附 录

1 方法

1.1 数据预处理

本方法基于 Drain 算法对日志进行解析，通过滑动窗口方法对日志进行分组。解析时，首先构造正则表达式对原始日志数据进行预处理，然后利用固定深度树进行解析，提取出时间戳和模板信息。日志解析完成后，根据日志的生成时间对日志事件进行排序，随后通过滑动窗口方法对日志进行分组，从而形成日志序列。假设滑动窗口的大小为 m，步长为 1，从第一个日志序列起，每次滑动都会生成一个新的日志序列。序列可以表示为 $L=\{e_1,e_2,\ldots,e_m\}$，其中 m 即为日志窗口大小，e_i 表示第 i 个日志事件。分组完成后，即可构建一个仅含正常日志序列的数据集 $D=\{L_1,L_2,\ldots,L_N\}$，其中 L_i 即表示第 i 个正常日志序列。

1.2 马尔可夫决策过程证明

t 时刻的状态可以表示为：

$$s_t=[k_{t-m+1},k_{t-m+2},\ldots,k_t], \quad (1)$$

s_{t+1} 时刻的状态可表示为：

$$s_{t+1}=[k_{t-m+2},k_{t-m+3},\ldots,k_{t+1}], \quad (2)$$

可以得到下一个时间步事件的转移概率：

$$\begin{aligned}P(k_{t+1}|s_t,a_t)&=P\left(k_{t+1}|k_{t-m+1},\ldots,k_t,a_t\right)\\&=P(k_{t+1}|k_0,a_0,\ldots,k_t,a_t),\end{aligned} \quad (3)$$

严格满足马尔可夫性质：

$$P(s_{t+1}|s_t,a_t)=P(s_{t+1}|s_0,a_0,\ldots,s_t,a_t). \quad (4)$$

1.3 DDQN 算法原理

DDQN 算法在策略 π 下 Q 值的定义可表示为：

$$Q_\pi(s,a)=E[R_1+\gamma R_2+\ldots|S_0=s,A_0=a,\pi], \quad (5)$$

其中折扣系数 $\gamma\in[0,1]$ 用于衡量即时奖励和延迟奖励的重要性。最优动作价值函数的定义如下:

$$Q_*(s,a)=max_\pi Q_\pi(s,a). \quad (6)$$

最优动作价值函数的估计可以通过 Q-learning 算法来学习。但是 Q-learning 算法受限于状态空间的规模，难以直接应用于高维状态空间问题。为了解决这一个问题，设计了一个参数化的动作值函数$Q(s,a;\theta_t)$，并通过标准 Q-learning 的学习方法更新参数。迭代公式如下：

$$\theta_{t+1}=\theta_t+\alpha(Y_t^Q-Q(S_t,A_t;\theta_t))\nabla_{\theta_t}Q(S_t,A_t;\theta_t), \quad (7)$$

其中 α 为学习率，目标 Q 值的定义如下：

$$Y_t^Q=R_{t+1}+\gamma max_aQ(S_{t+1},a;\theta_t). \quad (8)$$

在这一基础上提出 DQN 算法。该算法的关键点包括使用目标网络和使用经验回放机制。目标网络的参数 θ^- 与主 Q 网络的结构相同，只是在每 τ 个步骤中复制一次主 Q 网络的参数，在其余步骤中保持不变。此时 DQN 算法中的目标 Q 值定义如下：

$$Y_t^Q=R_{t+1}+\gamma max_aQ(S_{t+1},a;\theta_t^-). \quad (9)$$

无论是 Q-learning 方法还是 DQN 算法，都使用凭借主 Q 网络中的 Q 值来选择动作和评估动作价值，这种处理方式容易出现过度的价值估计。为了解决过度估计的问题，选择使用 Double Q-learning 方法，该方法有两个参数 θ 和 θ'，在每次更新时，使用一组参数执行贪婪策略选择动作，同时使用另一组参数计算对应动作的 Q 的估计值。目标值定义为:

$$\begin{aligned}Y_t^Q&=R_{t+1}\\&\quad+\gamma Q(S_{t+1},argmax_aQ(S_{t+1},a;\theta_t);\theta_t),\end{aligned} \quad (10)$$

对应的 DDQN 网络的误差即可定义为:

$$\begin{aligned}Y_t^{DoubleQ}&=R_{t+1}\\&\quad+\gamma Q(S_{t+1},argmax_aQ(S_{t+1},a;\theta_t);\theta_t'),\end{aligned} \quad (11)$$

DDQN 网络的损失函数即可定义为：

$$\mathcal{L}(\theta)=\mathrm{E}\left[\left(\begin{matrix}r_{t+1}\\+\gamma Q(s_{t+1},argmax_aQ(s_{t+1},a;\theta);\theta^-)\\-Q(s_t,a_t;\theta)\end{matrix}\right)\right]^2, \quad (12)$$

其中 r 为即时奖励，θ 为主 Q 网络的参数，θ^-为目标 Q 网络的参数，s 为当前状态，a 为当前状态下选择的动作，s_{t+1}为当前状态下采取动作 a_t 所能到达的新状态，a 表示在新状态 s' 下所能采取的动作，r 是环境的即时反馈，γ 为折扣系数。

1.4 Dueling DQN 算法的原理和优势

在 MDP 中，策略 π 的状态价值函数的定义如下:

$$\begin{aligned}v_{\pi(s)}&=\mathrm{E}_\pi[G_t|S_t=s]\\&=\mathrm{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^kR_{t+k+1}|S_t=s\right],\end{aligned} \quad (13)$$

