强化学习及其应用

Reinforcement Learning and Its Applications

第九章 强化学习的应用

Applications of RL

授课人：周晓飞
zhouxiaofei@iie.ac.cn

2023-6-16

第九章 强化学习的应用

9.1 AlphaGo Zero

9.2 NLP 文本分类

9.3 Policy Evaluation

9.4 Game

9.5 Sequence GAN（用于 NLP、序列学习）

9.6 控制过程

9.7 Visual Target-Driven

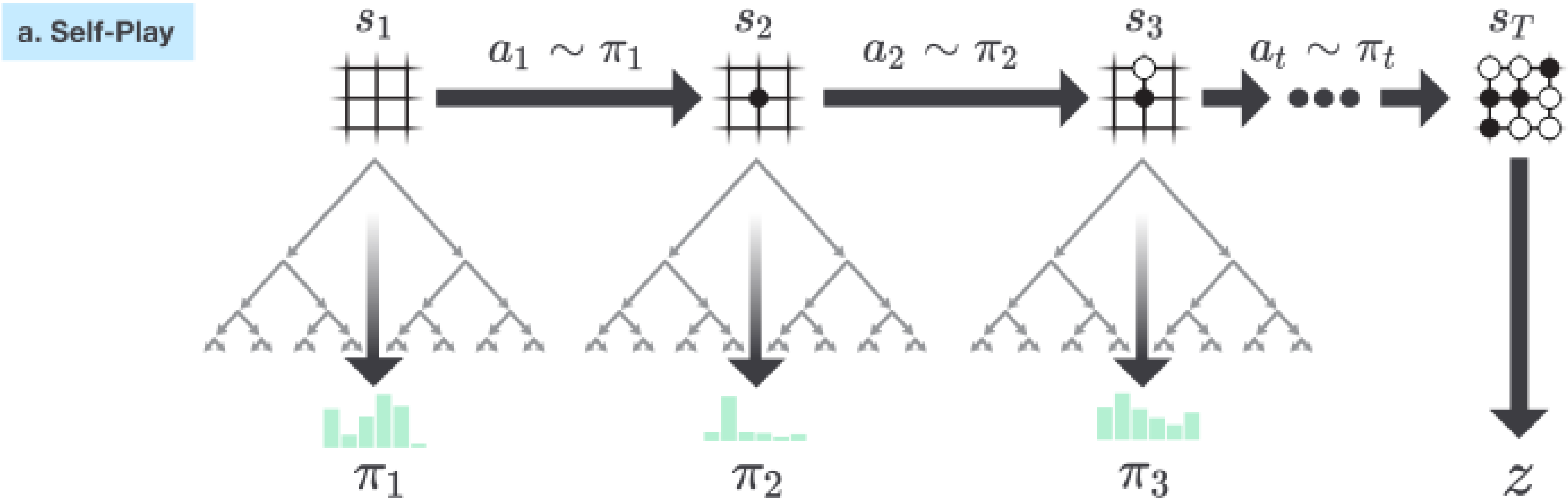
9.8 ChatGPT

AlphaGo Zero

专家棋局不再作为训练样本

MCTS互弈棋局作为训练数据

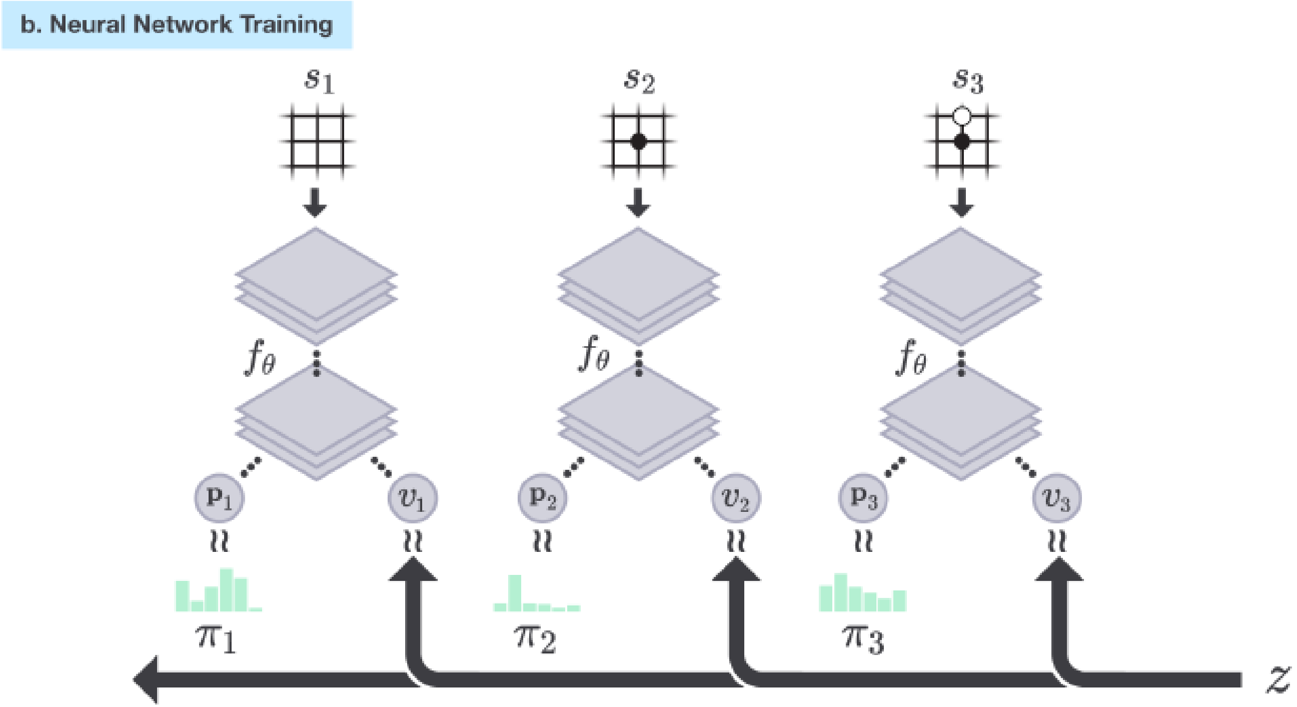
MCTS, $a_t \sim \boldsymbol{\pi}_t$



AlphaGo Zero

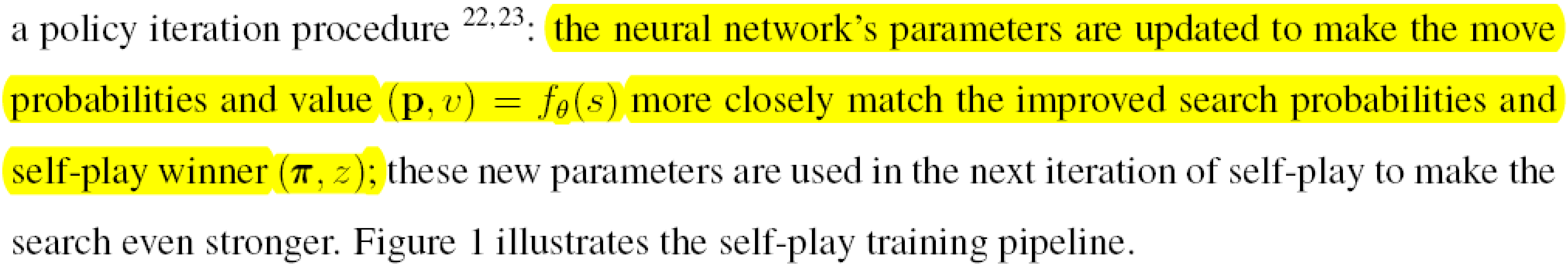
策略网络和值函数网络共享特征参数

$$(\mathbf{p}, v) = f_\theta(s), \qquad l = (z - v)^2 - \boldsymbol{\pi}^\top \log \mathbf{p} + c||\theta||^2 \qquad (1)$$



AlphaGo Zero

目标：MCTS与策略网络具有相同的策略



a policy iteration procedure [22,23]: the neural network’s parameters are updated to make the move probabilities and value $(\mathbf{p}, v) = f_\theta(s)$ more closely match the improved search probabilities and self-play winner $(\boldsymbol{\pi}, z)$; these new parameters are used in the next iteration of self-play to make the search even stronger. Figure 1 illustrates the self-play training pipeline.

AlphaGo Zero

MCTS的原理

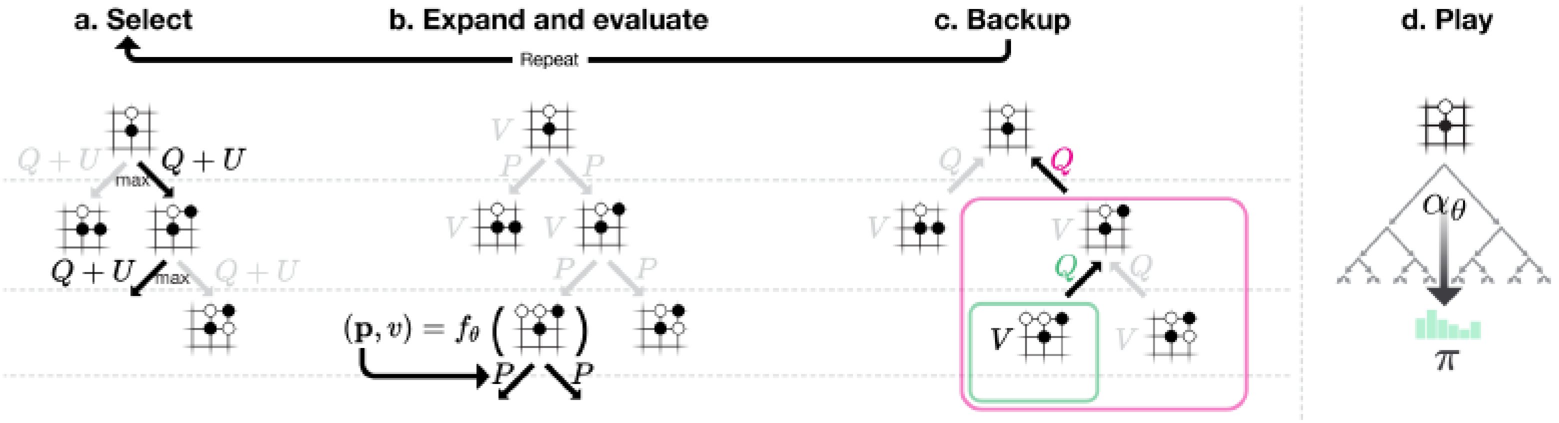
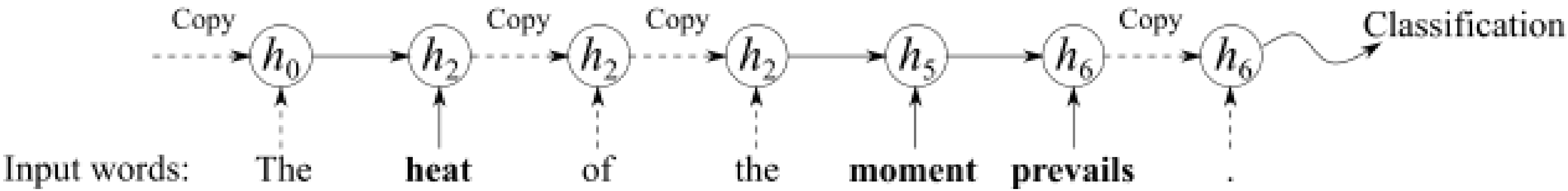


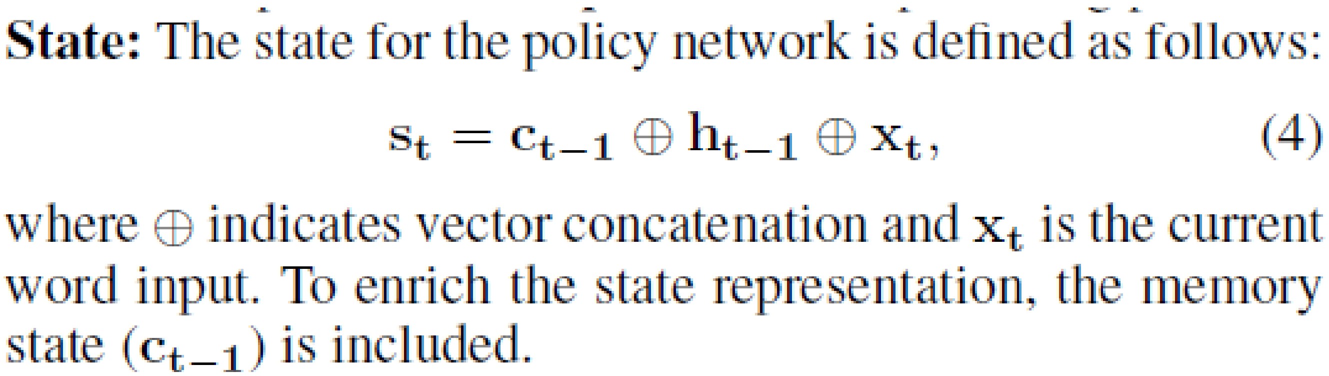
Figure 2: **Monte-Carlo tree search in *AlphaGo Zero*.** **a** Each simulation traverses the tree by selecting the edge with maximum action-value Q, plus an upper confidence bound U that depends on a stored prior probability P and visit count N for that edge (which is incremented once traversed). **b** The leaf node is expanded and the associated position s is evaluated by the neural network $(P(s,\cdot), V(s)) = f_\theta(s)$; the vector of P values are stored in the outgoing edges from s. **c** Action-values Q are updated to track the mean of all evaluations V in the subtree below that action. **d** Once the search is complete, search probabilities $\boldsymbol{\pi}$ are returned, proportional to $N^{1/\tau}$, where N is the visit count of each move from the root state and τ is a parameter controlling temperature.

NLP

Text Classification

Learning Structured Representation for Text Classification via RL (Zhang et al. AAAI 2018)




State: The state for the policy network is defined as follows:

$$\mathbf{s_t} = \mathbf{c_{t-1}} \oplus \mathbf{h_{t-1}} \oplus \mathbf{x_t}, \qquad (4)$$

where $\oplus$ indicates vector concatenation and $\mathbf{x_t}$ is the current word input. To enrich the state representation, the memory state ($\mathbf{c_{t-1}}$) is included.

