0 \leftarrow \mathbf{0}$ ▷ Replay buffer and hidden states
for each iteration **do**
  for each environment step **do**
    Sample actions $(\mathbf{a}_t, \mathbf{h}_t) \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{h}_{t-1})$
    Compute state $\mathbf{s}_t$ with actions ▷ See Eq. 2
    Compute reward $r_t = r_A^t + r_F^t$ ▷ See Eq. 4-6
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{h}_t, \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    for each critic update **do**
      $\theta \leftarrow \theta - \lambda \hat{\nabla}_\theta J_Q(\theta)$ ▷ Update by Eq. 15
    end for
    $\phi \leftarrow \phi - \lambda \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update by Eq. 11
    $\alpha \leftarrow \max(\alpha - \lambda_\alpha \hat{\nabla}_\alpha J(\alpha), \alpha_0)$ ▷ See Eq. 12
    $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$ ▷ Update Q-target
    $\beta \leftarrow \beta - \lambda \hat{\nabla}_\beta J_{\text{RAE}}(\beta)$ ▷ Update by Eq. 14
  end for
end for
Ensure: $\theta_1, \theta_2, \phi, \beta$

Table 1. SAC Hyperparameter Settings

Parameters	values
general learning rate (λ)	$3 \cdot 10^{-4}$
temperature learning rate (λ_α)	$5 \cdot 10^{-5}$
discounting factor (γ)	0.99
replay buffer size ($\mathcal{D}$)	10^6
target smoothing coefficient (τ)	0.005
temperature threshold (α_0)	10^{-4}
weight decay (w_0)	10^{-5}
anticipation loss coefficient (w_1)	1
fixation loss coefficient (w_2)	10
latent regularizer coefficient (w_s)	10^{-4}
sparse fixation reward parameter (η)	0.1
gradient updates per time step	4
actor gradient updates per time step	2
dim. of FC/LSTM layers output	64
dim. of latent embedding ($\mathbf{z}$)	64
dim. of state ($\mathbf{s}$)	128
dim. of action ($\mathbf{a}$)	3
sampling batch size	64
video batch size	5

2. Implementation Details

Network Architecture. As shown in Fig. 2 in the main paper, the saliency model is implemented with the existing CNN-based saliency model [1], which takes as input with the size $480 \times 640 \times 3$ and output the feature volume V^t with the size $60 \times 80 \times 64$ by default. The stochastic multi-task agent consists of a shared RAE and two policy networks, i.e., accident prediction and fixation prediction branches. In our implementation, the encoder of RAE consists of three fully-connected (FC) layers and the decoder is symmetric to the encoder. Each policy branch consists of two FC layers and one LSTM layer, followed by an FC layer for predicting means and an FC layer for predicting the variance. ReLU activations are used for all layers except for the last FC output layer. According to the default SAC setting, the output of policy networks $\mathbf{a}_t$ are activated by *tanh* functions so that the values are constrained in $(-1, 1)$. In order to map the values to accident scores a^t and fixation coordinates p^t, we linearly scale the values by

$$a^t = 0.5(\mathbf{a}_t^{(0)} + 1.0) \tag{17}$$

$$p^t = \psi(\mathbf{a}_t^{(1)}, \mathbf{a}_t^{(2)}) \tag{18}$$

where the equation of a^t applied to *tanh* activation is equivalent to *sigmoid* activation on FC layer output. The function ψ maps the scaling factors (within $(-1, 1)$) defined in image space $H \times W$ to the input space $h \times w$. This scaling process is illustrated in Fig. 1.

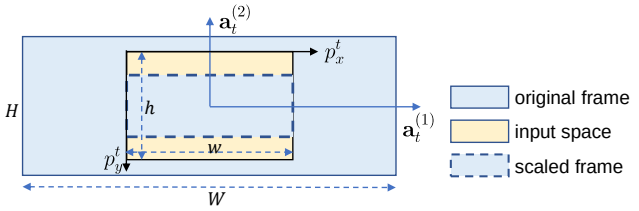

Figure 1. **The scaling process** (ψ). The continuous values $\mathbf{a}_t^{(1)}$ and $\mathbf{a}_t^{(2)}$ which are within $(-1, 1)$ defined in video frame space $H \times W$ are mapped into the discrete input space $h \times w$ to represent the 2-D coordinates of a fixation point.

Implementation. Our training algorithm is implemented based on the SAC source code[1]. Since the image foveation method [3] incurs computational cost due to the Gaussian pyramid filtering, we implement this algorithm as well as all the DRL environmental components by PyTorch to support for GPU acceleration. For DADA-2000 videos, the positive video clips (contains accident) are obtained by trimming the video into be 5 seconds where the beginning times are placed in the last one second with random jittering, while the negative video clips are randomly sampled without overlap with positive clips. The spatial and temporal resolutions for DADA-2000 videos are reduced with ratio 0.5 and interval 5, respectively, so that 30 time steps are utilized and for each step the observation frames are with the size 330×792. For DAD dataset, we only reduce the temporal resolution with interval 4 so that 25 time steps of each 5-seconds video clip are used.

References

[1] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara. A deep multi-level network for saliency prediction. In *ICPR*, 2016. 2

[2] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018. 1

[3] Wilson S Geisler and Jeffrey S Perry. Real-time foveated multiresolution system for low-bandwidth video communication. In *Human Vision and Electronic Imaging III*, volume 3299, pages 294–305, 1998. 2

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. 1

[1] https://github.com/pranz24/pytorch-soft-actor-critic