其中 π 是智能体遵循的策略，策略 π 指的是从状态到

动作概率分布的映射，$\pi(a|s)$ 表示的是在当前状态 s 下，选择动作 a 的概率。

为了表征在某个状态下采取某个动作的好坏，MDP 定义了策略 π 的动作价值函数 $q_\pi(s,a)$：

$$q_\pi(s,a) = \mathrm{E}_\pi[G_t|S_t = s, A_t = a] = \mathrm{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right]. \quad (14)$$

在这两个价值函数的基础上，Dueling DQN 定义了一个新的量，优势函数 $A_\pi(s,a)$：

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s), \quad (15)$$

根据状态价值和动作价值函数的定义，可以得出:

$$\mathrm{E}_{a\sim\pi(s)}[A_\pi(s,a)] = 0, \quad (16)$$

状态价值函数 $V_\pi(s)$ 衡量了某一状态的整体优劣，动作价值函数 $Q_\pi(s,a)$ 则表明了在某个状态下选择特定动作的优劣程度。优势函数通过从动作价值函数中减去状态价值函数，量化了每个动作在该状态下的相对重要性。在优势函数的基础上，Dueling DQN 对 Q 网络进行了重新设计。与 DQN 和 DDQN 中的 Q 网络类似，Dueling DQN 中的 Q 网络同样由多个卷积层构成。其独特之处在于，Dueling DQN 的 Q 网络输出层采用了两个全连接层，分别用于表示状态价值流和优势流。这使得两个全连接层可以独立地估计状态值和优势值，然后对两个全连接层的输出进行聚合，形成一个单独的 Q 值。普通的聚合公式如下：

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha), \quad (17)$$

该公式缺乏可识别性，且在实际使用时性能较差，在此基础上提出了两种新的聚合方式：

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha) - \max_{a'\in|A|}A(s,a';\theta,\alpha), \quad (18)$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'}A(s,a';\theta,\alpha), \quad (19)$$

式(18)中的价值流在真正意义上估计了价值函数，优势流也精确地反映了对优势函数的估计，但实际应用不稳定。因此可以用式(19)来代替式(18)，使用平均值替代最大值运算符，可以在保证模型效果的前提下，增强聚合模块的稳定性。Dueling DQN 算法主要优化了 Q 值的生成过程，具体的训练流程与 DDQN 算法基本一致，因此 Dueling DQN 网络的损失函数定义如下：

$$\mathcal{L}(\theta) = \mathrm{E}\left[\left(\begin{matrix} r_{t+1} \\ +\gamma Q(s_{t+1}, argmax_a Q(s_{t+1},a;\theta);\theta^-) \\ -Q(s_t,a_t;\theta)\end{matrix}\right)\right]^2, \quad (20)$$

其中 r 为即时奖励，θ_i 为主 Q 网络参数，为目标 Q 网络参数，s 为当前状态，a 为当前状态下选择的动作，s' 为当前状态下采取动作 a 所到达的新状态，a'表示在新状态 s' 下所能采取的动作，r 是环境的即时反馈，γ 为折扣系数。

2 实验

2.1 数据集

HDFS 数据集取自 Hadoop 分布式文件系统，共有数据 11,175,629 条，根据 *block_id* 将日志划分为不同的轨迹并对轨迹打上标签后，可得到 558221 条正常日志序列，16838 条异常日志序列。在训练阶段，随机使用 80%的正常日志序列进行训练，将剩下的数据均用于测试。

OpenStack 数据集：OpenStack 数据集共有数据 207820 条，根据 *Instance* 将日志划分为不同的轨迹并对轨迹打上标签后，可得到 1497 条正常日志序列，18434 条异常日志数量，198 条异常日志序列。在训练阶段，随机使用 80%的正常日志序列进行训练，将剩下的数据均用于测试。

2.2 实验设置

实验环境参数如表 1 所列。经过大量的参数调整实验，确定了 DQNCG 方法和 DQNAF 方法中的核心参数。其中，学习率初始化为 0.0001，从缓冲池中的采样数量 batch_size = 64。对于输入层，日志序列的输入维度即窗口大小设定为 7，输出维度设定为 128。隐藏层的输入维度为 128，输出维度为 256。输出层的输入维度为 256，输出维度为 29。激活函数选择 ReLU 函数，Top-k 候选集的大小分别为 k = 10 和 k = 6，实验迭代次数分别为 Episodes = 100 和 Episodes = 150，每次迭代过程中执行步数分别为 Step = 300 和 Step = 500。目标 Q 网络每迭代两次更新一次参数，即 K = 2。先验缓冲池存储样本的数量为 10000，探索缓冲池大小存储样本的数量为 100000，折扣系数设置为 0.99。

表 1 实验环境参数

Table 1 Experimental environment parameters

硬件参数		软件参数	
属性	参数	**属性**	参数
显卡	NVIDIA GeForce RTX 3060 Laptop GPU	**操作系统**	Windows11
CPU	AMD Ryzen 7 5800H with Radeon Graphics	**CUDA**	11.3
内存	16GB	**PyTorch**	1.12.1

2.3 评估指标

精确率：精确率描述的是被正确预测为正类的样本与所有被预测为正类样本的比例，反映了模型预测正类的准确性，精确率越高，模型标记正类的准度越高。精确率的定义如下：

$$Precision = \frac{TP}{TP + FP}. \tag{21}$$

召回率：召回率描述的是被正确预测为正类的样本与实际样本的比例，反映的是模型对正类样本的覆盖能力。召回率的定义如下：

$$Recall = \frac{TP}{TP + FN}. \tag{22}$$

F1 分数：F1 分数综合考虑了精确率和召回率的平衡。定义如下：

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{23}$$

准确率：准确率描述的是所有预测正确的样本占总预测样本数量的比例，反映了模型预测的准确性。准确率的定义如下:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{24}$$