NLP

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

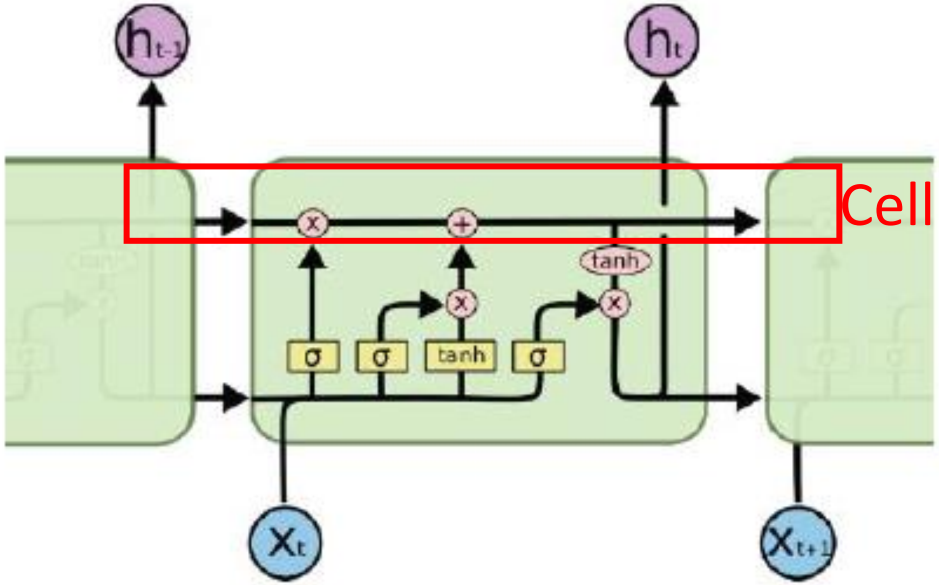
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

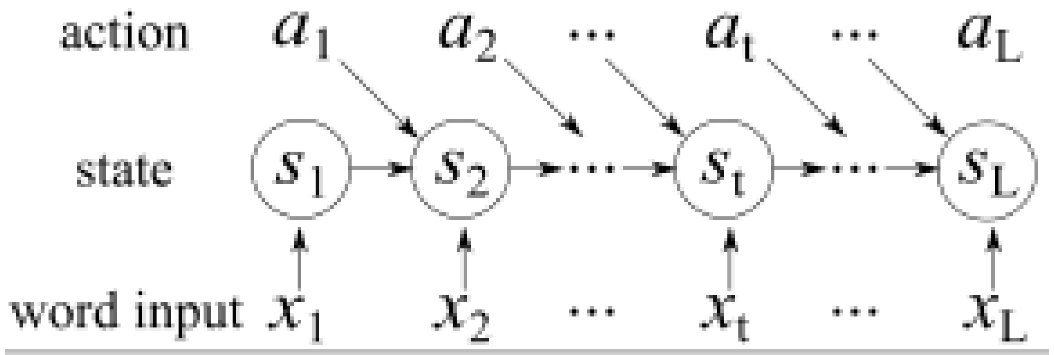
$$o_t = \sigma\left(W_o \ [\, h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

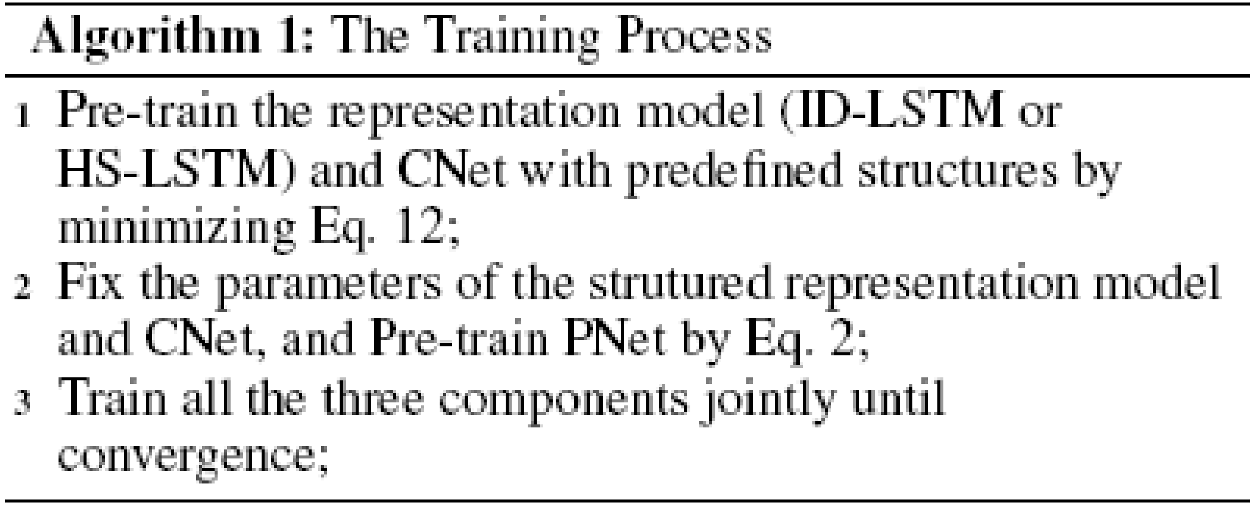


NLP

Action: $\{Retain,\ Delete\}$


action a_1 a_2 ⋯ a_t ⋯ a_L
state s_1 s_2 ⋯ s_t ⋯ s_L
word input x_1 x_2 ⋯ x_t ⋯ x_L

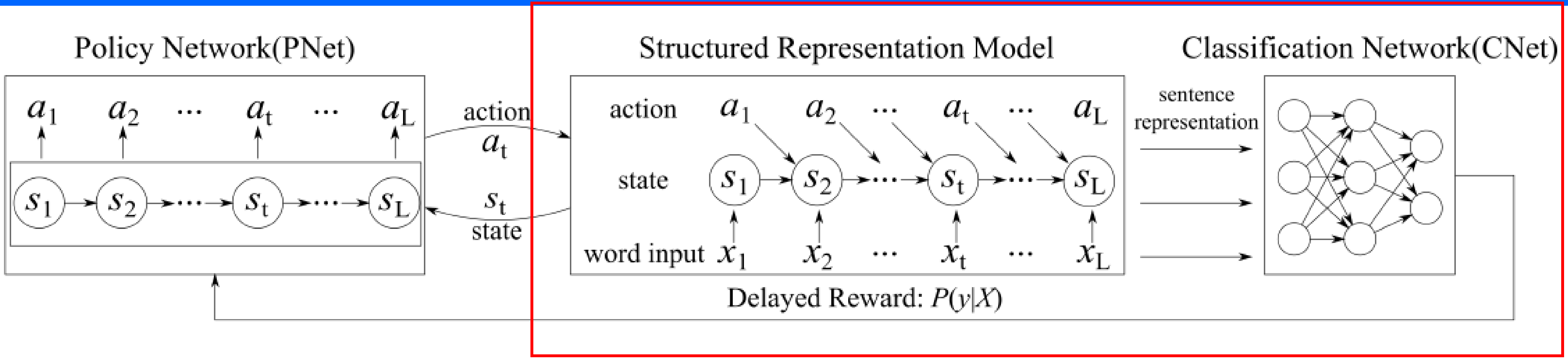
NLP



Algorithm 1: The Training Process

```
Pre-train the representation model (ID-LSTM or HS-LSTM) and CNet with predefined structures by minimizing Eq. 12;
Fix the parameters of the strutured representation model and CNet, and Pre-train PNet by Eq. 2;
Train all the three components jointly until convergence;
```

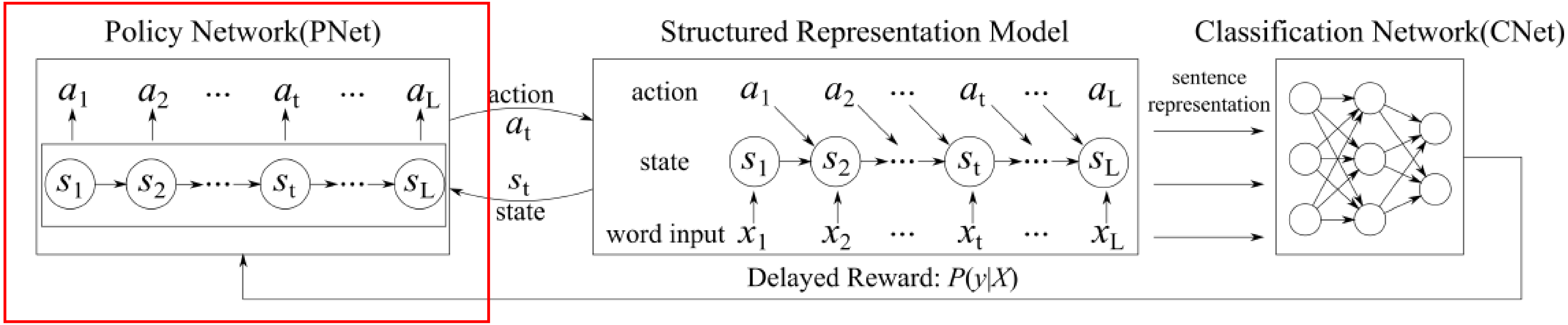
NLP


Policy Network(PNet)
a_1 a_2 ⋯ a_t ⋯ a_L
s_1 s_2 ⋯ s_t ⋯ s_L
action
a_t
s_t
state
Structured Representation Model
action a_1 a_2 ⋯ a_t ⋯ a_L
state s_1 s_2 ⋯ s_t ⋯ s_L
word input x_1 x_2 ⋯ x_t ⋯ x_L
Classification Network(CNet)
sentence representation
Delayed Reward: $P(y|X)$

Classification Network (CNet)

$$\mathcal{L} = \sum_{X \in \mathcal{D}} - \sum_{y=1}^{K} \hat{p}(y, X) \log P(y|X), \qquad (12)$$

NLP


Policy Network(PNet)
Structured Representation Model
Classification Network(CNet)
a_1 a_2 ··· a_t ··· a_L
s_1 s_2 ··· s_t ··· s_L
action a_t
s_t state
action
state
word input x_1 x_2 ··· x_t ··· x_L
sentence representation
Delayed Reward: $P(y|X)$

$$\pi(a_t|\mathbf{s_t};\Theta) = \sigma(\mathbf{W} * \mathbf{s_t} + \mathbf{b})$$

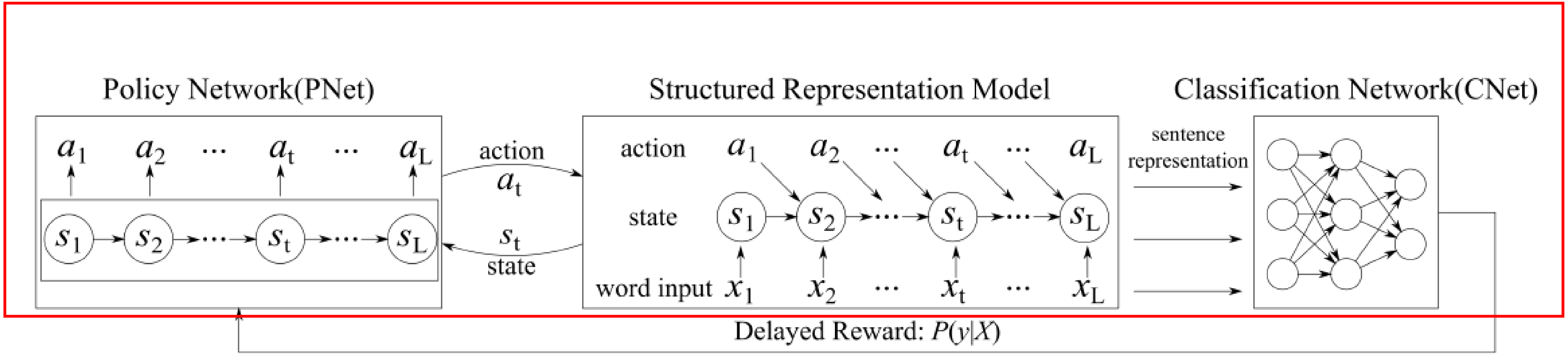
$$\begin{aligned} J(\Theta) &= \mathbb{E}_{(\mathbf{s_t},a_t)\sim P_\Theta(\mathbf{s_t},a_t)} r(\mathbf{s_1}a_1 \cdots \mathbf{s_L}a_L) \\ &= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} P_\Theta(\mathbf{s_1}a_1 \cdots \mathbf{s_L}a_L) R_L \\ &= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} p(\mathbf{s_1}) \prod_t \pi_\Theta(a_t|\mathbf{s_t}) p(\mathbf{s_{t+1}}|\mathbf{s_t}, a_t) R_L \\ &= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} \prod_t \pi_\Theta(a_t|\mathbf{s_t}) R_L. \end{aligned}$$

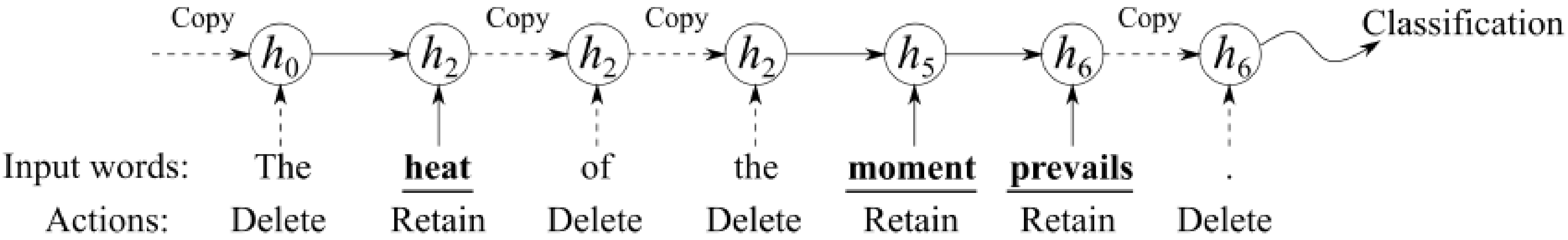
$$\nabla_\Theta J(\Theta) = \sum_{t=1}^{L} R_L \nabla_\Theta \log \pi_\Theta(a_t|\mathbf{s_t}). \qquad (2)$$