特异度：特异度描述的是预测正确的负样本在全部负样本中所占的比例，反映了模型在预测负样本上的可靠性。特异度的定义如下：

$$TNR = \frac{TN}{FP + TN}. \tag{25}$$

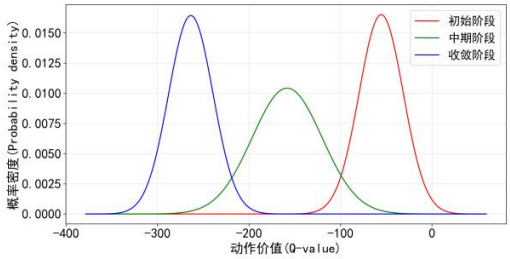
其中 TP（True Positives）表示异常序列被识别为异常序列的数量，FP（False Positives）表示正常序列被识别为异常序列的数量，FN（False Negatives）表示异常序列被识别为正常序列的数量，TN（True Negatives）表示正常序列被识别为正常序列的数量。

2.4 DQNCG 消融实验

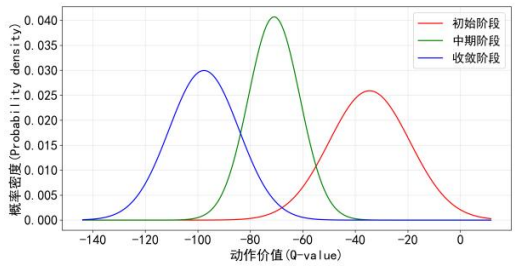
为验证引入先验缓冲池和设计基于先验动作的奖励方式先验动作这两种机制的有效性，将通过消融实验进一步对两种机制进行评估。首先在保持其他部分不变的情况下，移除先验缓冲池，探究先验缓冲池对模型性能的影响。随后，在相同条件下调整奖励方式，去掉奖励函数中的 R_3 部分，探究不同的奖励方式对模型性能的影响

为了说明两种机制能够有效地缓解奖励稀疏的问题，本小节比较了四种方式下 Q 值的分布情况。Q 值的分布情况结果如图 1 所示。

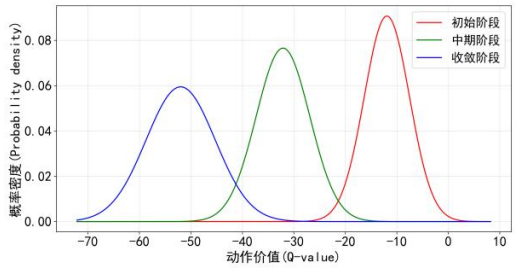
图 1(a)记录了未改进 DDQN 算法的 Q 值分布情况，当缺乏先验知识引导时，算法在初始阶段的 Q 值估计即呈现显著偏低的态势，且随着训练过程的推进 Q 值呈现出大幅度下降的趋势。这一现象源于状态空间的高维度特性与奖励信号的稀疏性，二者共同导致智能体在探索过程中难以有效发现并积累具有正向反馈的经验样本。图 1(b)表明，先验轨迹的引入使智能体展现出积极效应——与原始方法相比，改进后的模型在初始阶段和收敛阶段均保持了更高的 Q 值水平，这表明先验轨迹在一定程度上缓解了无引导学习场景下 Q 值估计偏差的问题。图 1(c)说明，引入先验动作后，智能体在初始阶段对 Q 值的估计获得了显著提升，并且在不同学习阶段间表现出更为稳定的波动特性。在收敛阶段，Q 值分布的均值能够维持在相对较高的水平，这表明先验动作的引入有效减缓了先验信息随探索轨迹增加而衰减的趋势。图 1(d)反映出 DQNCG 在探索过程中的稳定性，从 Q 值分布特性来看，该算法在初始阶段即展现出最高的 Q 值估计水平，且在不同学习阶段中各阶段的 Q 值波动幅度最小。这种在全阶段范围内的优异表现，充分验证了 DQNCG 算法有效缓解了日志序列图构建任务中奖励稀疏的问题。



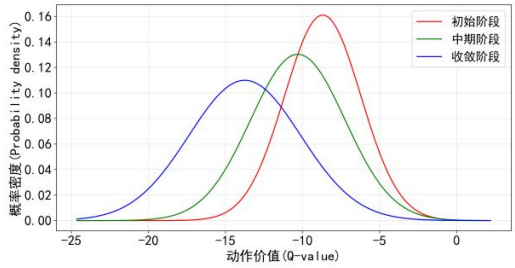
(a) DQNCG Mask Both Q Value



(b) DQNCG Mask Prior Action Q Value



(c) DQNCG Mask Prior Pool Q Value



(d) DQNCG Q Value

图 1 Q 值分布情况

Fig.1 Distribution of Q values

为了说明两种机制对模型性能以及收敛速度的影响，本小节还比较了不同迭代次数下四种方法的 F1-Score，如图 2 所示。在迭代 300 次时，DQNCG 已经能够取得较好的效果，随着迭代次数的增加，模型的性能保持稳定，F1-Score 并未发生大的变化。而 DQNCG Mask Prior Action 和 DQNCG Mask Prior Pool 两种方法随着迭代次数的进一步增加，模型的性能持续提高。前者提升较慢的原因在于先验缓冲池的引入仅提供了有效的轨迹，轨迹中涉及到的状态或动作有限，且容易陷入局部最优。后者仅仅提供了先验动作指导智能体进行有效的探索，随着探索次数的增加，智能体对环境的认知不断提升，逐渐学习到有价值的动作和状态，不容易陷入局部最优。未引入两种机制 DDQN 算法的 F1-Score 指标始终处于较低区间，且在整个训练周期内，该指标几乎未出现明显波动。这一现象表明，未改进的 DDQN 算法在面对奖励稀疏问题时存在缺陷，难以有效收敛，无法构建准确、稳定的策略模型。

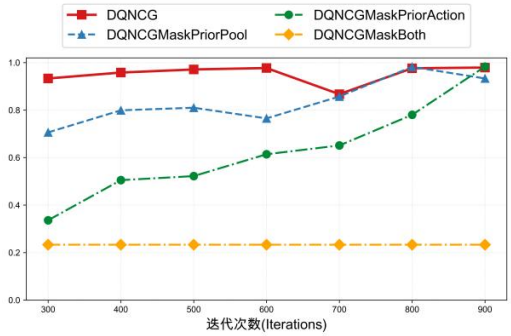


图 2 不同迭代次数下 F1-Score 分数变化

Fig.2 Changes in F1-Score under different numbers of iterations

消融实验结果如表 2 所列。引入先验缓冲池和先验动作奖励后，智能体可以直接使用缓冲池中的专家数据进行前期训练，并选择价值较为明显的动作，减少了无价值的探索，有效缓解了因为状态空间过大引起的探索时间过长和奖励稀疏的问题。

表 2 消融实验结果

Table 2 Ablation experiment results

	Precision	Recall	F1-score	Time
DQNCG Mask Prior Action	0.687	0.904	0.781	1939s
DQNCG Mask Prior Pool	0.930	0.976	0.952	3130s
DQNCG	0.968	0.997	0.982	1501s

2.5 DQNAF 消融实验

本节将在 HDFS 数据集上讨论先验动作、轨迹生成方式和奖励方式对模型的影响。

2.5.1 先验动作的有效性分析

异常序列生成器在探索环境时会与正常序列生成器进行对抗。在同一状态下，异常序列生成器选择的动作与正常序列生成器生成的动作不同才给予一定的奖励。在这种奖励模式下，异常序列生成器在探索过程中会倾向于选择正常序列生成器未生成的动作。正常序列生成器生成集合之外的动作并非全部具有价值。因为部分潜在异常可以通过已存在的事件进行判断，而部分异常则无法基于现有事件作出判断。在 HDFS 数据集的异常集合中，潜在异常的占比最高，

达到 85.5%，该类异常可通过已发生的事件进行判断；未知异常的占比为 13.8%，这种异常无法基于现有事件作出判定。但需注意的是，未知异常中很大一部分可以通过已发生的事件来判断，真正完全未知的异常仅占 0.7%。在日志解析环节，理论上，动作空间涵盖了所有相关日志模板所对应的动作。然而，鉴于系统日志数据处于持续更新的状态，当选取特定时间段的数据用于模型训练时，训练数据集中的日志序列，可能无法涵盖全部日志模板。这就意味着，部分日志事件不会在正常序列生成器的输出结果中出现。在模型探索阶段，由于奖励的设置，那些在训练数据中未曾出现的动作，反而可能被优先选中。这极有可能致使模型对系统环境产生错误认知，不仅干扰模型学习系统正常运行模式，还可能在异常检测任务中，引发误判，降低模型整体的可靠性与准确性。

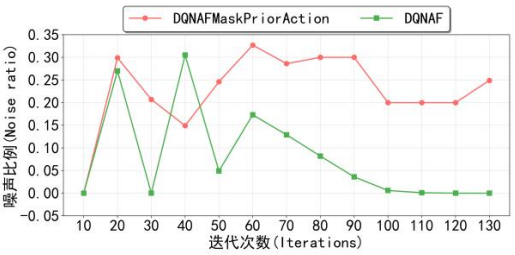
为了减少未知动作和未知异常的影响，在训练过程中，DQNAF 也引入了先验动作集合，利用先验动作集合来辅助评估动作的价值。两种方法在混淆矩阵上的指标如表 3 所列，可以发现二者在五种指标上有一定的差距。

表 3 先验动作有效性分析实验结果

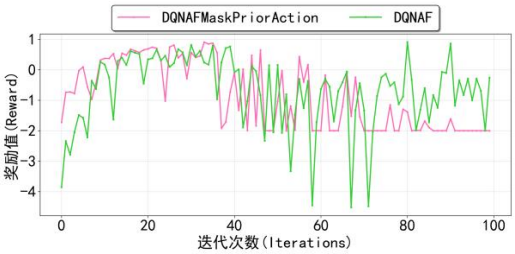
Table 3 Experimental results on the effectiveness analysis of prior actions

Method	Precision	Recall	F1-Score	TNR	ACC
DQNAF Mask Prior Action	0.905	0.940	0.922	0.991	0.986
DQNAF	0.970	0.978	0.974	0.996	0.993

噪声比例及奖励收敛曲线如图 3 所示。由图 3(a)可以看出，加入训练集中的先验动作后，预测集中噪声的比例显著降低。即使随着迭代次数的增加，噪声比例存在一定波动，但始终维持在一个较低的水平。在未加入先验动作的方法中，预测集中的噪声比例相对较高。尽管这一比例会随着迭代次数的变化而出现起伏，但其整体水平仍远高于前者。图 3(b)反映了两种下奖励的收敛情况，随着迭代次数的增加，未加入先验动作的方法中奖励收敛于-2，陷入了局部最优。这主要是因为没有加入先验动作，导致智能体在选择动作时更倾向于未知的动作，而这些动作对应的奖励为-2。