$$R_L = \log P(c_g|X) + \gamma L'/L, \qquad (6)$$

where L' denotes the number of deleted words (where the

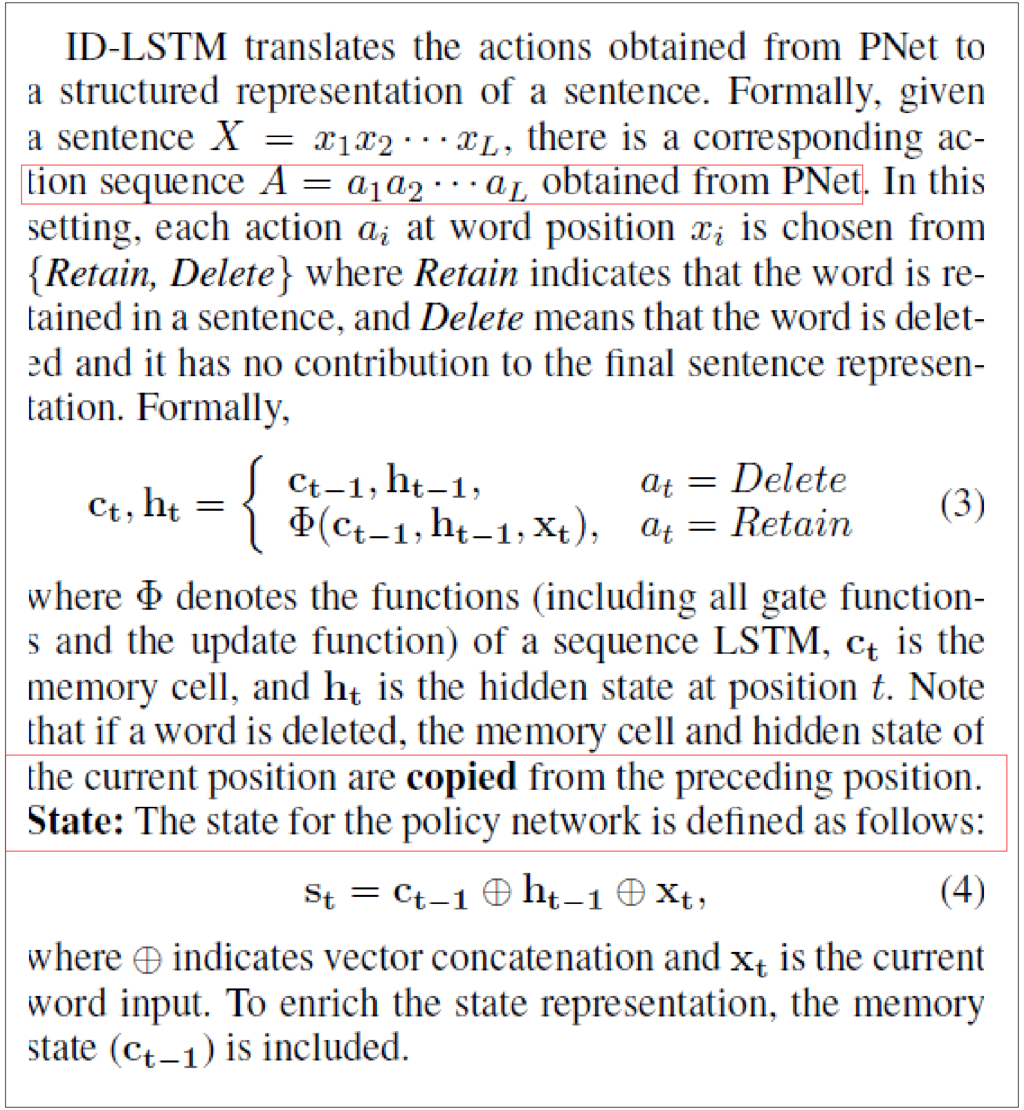
NLP


Policy Network(PNet)
Structured Representation Model
Classification Network(CNet)
a_1 a_2 ··· a_t ··· a_L
s_1 s_2 ··· s_t ··· s_L
action
a_t
s_t
state
action a_1 a_2 ··· a_t ··· a_L
state s_1 s_2 ··· s_t ··· s_L
word input x_1 x_2 ··· x_t ··· x_L
sentence representation
Delayed Reward: $P(y|X)$


Copy
h_0
h_2
Copy
h_2
Copy
h_2
h_5
h_6
Copy
h_6
Classification
Input words: The **heat** of the **moment** **prevails** .
Actions: Delete Retain Delete Delete Retain Retain Delete

(a) Information Distilled LSTM (ID-LSTM)

NLP



ID-LSTM translates the actions obtained from PNet to a structured representation of a sentence. Formally, given a sentence $X = x_1 x_2 \cdots x_L$, there is a corresponding action sequence $A = a_1 a_2 \cdots a_L$ obtained from PNet. In this setting, each action a_i at word position x_i is chosen from $\{$*Retain, Delete*$\}$ where *Retain* indicates that the word is retained in a sentence, and *Delete* means that the word is deleted and it has no contribution to the final sentence representation. Formally,

$$\mathbf{c_t}, \mathbf{h_t} = \begin{cases} \mathbf{c_{t-1}}, \mathbf{h_{t-1}}, & a_t = \textit{Delete} \\ \Phi(\mathbf{c_{t-1}}, \mathbf{h_{t-1}}, \mathbf{x_t}), & a_t = \textit{Retain} \end{cases} \quad (3)$$

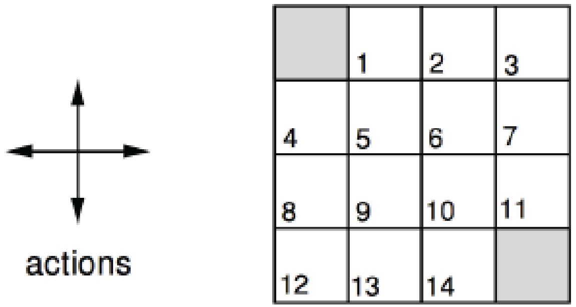
where Φ denotes the functions (including all gate functions and the update function) of a sequence LSTM, $\mathbf{c_t}$ is the memory cell, and $\mathbf{h_t}$ is the hidden state at position t. Note that if a word is deleted, the memory cell and hidden state of the current position are **copied** from the preceding position.
State: The state for the policy network is defined as follows:

$$\mathbf{s_t} = \mathbf{c_{t-1}} \oplus \mathbf{h_{t-1}} \oplus \mathbf{x_t}, \quad (4)$$

where $\oplus$ indicates vector concatenation and $\mathbf{x_t}$ is the current word input. To enrich the state representation, the memory state ($\mathbf{c_{t-1}}$) is included.

Policy Evaluation

Small Gridworld



$r = -1$
on all transitions

- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states $1, \ldots, 14$
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Policy Evaluation

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\left(\underbrace{\boxed{\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')}}_{Q(s, a)}\right)$$

本问题中的 $P^a_{ss'} = 1$

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\left(\underbrace{\boxed{\mathcal{R}_s^a + \gamma\, v_\pi(s')}}_{Q(s, a)}\right)$$

v_k for the Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

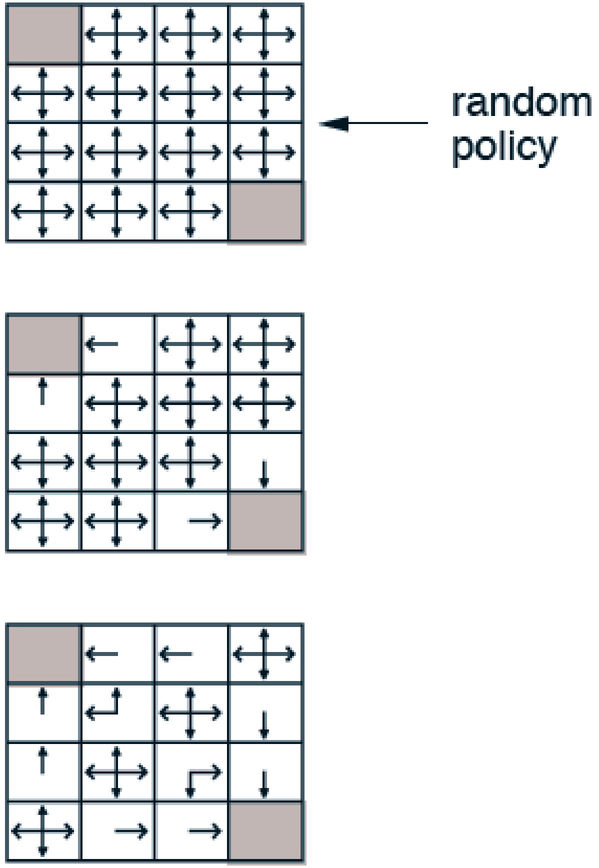
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

Greedy Policy w.r.t. v_k



Policy Evaluation

左侧方法是给定策略的最优估计值，直到该策略分布下的值收敛，然后再优化策略.

如果用贪婪的策略观察，策略很快已经收敛了. 如右图

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

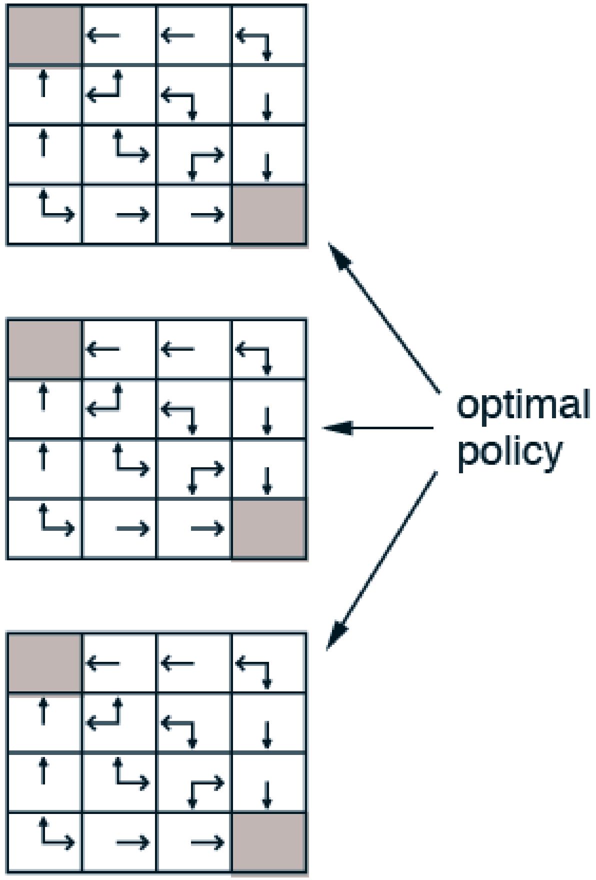
$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

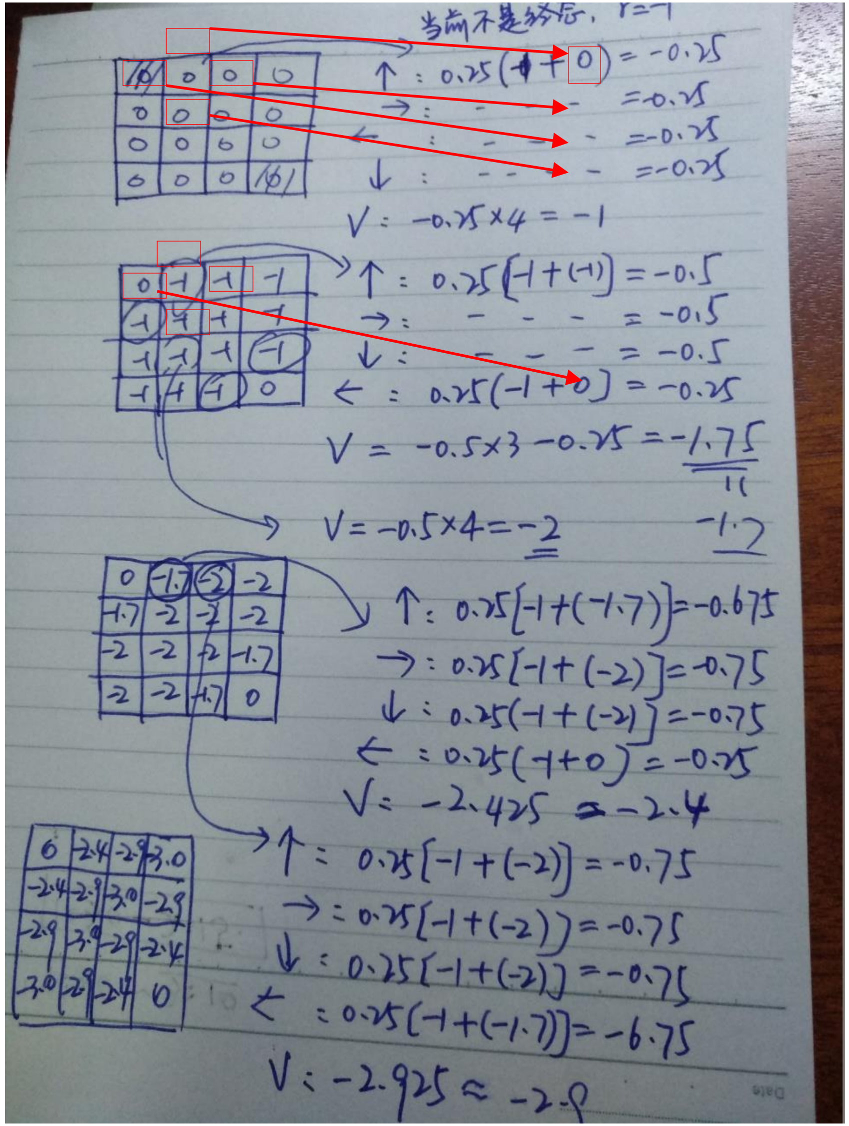
$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