(a) 噪声比例变化结果

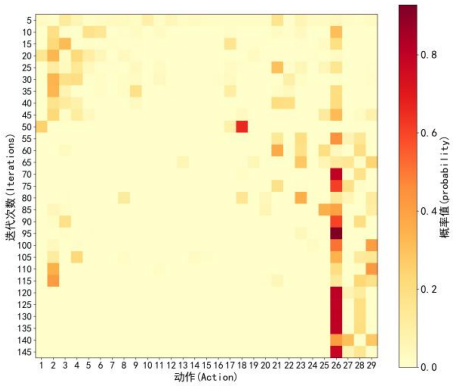


(b) 奖励收敛情况

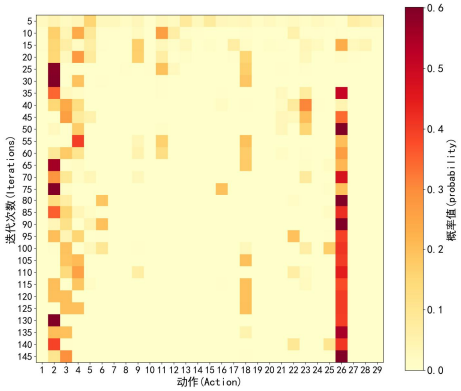
图 3 先验动作有效性分析曲线

Fig.3 Curve of effectiveness analysis of prior actions

为直观说明引入先验动作对智能体决策的影响，本节对智能体在迭代过程中选择动作的情况进行了统计，计算了各动作被选择的概率，热力图由图 4 所示。图 4(c)中，红色格子所对应的序号表示未曾出现过的日志事件，即本节定义的非先验动作。由这些动作生成的日志序列被归类为未知异常。从热力图 4(a)可以看出，在未引入先验动作时，随着迭代过程的增加，非先验动作被选择的概率逐渐提升，并最终趋于稳定。这导致预测集合中出现过多的未知异常，同时降低已知潜在异常的检测概率，从而增加模型漏报和误报的可能性。引入先验动作的 DQNAF 方法在训练过程中未受到非先验动作的干扰，随着迭代次数增加，智能体更倾向于选择能够生成已知潜在异常的动作。



(a) DQNAF Mask Prior Action Heat Map



(b) DQNAF Heat Map

(c) Event

图 4 动作选择分析

Fig.4 Action selection analysis

2.5.2 轨迹生成方式的性能分析

在异常序列生成器的训练过程中，由于引入了正常序列生成器，两个智能体均可对环境施加影响，进而生成新的状态。而基于 Q-learning 理论构建的 Dueling DQN 网络，行为策略与目标策略相互独立，实现了有效解耦。这一特性为优化网络性能提供了理论基础。一般算法流程中，鉴于异常序列生成器是需要训练的目标模型，因此由其选定的动作来驱动新状态的产生。在 DQNAF 中，异常序列生成器所给出的动作往往更倾向于异常工作流。对正常日志序列训练的正常序列生成器而言，在许多异常状态下，它只能提供相对正确的选择。这种特性一定程度上会影响两个智能体之间的对抗效果。如果由正常序列生成器来影响环境，由于正常序列生成器已经趋于使用经验而非探索，产生的轨迹及对应的状态数量有限，不利于智能体探索。为验证异常序列生成器驱动新状态产生的有效性，本节通过对比实验比较了两种轨迹生成方式对模型性能的影响。两种轨迹生成方式对模型性能的影响如图 5 所示。

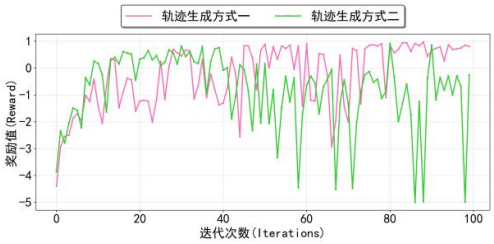
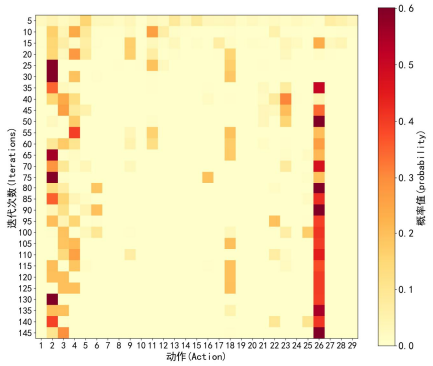


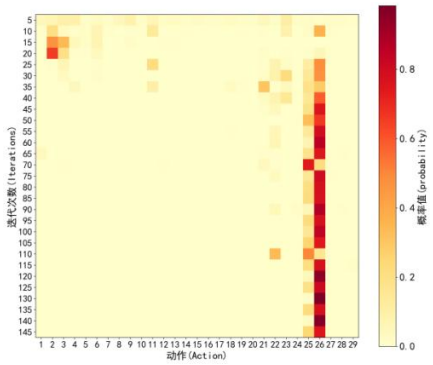
图 5 不同轨迹生成方式下奖励的变化情况

Fig.5 The variation of rewards under different trajectory generation methods

图中轨迹生成方式一、二分别为由异常序列生成器和由正常序列生成器产生的轨迹。由图可知，随着迭代次数的增加，方式一能够长时间维持较高的奖励水平；而方式二虽然在初期可获得较高的奖励，但随后奖励一直保持较低水平，并且波动较大。



(a) Trace1 Heat Map



(b) Trace2 Heat Map

图 6 不同轨迹生成方式下的动作选择分析

Fig.6 Analysis of action selection under different trajectory generation methods

图 6 也反映了这一情况。由图 6(b)可以看出，在迭代初期，其他动作还存在着被选择的概率，随着迭代次数的增加，其他动作被选择的概率逐渐降低为 0。这表明异常序列生成器在初期还会选择其他的动作，随着训练的进行，预测模型几乎全部选择动作 25，26，陷入了局部最优。

两种方法在评价指标上的差异如表 4 所列。其中，轨迹生成方式二的各项指标均表现较低。因为方式二生成的轨迹偏向于正常序列，这导致智能体在探索过程中未能充分获取正常序列状态之外的其他有价值的状态。这种局限性影响了智能体对环境的探索与学习能力，进而导致其对各状态价值的估计出现偏差，最终造成了奖励值和评价指标偏低的结果。