值收敛了



Policy Evaluation



当前不是终点，r=-1

/0/	0	0	0
0	0	0	0
0	0	0	0
0	0	0	/0/

↑ : 0.25(-1 + 0) = -0.25

→ : – – – = -0.25

← : – – – = -0.25

↓ : – – – = -0.25

V : -0.25×4 = -1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

↑ : 0.25[-1 + (-1)] = -0.5

→ : – – – = -0.5

↓ : – – – = -0.5

← : 0.25(-1 + 0] = -0.25

V = -0.5×3 - 0.25 = -1.75 ≈ -1.7

V = -0.5×4 = -2

0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0

↑ : 0.25[-1 + (-1.7)] = -0.675

→ : 0.25[-1 + (-2)] = -0.75

↓ : 0.25(-1 + (-2)] = -0.75

← : 0.25(-1 + 0) = -0.25

V : -2.425 ≈ -2.4

↑ : 0.25[-1 + (-2)] = -0.75

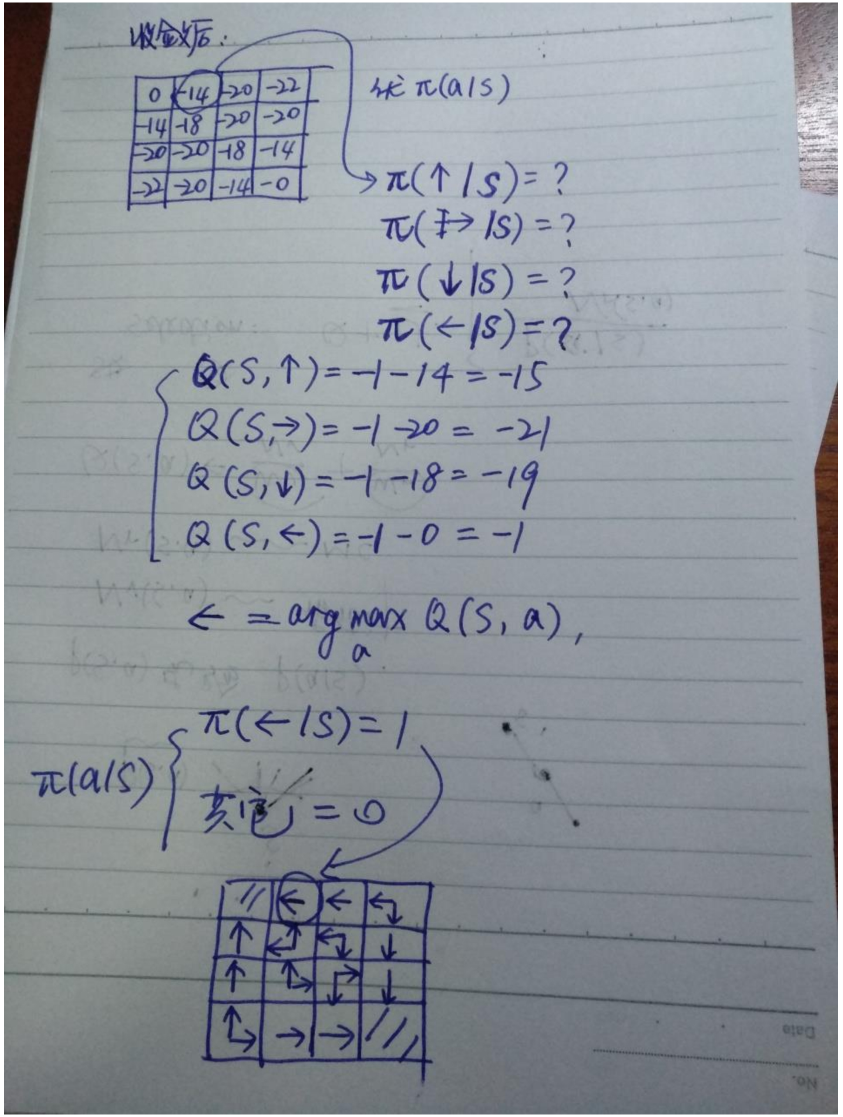
→ : 0.25[-1 + (-2)) = -0.75

↓ : 0.25[-1 + (-2)] = -0.75

← : 0.25(-1 + (-1.7)] = -6.75

V : -2.925 ≈ -2.9

0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0



收敛后：

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	-0

求 π(a|s)

π(↑|s) = ?

π(→|s) = ?

π(↓|s) = ?

π(←|s) = ?

$$\begin{cases} Q(s,\uparrow) = -1-14 = -15 \\ Q(s,\rightarrow) = -1-20 = -21 \\ Q(s,\downarrow) = -1-18 = -19 \\ Q(s,\leftarrow) = -1-0 = -1 \end{cases}$$

$$\leftarrow = \arg\max_a Q(s,a),$$

$$\pi(a|s) \begin{cases} \pi(\leftarrow|s) = 1 \\ \text{其它} = 0 \end{cases}$$

//	←	←	←↓
↑	←↑	←↓	↓
↑	↑→	→↓	↓
↑→	→	→	//

Atari with Deep Reinforcement Learning

DeepMind Technologies: Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning.

Abstract: We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

Game

MDP for Atari

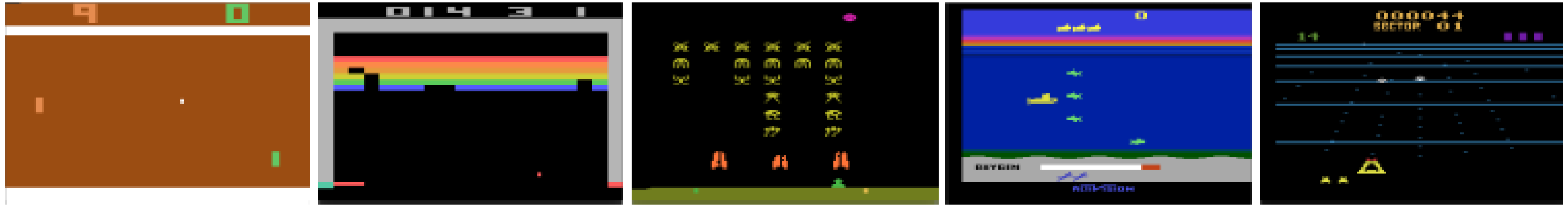


Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

- **S:** $s_t = x_1, a_1, x_2, ..., a_{t-1}, x_t$ **, where *x*: Observation (Image);**

- **A: the set of legal game actions for agent, A ={a$_1$,…,a$_k$}**

Game

Q-Network

S, a, S’ ~ MDP, ε-greedy(policy)

$$L_i\left(\theta_i\right) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q\left(s,a;\theta_i\right)\right)^2\right]$$

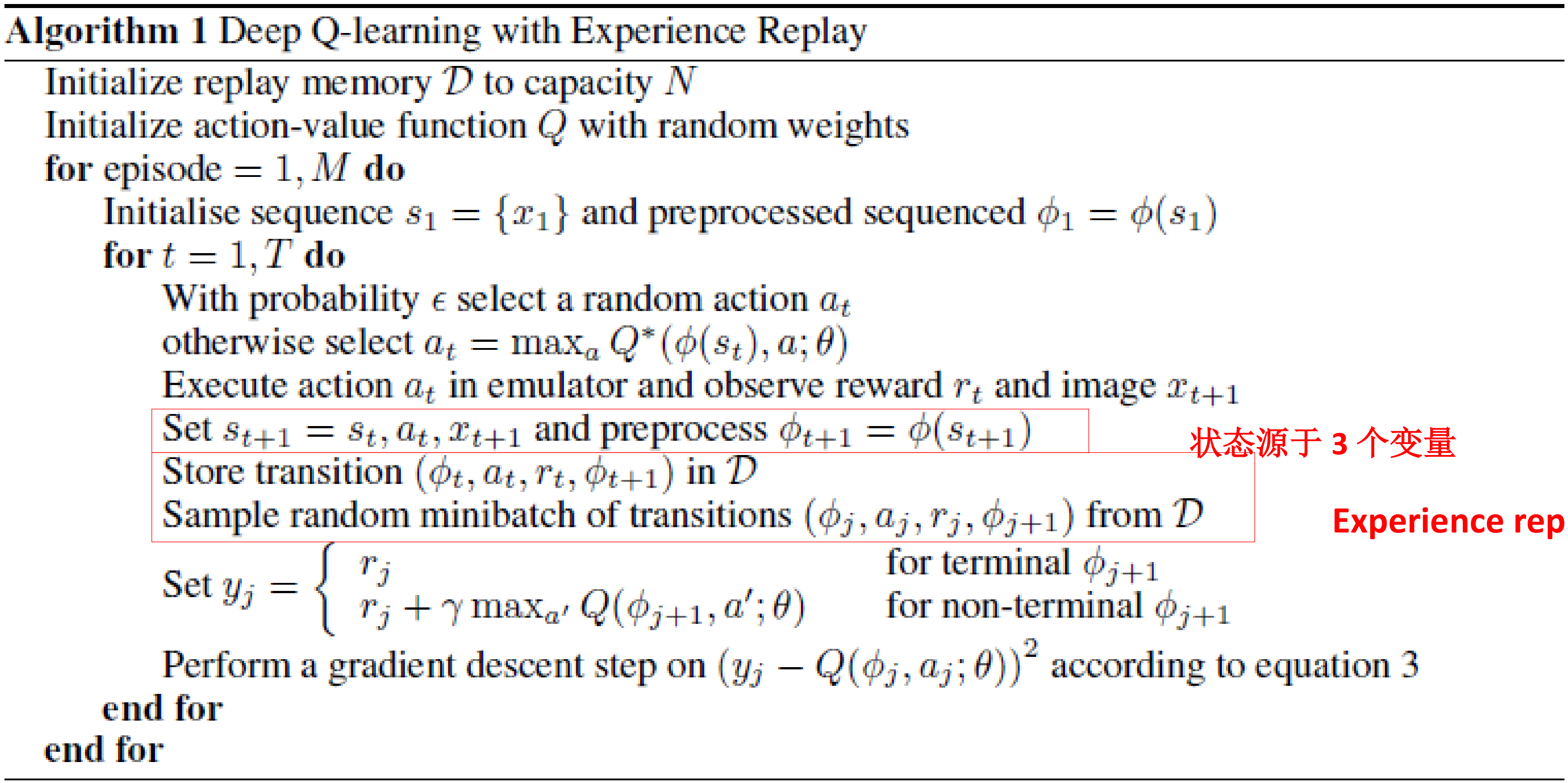
$$\text{where } y_i \;=\; \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1})|s,a\right]$$

$$\nabla_{\theta_i} L_i\left(\theta_i\right) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

Note that this algorithm is *model-free*

Game

Deep Q-Learning



Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
for episode $= 1, M$ do
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for
```

状态源于 3 个变量

Experience replay

Sequence GAN

引入 RL 和 GAN

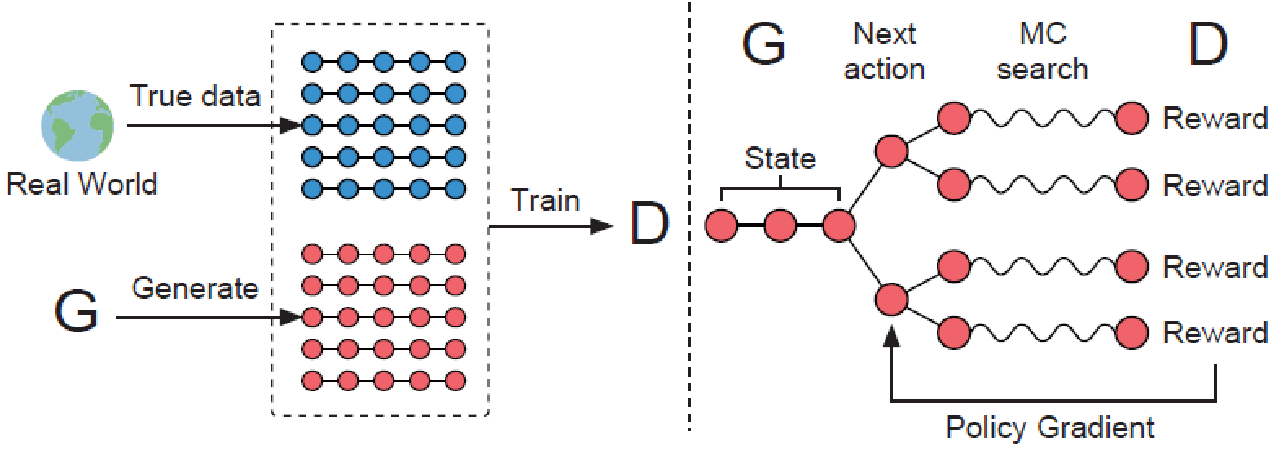


Figure 1: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

Sequence GAN

Generator

S_t ：时刻t的状态，即t时刻之前产生的句子 $(y_1,\ldots y_{t\text{-}1})$

a_t ：时刻t的行动，根据当前状态去选择下一个词 y_t

$G_\theta(y_t \mid Y_{1:t-1})$ ：表示策略，根据当前状态去选择下一个词的概率

注：这里的策略是deterministic的，即在当前状态下选择一个行动之后，转移到的下一个时刻的状态是唯一确定的。

Sequence GAN

Generator

生成器的目标函数：最大化期望回报

$$J(\theta) = \mathbb{E}[R_T|s_0,\theta] = \sum_{y_1\in\mathcal{Y}} G_\theta(y_1|s_0)\cdot Q_{D_\phi}^{G_\theta}(s_0,y_1)$$

重点是如何估计Q函数？

只对于一个完整的句子进行回报评估，用鉴别器的输出作为奖励

$$Q_{D_\phi}^{G_\theta}(a=y_T, s=Y_{1:T-1}) = D_\phi(Y_{1:T})$$

Sequence GAN

Generator

G_β ：Roll-out policy，用来采样的策略，这里用的其实就是当前的生成器。

接下来估计时刻t的Q值，从前状态开始，使用roll-out policy采样后边的T-t个词，一共采样N个句子：

$$\left\{Y_{1:T}^{1}, \ldots, Y_{1:T}^{N}\right\} = \mathrm{MC}^{G_\beta}(Y_{1:t}; N)$$

MC方法：将以上N个句子的回报的均值作为 t时刻的回报。任意时刻的回报可以表示为：

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^{n}),\ Y_{1:T}^{n} \in \mathrm{MC}^{G_\beta}(Y_{1:t}; N) & \text{for} \quad t < T \\ D_\phi(Y_{1:t}) & \text{for} \quad t = T \end{cases} \tag{4}$$

Sequence GAN

Generator

Generative Model：使用的是RNN

$$\boldsymbol{h}_t = g(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

h_t ：当前状态的表示。

x_t ：当前输入单词的embedding。

策略网络：

$$p(y_t|x_1, \ldots, x_t) = z(\boldsymbol{h}_t) = \mathrm{softmax}(\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}_t),$$

Sequence GAN

Discriminator

Discriminative Model：

输入：所有时刻词embedding的拼接，形成TxK的矩阵

$$\mathcal{E}_{1:T} = \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus \ldots \oplus \boldsymbol{x}_T$$

然后，进行窗口大小为l的卷积操作

$$c_i = \rho(\boldsymbol{w} \otimes \mathcal{E}_{i:i+l-1} + b)$$

接着，进行最大池化

$$\tilde{c} = \max\left\{c_1, \ldots, c_{T-l+1}\right\}$$

最后，输入到全连接层，通过sigmoid函数得到最终的输出。

Sequence GAN

Sequence GAN

鉴别器优化：

$$\min_{\phi} -\mathbb{E}_{Y \sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta}[\log(1 - D_\phi(Y))] \tag{5}$$

生成器优化： 策略梯度

$$\begin{aligned}
\nabla_\theta J(\theta) &\simeq \sum_{t=1}^{T} \sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t | Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \\
&= \sum_{t=1}^{T} \sum_{y_t \in \mathcal{Y}} G_\theta(y_t | Y_{1:t-1}) \nabla_\theta \log G_\theta(y_t | Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \\
&= \sum_{t=1}^{T} \mathbb{E}_{y_t \sim G_\theta(y_t | Y_{1:t-1})}[\nabla_\theta \log G_\theta(y_t | Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)]
\end{aligned}$$

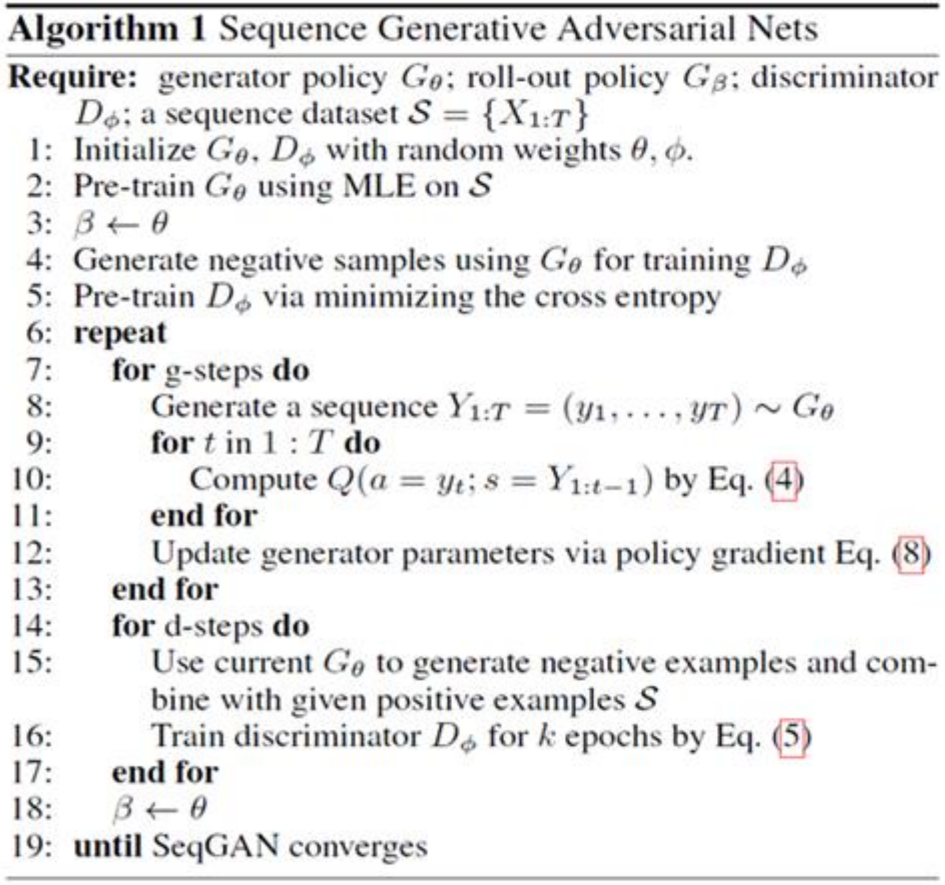
生成器参数更新：梯度上升

$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta) \tag{8}$$

Sequence GAN

Sequence Generative Adversarial Nets

算法流程：



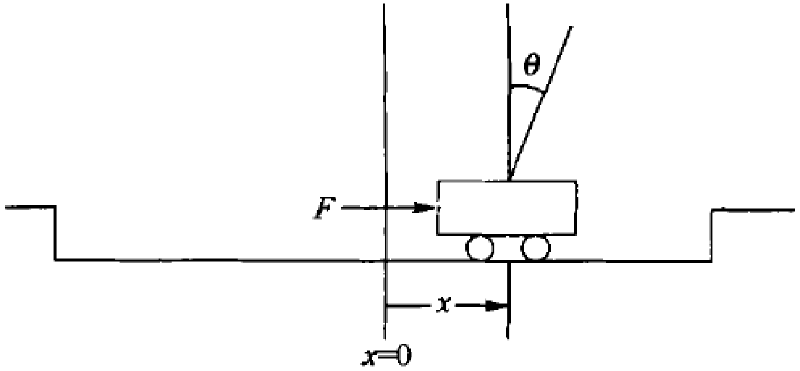
Algorithm 1 Sequence Generative Adversarial Nets

```
Require: generator policy $G_\theta$; roll-out policy $G_\beta$; discriminator
    $D_\phi$; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$
Initialize $G_\theta$, $D_\phi$ with random weights $\theta, \phi$.
Pre-train $G_\theta$ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using $G_\theta$ for training $D_\phi$
Pre-train $D_\phi$ via minimizing the cross entropy
repeat
    for g-steps do
        Generate a sequence $Y_{1:T} = (y_1, \dots, y_T) \sim G_\theta$
        for $t$ in $1 : T$ do
            Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
        end for
        Update generator parameters via policy gradient Eq. (8)
    end for
    for d-steps do
        Use current $G_\theta$ to generate negative examples and combine with given positive examples $\mathcal{S}$
        Train discriminator $D_\phi$ for $k$ epochs by Eq. (5)
    end for
    $\beta \leftarrow \theta$
until SeqGAN converges
```

控制过程

倒立摆

起摆与平衡控制:

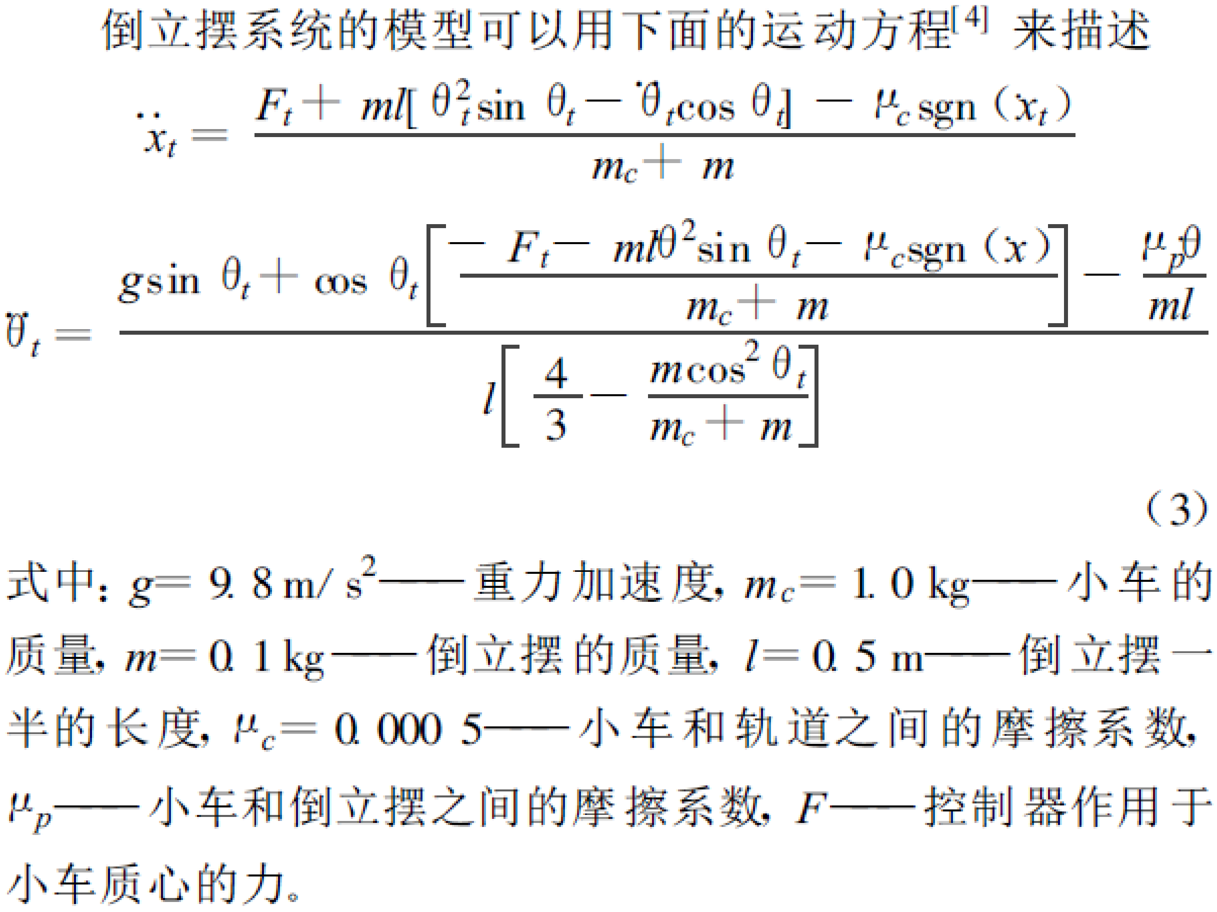


S: $\boldsymbol{S} = [\, x \quad \theta \quad \dot{x} \quad \dot{\theta} \,]^{\mathrm{T}}$

A: F 在每个状态有 5 个可选动作：$F=$ ＋40 N，－40 N，＋15 N，－15 N，0 N。

式中：x—— 小车在轨道上的位置，θ—— 倒立摆偏离垂直方向的角度，x—— 小车的运动速度，θ—— 倒立摆的角速度。

控制过程



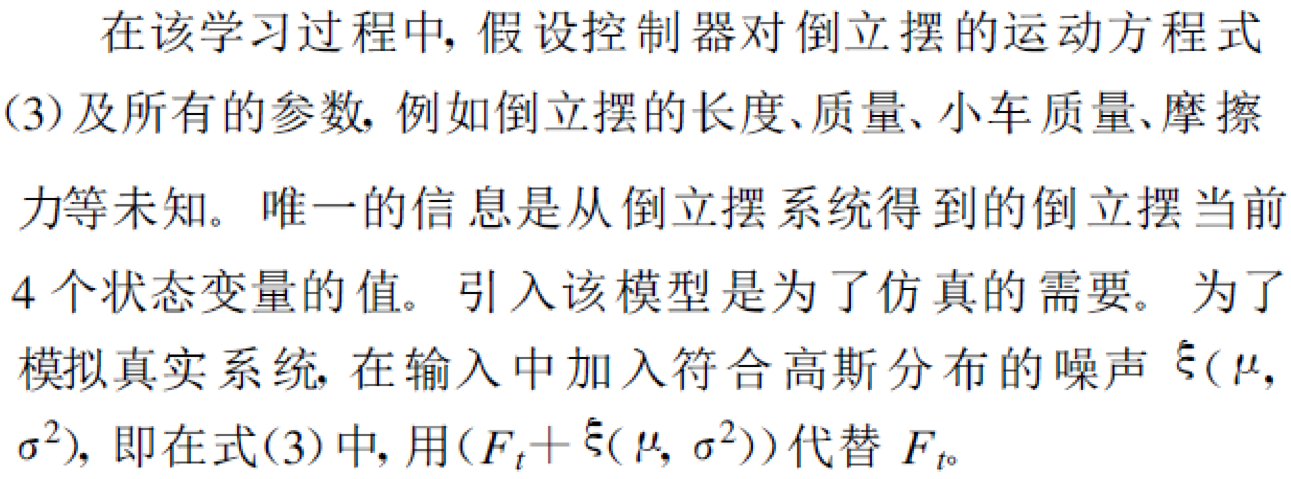
倒立摆系统的模型可以用下面的运动方程[4] 来描述

$$\ddot{x}_t = \frac{F_t + ml[\theta_t^2 \sin\theta_t - \dot{\theta}_t \cos\theta_t] - \mu_c \operatorname{sgn}(x_t)}{m_c + m}$$

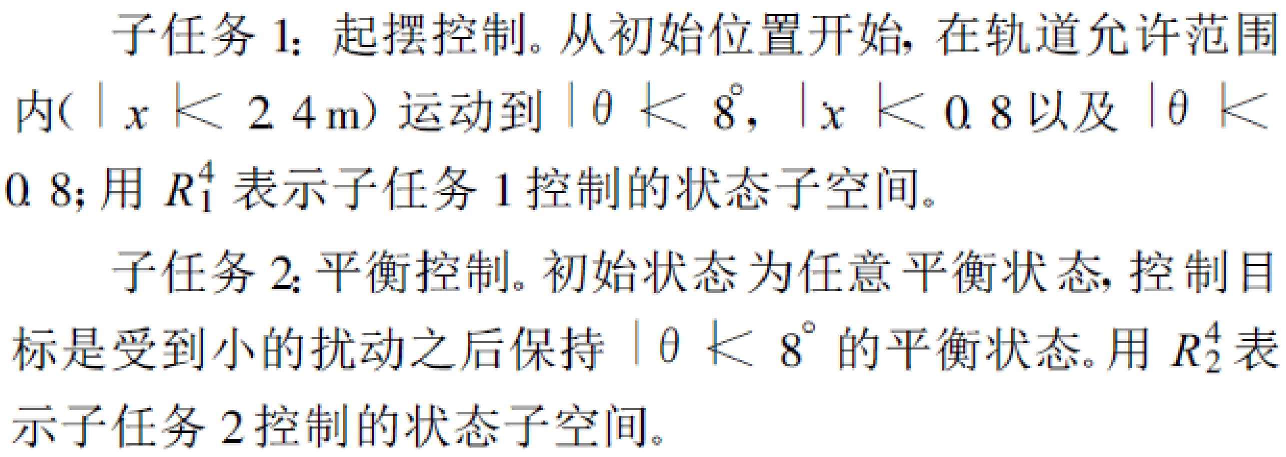
$$\ddot{\theta}_t = \frac{g\sin\theta_t + \cos\theta_t \left[\frac{-F_t - ml\theta^2 \sin\theta_t - \mu_c \operatorname{sgn}(x)}{m_c + m}\right] - \frac{\mu_p \theta}{ml}}{l\left[\frac{4}{3} - \frac{m\cos^2\theta_t}{m_c + m}\right]} \qquad (3)$$

式中：$g = 9.8\ \mathrm{m/s^2}$——重力加速度，$m_c = 1.0$ kg——小车的质量，$m = 0.1$ kg——倒立摆的质量，$l = 0.5$ m——倒立摆一半的长度，$\mu_c = 0.0005$——小车和轨道之间的摩擦系数，μ_p——小车和倒立摆之间的摩擦系数，F——控制器作用于小车质心的力。

控制过程



在该学习过程中，假设控制器对倒立摆的运动方程式(3)及所有的参数，例如倒立摆的长度、质量、小车质量、摩擦力等未知。唯一的信息是从倒立摆系统得到的倒立摆当前4个状态变量的值。引入该模型是为了仿真的需要。为了模拟真实系统，在输入中加入符合高斯分布的噪声 $\xi(\mu, \sigma^2)$，即在式(3)中，用$(F_t+\xi(\mu, \sigma^2))$代替 F_t。



子任务1：起摆控制。从初始位置开始，在轨道允许范围内($|x| < 2.4$ m)运动到 $|\theta| < 8°$，$|x| < 0.8$ 以及 $|\theta| < 0.8$；用 R_1^4 表示子任务1控制的状态子空间。

子任务2：平衡控制。初始状态为任意平衡状态，控制目标是受到小的扰动之后保持 $|\theta| < 8°$ 的平衡状态。用 R_2^4 表示子任务2控制的状态子空间。

控制过程

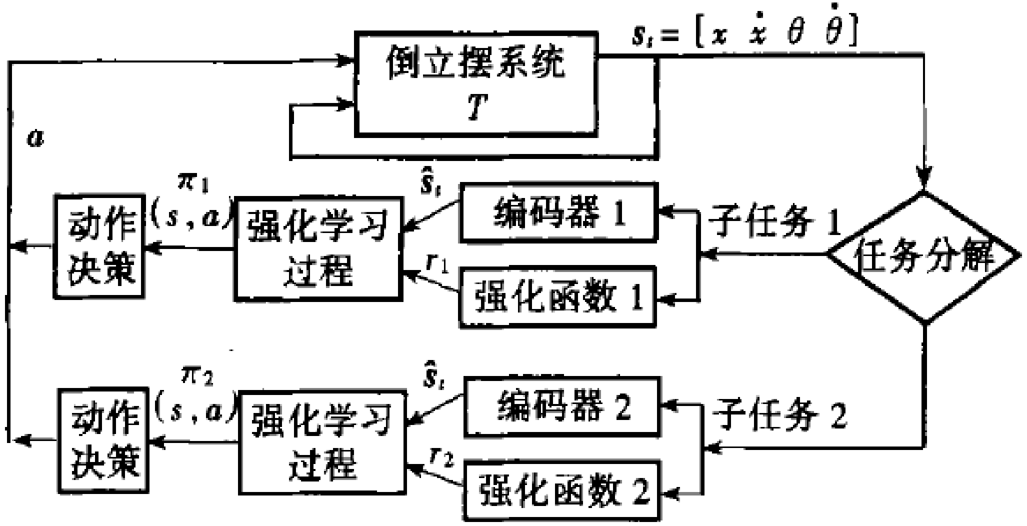


图 2　基于强化学习的倒立的倒立摆控制结构图

图中强化学习和动作决策部分合称为强化学习控制器。编码器的作用是将整个倒立摆的连续状态空间 R^4 转化成离散的状态空间 R^4，并相应地将连续状态向量 s_t 转化为离散状态向量 $\hat{s}_p$ 并对其编码。