表 4 不同轨迹产生方式实验结果

Table 4 Experimental results of different trajectory generation methods

Method	Precision	Recall	F1-Score	TNR	ACC
轨迹生成方式一	0.705	0.997	0.826	0.936	0.944
轨迹生成方式二	0.230	0.987	0.373	0.497	0.562

2.5.3 轨迹生成方式的性能分析

在 DQNAF 方法中，由于需要两个智能体进行对抗，其具体的对抗反馈需通过奖励函数来体现。为了直观地反映对抗特性，本节设计了一个简洁的奖励函数，其定义如下：

$$r(s,a,s') = \begin{cases} r_1, & \text{if } a \in A_{est} \\ r_2, & \text{if } a \notin A_{est} \end{cases}, \tag{26}$$

预测模型产生的动作如果处于日志序列图生成模型估计的动作集合中，则给予一个较大的惩罚 r_1，否则给与一个较大的奖励 r_2，该奖励函数仅考虑了两个智能体之间的关系，并未考虑更多环境中的因素，该奖励函数为方式一。本文设计了一种分层奖励函数，该方法考虑到智能体之间的对抗关系，并且引入了先验动作集合，该奖励函数为方式二。

考虑到实际软件系统中的故障并非总是突然发生，有些故障可能是在执行一定次数的正常操作后才逐渐显现出来的，因此本节设计了一种线性奖励函数，即方式三。该奖励函数的定义如下：

$$r(s) = \alpha R_{action}(a) + \beta R_{step}, \tag{27}$$

其中 α 为状态奖励的权重，β 为时间成本的权重，$R_{action}(s)$和$R_{step}(s)$定义如下：

$$R_{action}(a) = \begin{cases} 1 & \text{if } a \in A_{est} \\ 0 & \text{if } a \notin A_{est} \end{cases}, \tag{28}$$

$$R_{step}(s) = stepCount, \tag{29}$$

其中，$stepCount$ 表示智能体到达该状态时所使用的步数。为了估计智能体探索正常操作后产生的异常，本节设置 $\beta = 0.01$。但在构建马尔可夫决策过程时，并没有明确指定终止状态，线性奖励中也没有负反馈。由于奖励机制对步数限制的惩罚力度不足，智能体可能倾向于延长探索步数以优化累积奖励，进而导致异常预测结果出现滞后性。

本节设计了一种综合线性奖励和分层奖励的奖励函数，即方式四。该函数定义如下：

$$r(s,a,s') = \begin{cases} r_1, \text{if } stepCount = MAX\_STEP \& a \in A_{est} \\ r_2, \text{if } stepCount = MAX\_STEP \& a \notin A_{est}, \\ r_3, \text{if } stepCount > MAX\_STEP \\ r_4, \text{if } stepCount < MAX\_STEP \end{cases} \tag{30}$$

其中 *MAX_STEP* 为最大允许步数阈值，若智能体单次轨迹探索步数 $t > MAX\_STEP$，则施加高强度负奖励 $r_3 = -100$；对于规定步数内未找到异常的行为给与一定的惩罚 $r_1 = -1$，并对规定步数内寻找到异常的行为给予正向激励 $r_2 = 1$，对于 r_4 定义如下:

$$r_4 = R'_{action}(a) + \beta R_{step}, \tag{31}$$

$$R'_{action}(a) = \begin{cases} 0, & \text{if } a \in A_{est} \\ 0.5, & \text{if } a \notin A_{est} \end{cases}. \tag{32}$$

为平衡深度探索与实时检测需求，需约束智能体在逐步搜索潜在异常的过程中同步执行即时状态评估，进而确保对当前序列的异常特征保持持续敏感性，因此重新定义了 R'_{action}。

为了比较上述几种奖励方式对模型性能的影响，本节在保证其他参数相同的条件下进行了实验，并给出了相应的实验结果如表 5 所列。

表 5 不同奖励方式下实验结果

Table 5 Experimental results under different reward methods

Method	Precision	Recall	F1-Score	TNR	ACC
奖励方式一	0.990	0.590	0.739	0.999	0.945
奖励方式二	0.937	0.823	0.876	0.994	0.979
奖励方式三	0.251	0.987	0.400	0.552	0.611
奖励方式四	0.705	0.997	0.826	0.936	0.944

实验结果表明，奖励方式二和奖励方式四的 F1-Score 值较高，模型的性能相对较好。两者的区别在于，奖励方式二的召回率相对较低，这是因为这种奖励方式忽略了隐藏在正常日志流中的异常序列；而奖励方式四的召回率达到了 99.7%，对异常模式的覆盖范围非常广，因此适合用于作为异常序列生成器的奖励机制。虽然奖励方式三在召回率上表现优秀，但其精确率和特异度较差，预测结果中正常序列出现的概率较高，容易导致误报。

2.6 DQNCG 参数分析

影响模型性能的参数主要包括奖励值的大小、滑动窗口大小、预测窗口的大小以及先验动作的比例。奖励值的大小影响着智能体的学习速度、策略的收敛性以及探索和利用的平衡；日志序列的窗口大小关系到上下文信息的捕捉和计算的复杂度，窗口过大或过小都会影响模型的泛化能力；预测窗口的大小关系到有价值的策略和冗余策略的取舍，窗口过大会导致冗余策略的存在，而窗口过小又会遗漏部分有价值的策略，从而对模型性能产生负面影响。先验动作的比例影响着智能体对环境的探索能力，先验动作比例过高，会使智能体缺乏探索能力，易陷入局部最优，比例过低又不能起到引导智能体探索和学习的作用，从而影响模型的训练和性能。