本文定义即时奖励/ 惩罚信号 r 为

$$r_1 = \begin{cases} 100 & |\theta| < 8^\circ,\ |x| < 0\ 8 \text{且} |\theta| < 0.\ 8 \\ -50 & x \geqslant 2\ 4 \text{或超过了允许尝试时间} \\ -1 & \text{其它} \end{cases}$$

$$r_2 = \begin{cases} -1 & |x| \geqslant 2\ 4 \text{或} \theta \geqslant 8^\circ \\ 0 & \text{其它} \end{cases}$$

式中：r_1 和 r_2——子任务 1 和子任务 2 的即时奖励/ 惩罚信号。

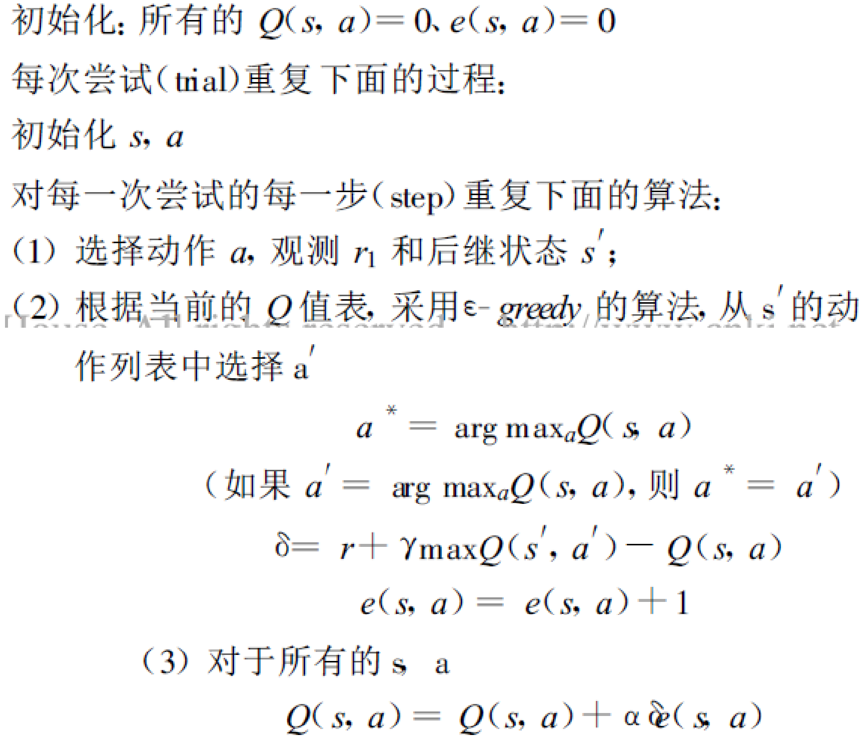
控制过程

可见，episodes：S1, F1, r2, S2, F2, …. 估计 Q（S，a），然后 max$_a$ Q(S, a)。

例如：任务一的 Model Free 方法： MC 估计 Q（S，a）,需要尝试整个 trial。（终态是什么？）

Sarsa(λ) 估计 Q（S，a），然后 max$_a$ Q(S, a)，实时的修正 a。

论文方法 sarsa(λ)结构中由 Q-learning 替换 TD 估计：（当然，也可以采用其他方法）



初始化：所有的 $Q(s, a)=0$、$e(s, a)=0$

每次尝试(trial)重复下面的过程：

初始化 s，a

对每一次尝试的每一步(step)重复下面的算法：

（1）选择动作 a，观测 r_1 和后继状态 s'；

（2）根据当前的 Q 值表，采用ε- *greedy* 的算法，从 s′ 的动作列表中选择 a′

$$a^{*}=\arg\max_a Q(s, a)$$

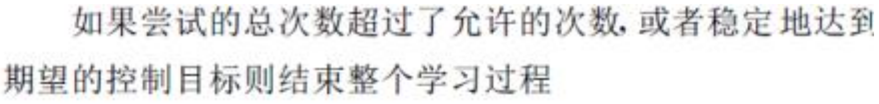
$$（如果\ a'=\arg\max_a Q(s, a)，则\ a^{*}=a'）$$

$$\delta=r+\gamma\max Q(s', a')-Q(s, a)$$

$$e(s, a)=e(s, a)+1$$

（3）对于所有的 s a

$$Q(s, a)=Q(s, a)+\alpha\delta e(s, a)$$



如果尝试的总次数超过了允许的次数，或者稳定地达到期望的控制目标则结束整个学习过程

控制过程

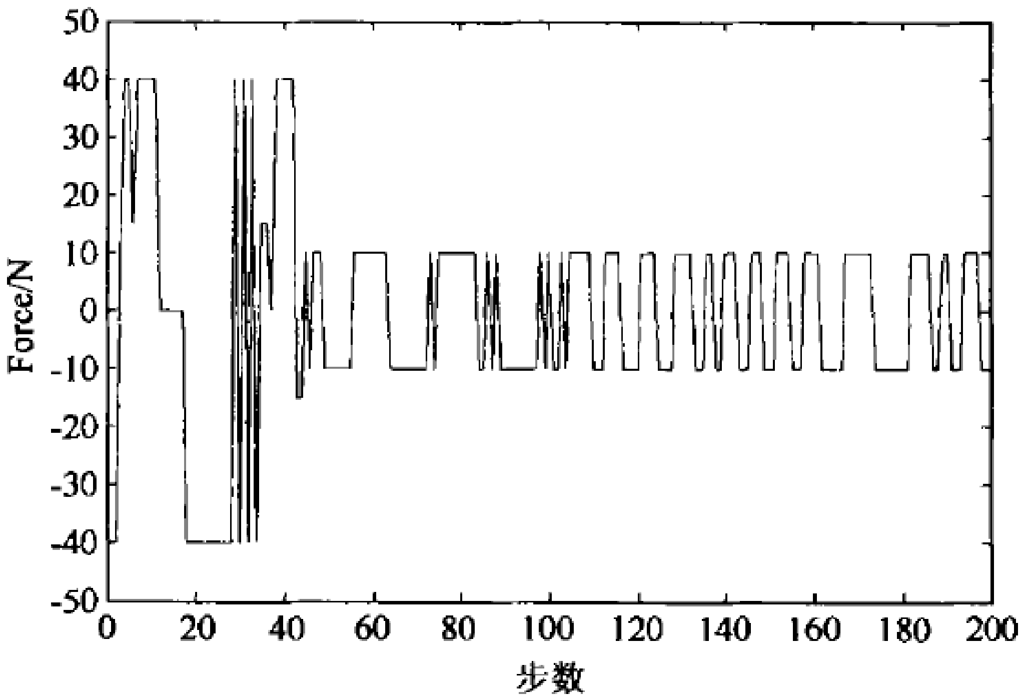
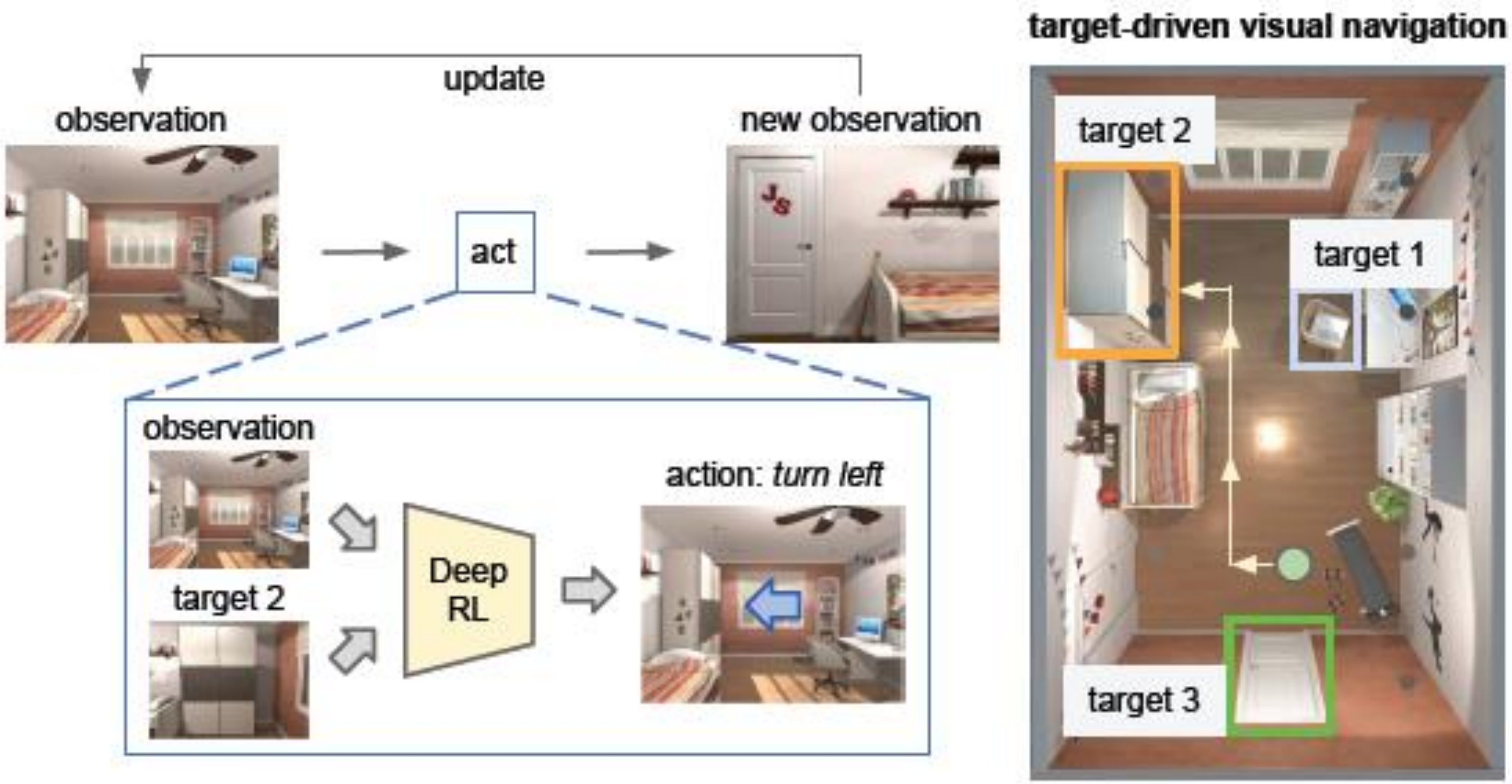


图 8 基于强化学习控制动作 F 的曲线

可见，强化学习是一种试探学习，智能体在没有教师指导情况下与环境进行交互，从环境得到反馈来学习如何适应环境和完成任务。

Visual Target -Driven

Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning (Zhu, et al. 2016 arXiv:1609.05143v1)



Visual Target -Driven

问题描述

The goal of our deep reinforcement learning model is to navigate towards a visual target with a minimum number of steps.

The model takes the current observation and the image of the target as input and generates an action in the 3D environment as the output.

The model learns to navigate to different targets in a scene without re-training.

Visual Target -Driven

Create 3D scenes as model

For example: to create indoor scenes for our framework, we provided reference images to artists to create a 3D scene with the texture and lighting similar to the image.

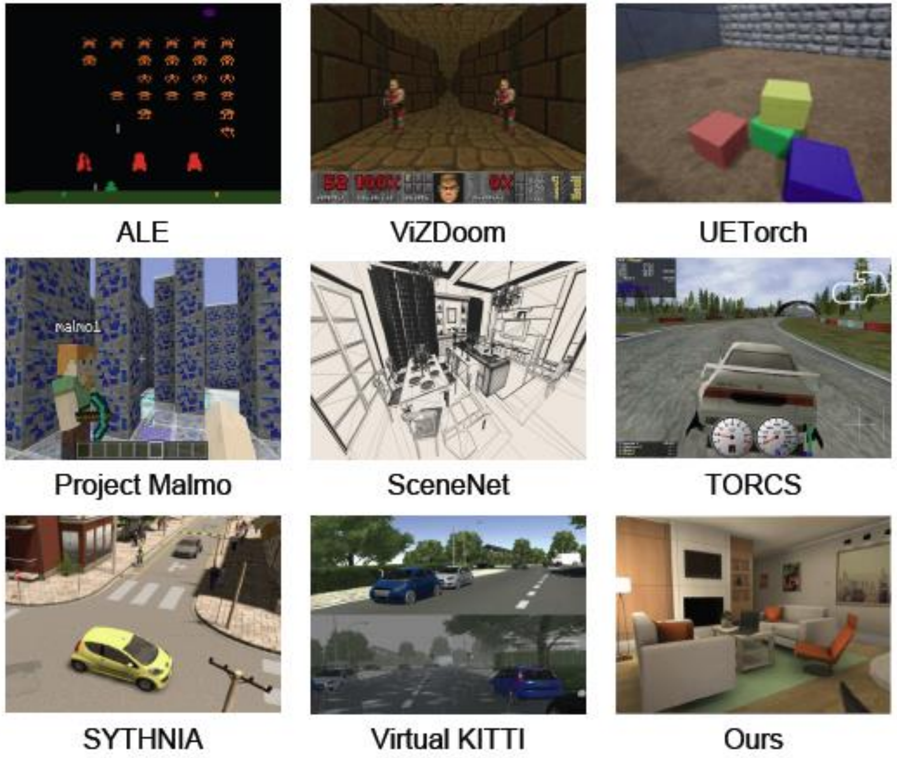


Fig. 2. Screenshots of our framework and other simulated learning frameworks: ALE [36], ViZDoom [37], UETorch [38], Project Malmo [39], SceneNet [40], TORCS [41], SYNTHIA [42], Virtual KITTI [43].

Visual Target -Driven

Many Scenes

So far we have 32 scenes that belong to 4 common scene types in a household environment: kitchen, living room, bedroom, and bathroom. On average, each scene contains 68 object instances.

Visual Target -Driven

The framework provides a rich interaction platform for AI agents.

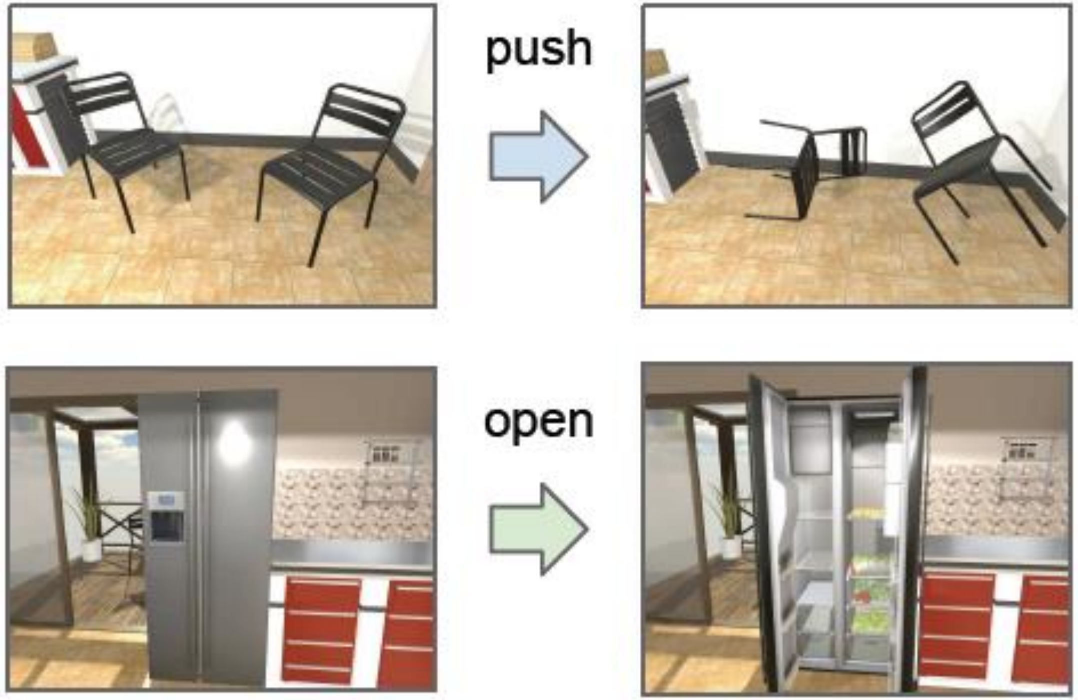


Fig. 3. Our framework provides a rich interaction platform for AI agents. It enables physical interactions, such as pushing or moving objects (the first row), as well as object interactions, such as changing the state of objects (the second row).

Visual Target -Driven

Actions: moving forward, moving backward, turning left, and turning right.

Observations and Goals: Both observations and goals are images taken by the agent’ s RGB camera in its first person view.

Reward design: We focus on minimizing the trajectory length to the navigation targets. Other factors such as energy efficiency could be considered instead. Therefore, we only provide a goal reaching reward (10:0) upon task completion. To encourage shorter trajectories, we add a small time penalty (-0.01) as immediate reward.

Visual Target -Driven

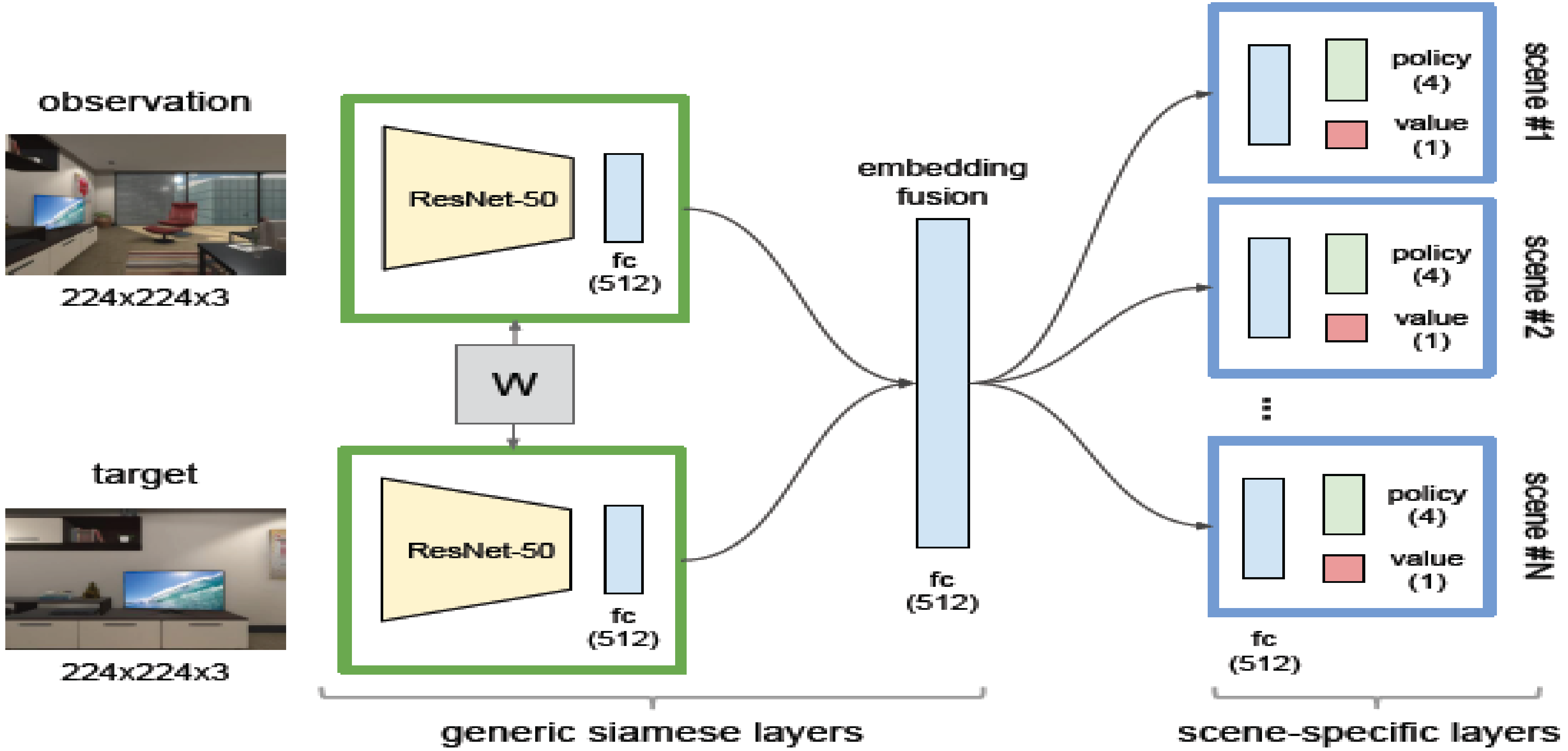
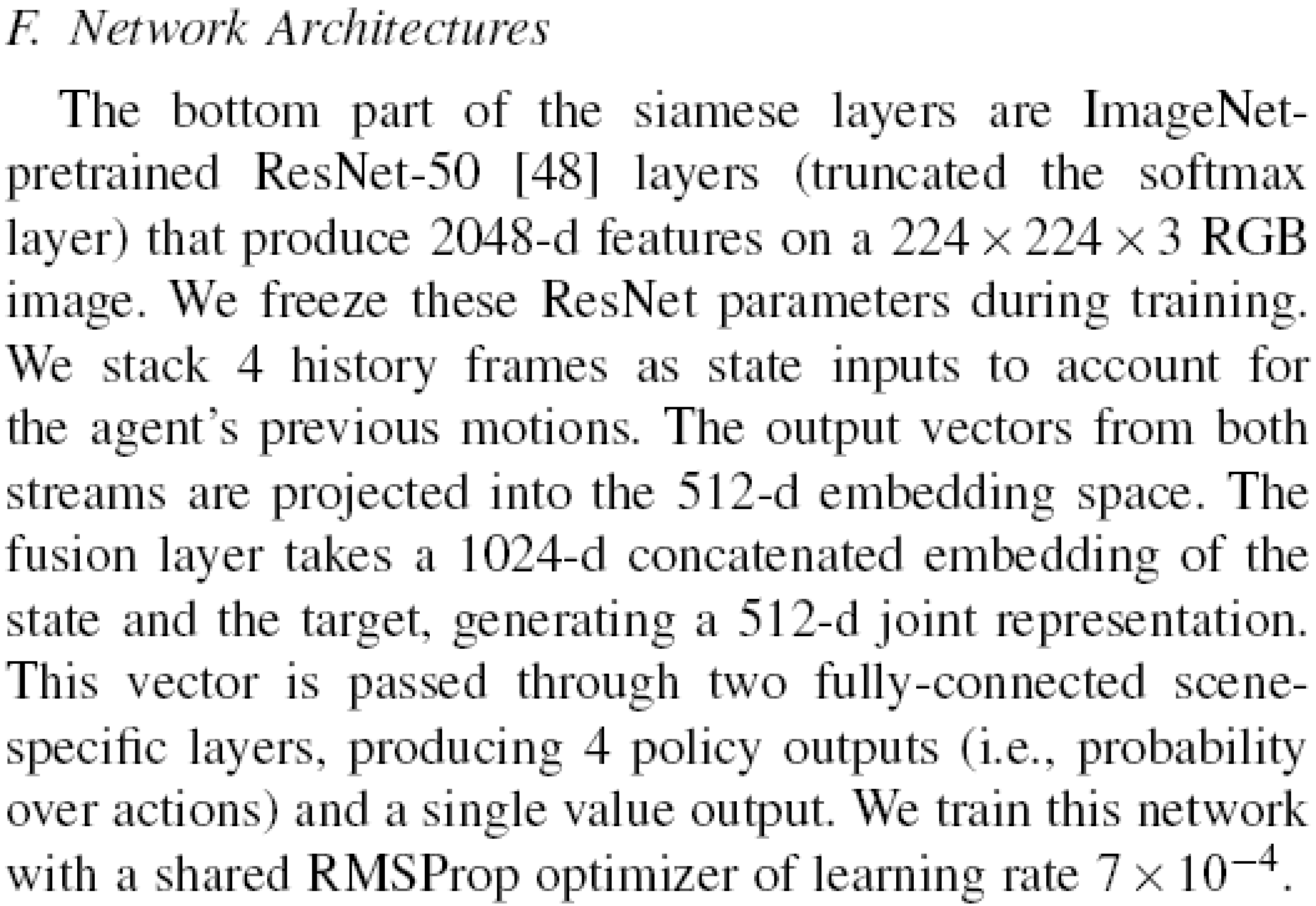


Fig. 4. Network architecture of our deep siamese actor-critic model. The numbers in parentheses show the output dimensions. Layer parameters in the green squares are shared. The ResNet-50 layers (yellow) are pre-trained on ImageNet and fixed during training.

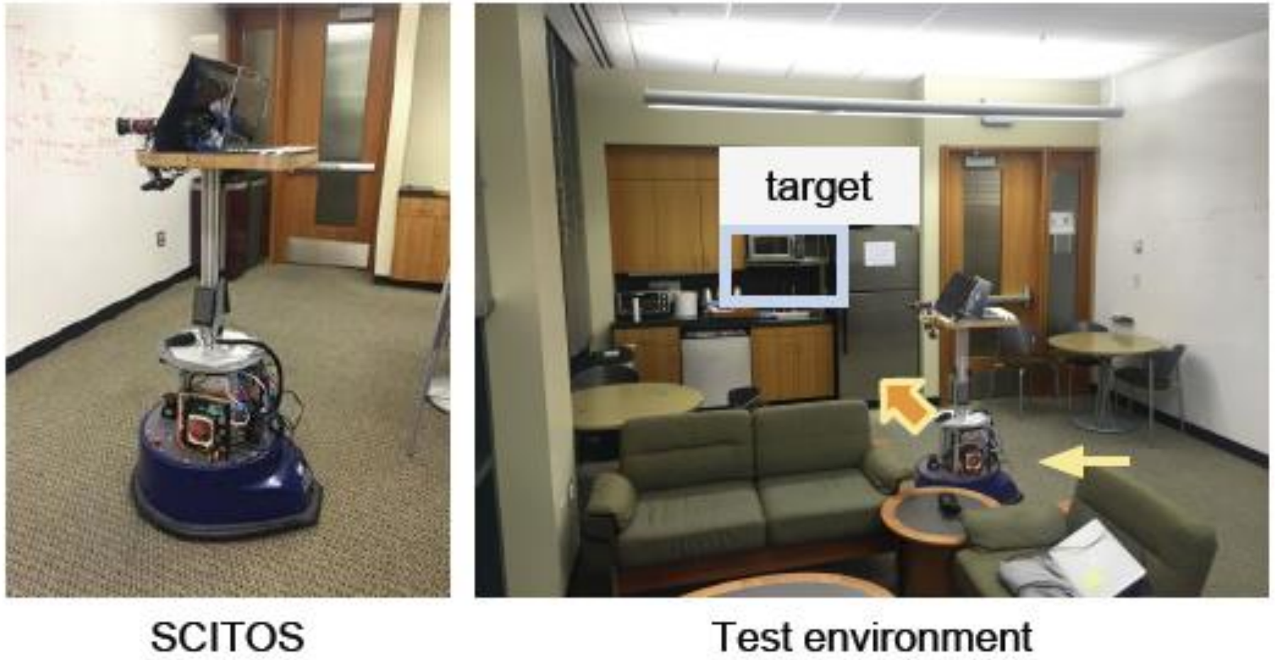
Visual Target -Driven



F. Network Architectures

The bottom part of the siamese layers are ImageNet-pretrained ResNet-50 [48] layers (truncated the softmax layer) that produce 2048-d features on a $224 \times 224 \times 3$ RGB image. We freeze these ResNet parameters during training. We stack 4 history frames as state inputs to account for the agent's previous motions. The output vectors from both streams are projected into the 512-d embedding space. The fusion layer takes a 1024-d concatenated embedding of the state and the target, generating a 512-d joint representation. This vector is passed through two fully-connected scene-specific layers, producing 4 policy outputs (i.e., probability over actions) and a single value output. We train this network with a shared RMSProp optimizer of learning rate 7×10^{-4}.

Visual Target -Driven



SCITOS

Test environment

Fig. 9. **Robot experiment setup.** Our experiments are conducted on a SCITOS mobile robot. On the left, we show a picture of the SCITOS robot. On the right, we show the test environment and one target (microwave) that we have used for evaluation.