2.6.1 奖励值的大小分析

在智能体的训练过程中，奖励机制是引导智能体学习的核心要素。智能体与环境互动时，环境反馈的奖励，不仅为智能体的学习方向提供指引，更是计算价值函数的重要依据。通过对价值函数的迭代优化，智能体逐步探索并获取最优策略，实现学习目标。在既定的奖励模式下，奖励值的大小设定，直接影响智能体对不同行为的偏好，进而显著左右模型性能。为系统剖析奖励值大小对模型性能产生的影响，本研究确保奖励方式统一，维持其他各项参数恒定，针对不同大小的奖励值展开了多组对比实验。实验过程中，通过计算召回率、精确率与 F1-Score 等关键指标，对模型在不同奖励值设定下的性能表现进行量化评估，实验结果如图 7 所示。

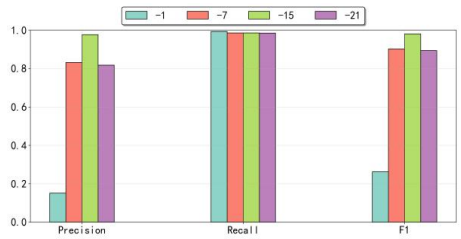


图 7 不同奖励大小的评价指标折线图

Fig.7 Line chart of evaluation metrics for different reward sizes

研究选择了 4 个差距较大的 R_3 的值进行参数分析实验。由实验结果图可以看出，四种奖励方式在 Recall 指标上相差不大，这表明 R_3 的设置对召回率的影响不大，这是因为一定的负反馈足以引导智能体分辨出明显不符合正常模式的序列。但各参数在精确率和 F1-Score 的差距上非常大，$R_3 = -1$ 时精确率不足 0.2，而 $R_3 = -15$ 时精确率非常高。这是因为负反馈的绝对值过小，会影响智能体学习和捕获更复杂的正常模式，使智能体难以有效区分状态和动作的价值，因此出现了不同方式在精确率上的差异。而当 $R_3 = -21$ 时，模型的性能又有所下降，出现这种结果的主要原因在于，奖励值过大，智能体过度优化特定行为，导致了探索不足的问题。

2.6.2 滑动窗口大小分析

在构建日志正常序列图时，获取包含足够上下文信息的日志序列，是决定任务成效的关键。日志序列的窗口大小，作为影响上下文信息获取量的关键变量，深刻左右着模型的整体性能。窗口过大或过小，都可能导致模型难以准确捕捉日志间的内在联系，进而影响模型的表现。为深入探究窗口大小与模型性能之间的关系，本研究在保持其他参数恒定的前提下，针对不同窗口大小展开实验。实验过程中，分别计算召回率、精确率以及 F1-Score 等指标，以此量化评估模型性能，实验结果如图 8 所示。

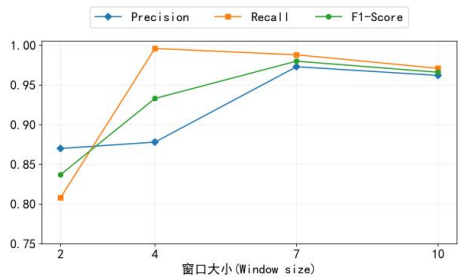


图 8 不同窗口大小的评价指标折线图

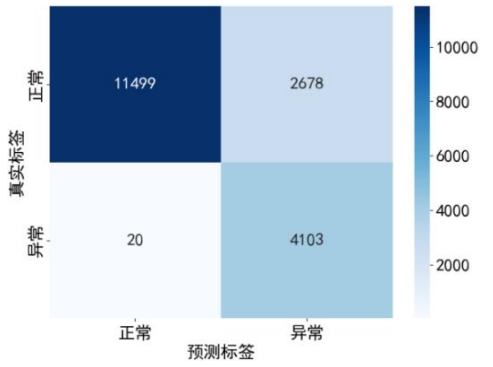
Fig.8 Line chart of evaluation metrics for different window sizes

研究选择了 4 个窗口大小值 {2, 4, 7, 10} 进行参数分析实验，三个评价指标结果如图 2 所示。由图可以看出，当窗口大小为 4 时，取得了较高的召回率，但是此时的精确率非常的低，F1-Score 分数也比较低。这是因为窗口过小，无法捕获足够的上下文信息，一个日志序列所能提取的信息有限，使得智能体难以判断状态的价值。当窗口大小为 7 时，三个评价指标都取得了较高得值，精确率和 F1-Score 达到了顶峰。在窗口大小为 10 时，三个指标相对于窗口大小为 7 时有一定的下降。这是因为窗口过大，引入了噪声干扰模型，导致模型过于拟合噪声或无关模式，降低模型的泛化能力。综合考虑状态空间中状态的数量以及计算复杂度，在 HDFS 数据集上，将窗口大小设定为 7 时，既能有效提取足够的上下文信息，又能确保模型顺利收敛。

2.6.3 预测窗口大小分析

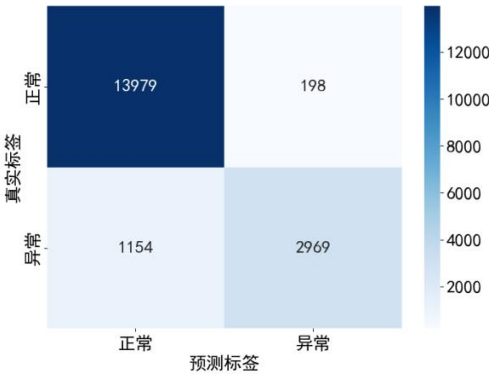
智能体在与环境交互的过程中，通过对价值函数的更新，能够评估各状态在构建正常日志流任务中的价值。在此任务中，特定日志序列在单个时间步内，能够衍生出多种符合规范的正常日志序列，这使得智能体所处环境中充斥着冗余或极为相似的策略。因此，许多动作所对应的 Q 值十分接近，差异微小。在这种情况下，单纯依赖最优策略（即选择 Q 值最大的动作）难以有效应对复杂多变的环境。本小节旨在深入探究可选动作数量对模型性能的影响。为确保实验结果的可靠性与有效性，在实验过程中，严格维持其他参数恒定，仅改变生成序列时选择 Q 值的数量 k，对应的混淆矩阵热力图如图 9 所示。