ChatGPT 人类强化学习反馈

GPT 家族

- ✧ GPT：　2018.6，1.17 亿参数；生成或无监督预训练
- ✧ GPT-2：2019.2，3 亿参数；多任务学习
- ✧ GPT-3：2020.5，15亿参数；情景学习
- ✧ InstructGPT：2022.1,1750亿参数；
- ✧ ChatGPT：2022.11,13亿参数；

ChatGPT 人类强化学习反馈

InstructGPT、ChatGPT

两个特点：

1. 用户意图对齐：Prompt + 人类标注 的 fine tuning，意图（有用、真实、无害）微调；

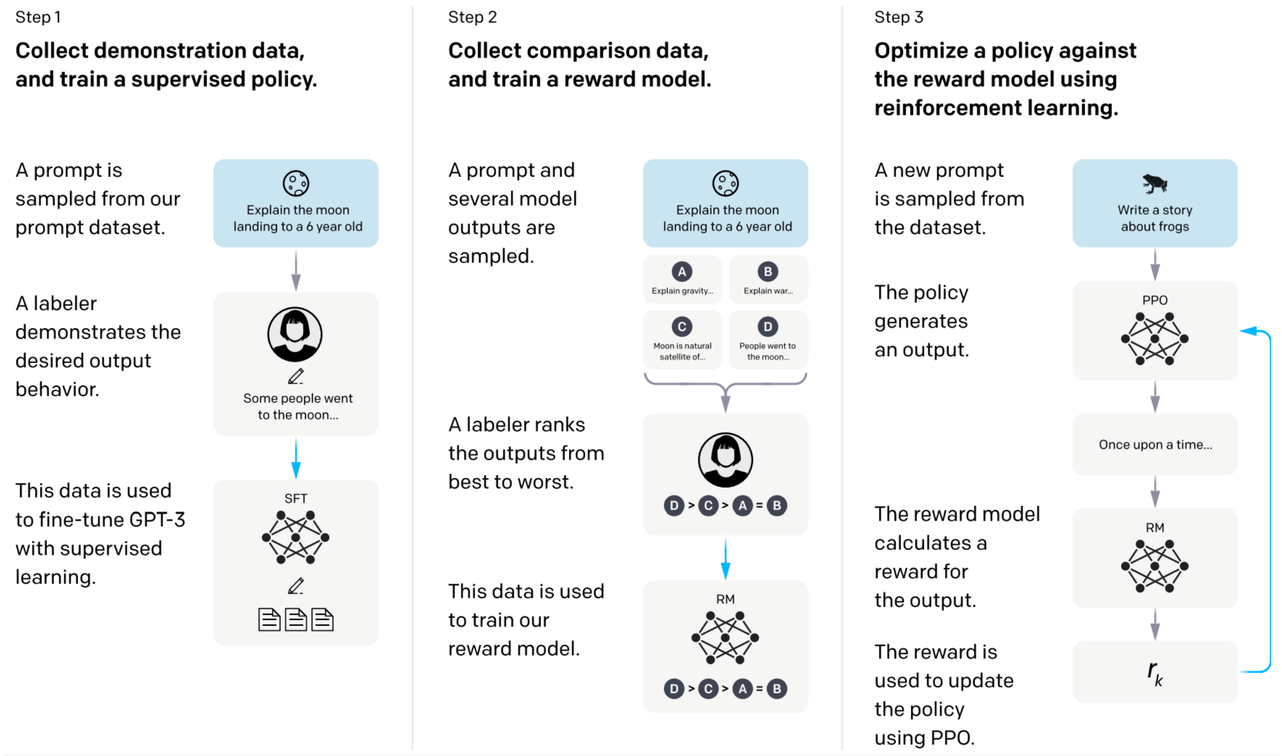
2. 人类强化学习反馈：人类评分反馈.

Training language models to follow instructions with human feedback

ChatGPT 人类强化学习反馈

RLHF 模型原理

三个步骤：（1）预训练一个**语言模型**（2）收集比较并训练**奖励模型**（3）基于奖励模型用强化学习的方式微调**语言模型**。



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

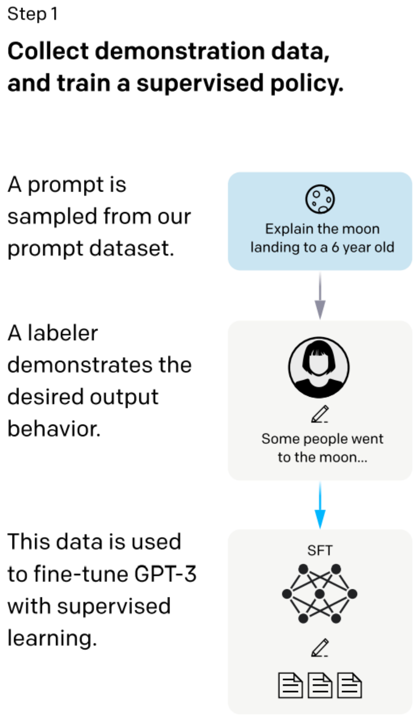
The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

ChatGPT 人类强化学习反馈

RLHF 模型原理

（1）有监督地微调语言模型



Step 1

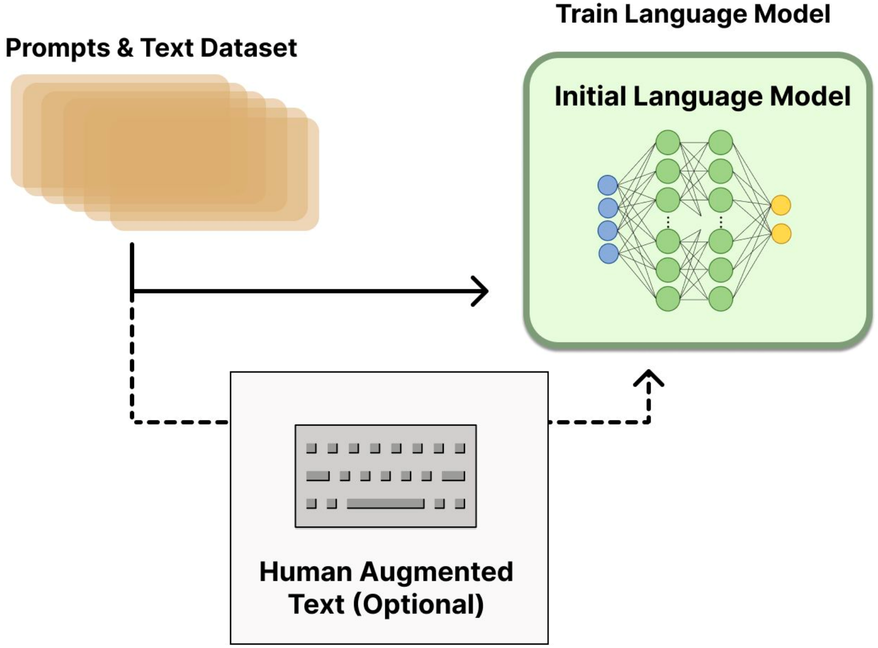
Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

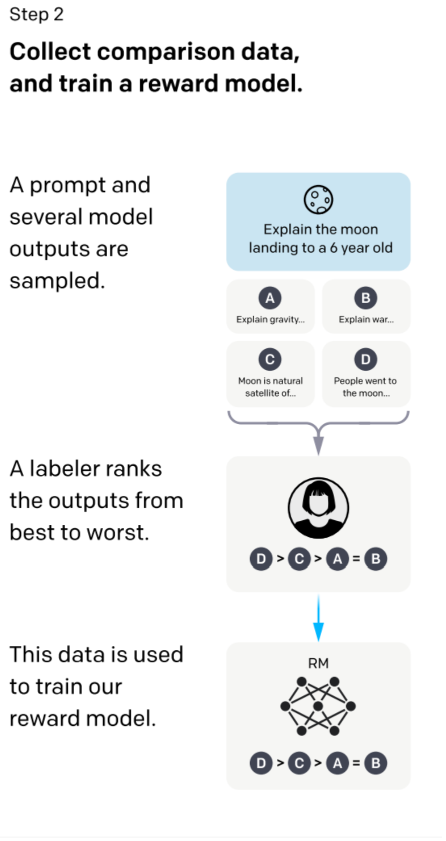
Explain the moon landing to a 6 year old
Some people went to the moon...
SFT



ChatGPT 人类强化学习反馈

RLHF 模型原理

（2）训练奖励模型



Step 2

Collect comparison data, and train a reward model.

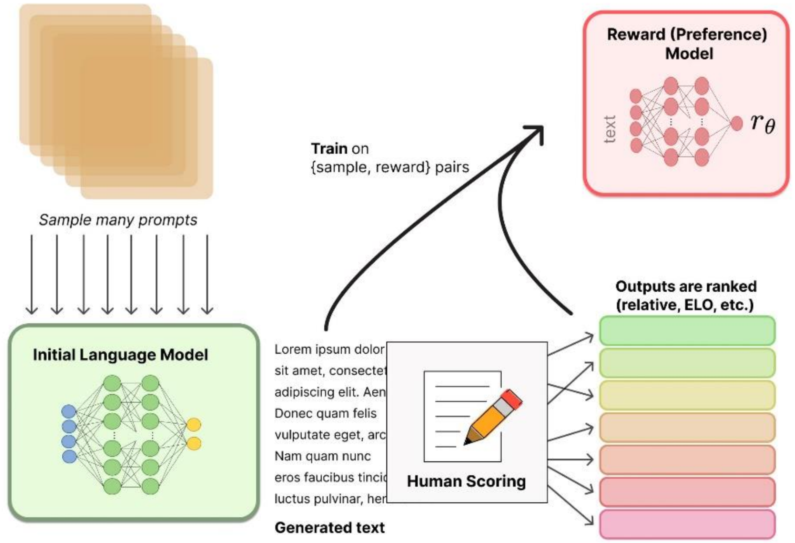
A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B

$$\operatorname{loss}(\theta)=-\frac{1}{\binom{K}{2}}E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x,y_w\right)-r_\theta\left(x,y_l\right)\right)\right)\right]$$

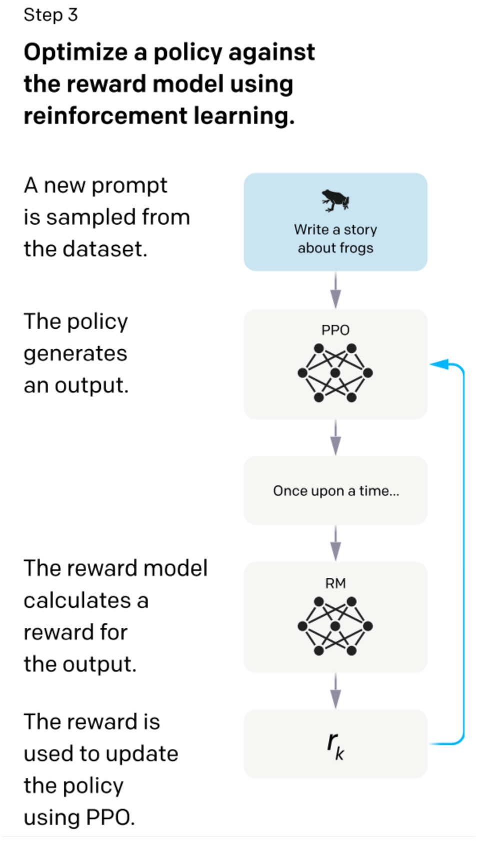


ChatGPT 人类强化学习反馈

RLHF 模型原理

（3）优化强化学习策略模型

$$\text{objective}\,(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]$$



Step 3

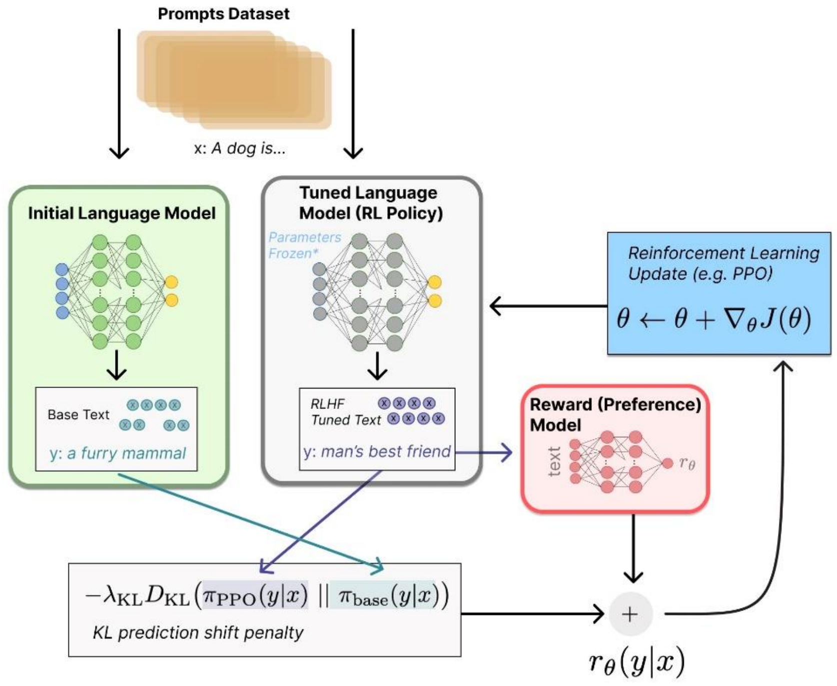
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Prompts Dataset
x: A dog is...
Initial Language Model
Tuned Language Model (RL Policy)
Parameters Frozen*
Base Text
y: a furry mammal
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_{\mathrm{PPO}}(y|x) \,||\, \pi_{\mathrm{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

本讲参考文献

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning:　An Introduction. (Second edition, in progress，draft).
2. David Silver，Slides@　《Reinforcement Learning:An Introduction》,2016.
3. SeqGAN. arXiv:1609.05473v1, 2016.
4. Learning Structured Representation for Text Classification via RL (Zhang et al. AAAI 2018).
5. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning(Zhu, et al. 2016 arXiv:1609.05143v1)
6. David Silver，et al. Mastering the Game of Go without Human Knowledge.(AlphaGo Zero, 2017)