(a) k=2



(b) k=8

(c) k=10



(d) k=15

图 9 k 值大小分析

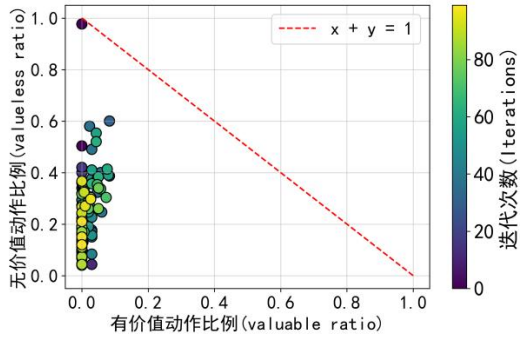
Fig.9 Analysis of k value size

研究选择了 4 个窗口大小值 {2, 8, 10, 15} 进行实验。从图中可以看出，随着 k 值的增大，矩阵左上角的颜色越来越深，模型对负类样本（正常日志）的判断能力越来越强，说明模型生成正常日志序列的能力逐步提升。矩阵右下角的颜色逐渐变浅，表明模型对正类样本（异常序列）的判断能力减弱。这主要是因为 k 值较小，智能体倾向于仅生成安全性极高且出现频率极高的正常序列，而无法生成其他同样可能

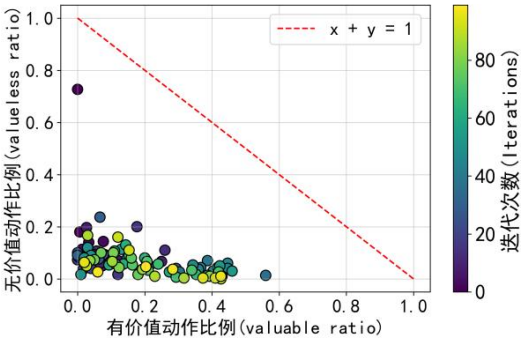
的正常序列。在同一状态下，后续可能存在多种正常事件和正常序列，但过小的窗口会忽略这些其他序列，从而导致在检测过程中，对测试集中正常序列的覆盖程度较低。随着 k 值的增大，智能体生成的正常序列较多，覆盖能力也随之增强。但窗口过大会导致价值较低的非正常动作被视为正常动作，引入了过多的噪声，增大异常样本被误判为正常样本的概率。

2.6.4 先验动作比例分析

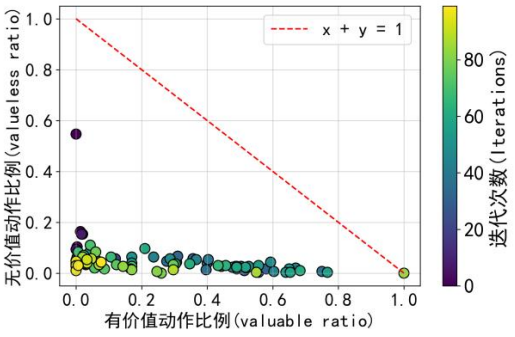
本文在DQNCG方法中设计了一种基于先验动作的奖励方式。该方式中先根据先验动作出现的频率进行排序，再按照一定的比例筛选出部分先验动作。本节在保证其他参数相同的情况下，对不同的先验动作的比例进行实验，以探究先验动作比例对模型性能的影响。实验结果如图 10 所示。



(a) Prior Action Threshold=0.1



(b) Prior Action Threshold=0.4



(c) Prior Action Threshold=0.7

图 10 先验动作比例分析

Fig.10 Analysis of the proportion of prior actions

图中纵坐标表示无价值的动作在迭代过程中出现的概率，横坐标表示有价值的动作在迭代过程中出现的概率，圆点颜色的深浅表示迭代的次数。本节中将智能体选择的动作分成了三类：无价值动作、有价值动作和低价值动作。无价值动作指的是获得较大负反馈的动作，这类动作不会产生正常的序列；有价值的动作指的是获得正反馈的动作，这类动作能够产生正常的序列；低价值动作指的是仅有微小惩罚的动作，该类动作代表着智能体对训练集之外的正常模式的探索，即是先验动作的子集。虚线表示无价值动作和有价值动作的概率和为 1 的情况，点与虚线的距离越远，说明低价值动作被选择的概率越高。当先验动作的比例较小时，随着迭代次数的增加，有价值动作的概率保持不变，无价值动作的概率在短暂的提升以后又开始下降。点集离虚线较远说明智能体一直在探索训练集之外的正常模式，未能有效学习到环境中明显的正常模式。随着先验动作比例的增大，有价值动作被选择的概率逐渐提升，点也随之逐渐靠近虚线。这说明智能体学习到了环境中明显的正常模型，先验动作在一定程度上缓解了稀疏奖励带来的问题。当先验动作的比例达到 0.7 时，此时已经产生了近乎与虚线重合的点，有价值动作被选择的概率也显著提升。这表明模型已能够准确估计环境中各状态的价值，并具备了模拟正常日志流以生成合理日志序列的能力。

2.7 DQNAF 参数分析

影响模型性能的参数主要包括预测序列的长度、预测窗口的大小和DQNCG的窗口大小预测序列的长度关系到模型的收敛性和预测结果的及时性；预测窗口大小关乎预测结果的准确性和覆盖范围；DQNCG

的窗口大小影响着模型的整体性能。

2.7.1 预测序列的长度分析

在日志异常预测任务中，预测序列的长度是一个重要的参数，它控制模型收敛效率与预测时效性的平衡。为了探究本方法中预测序列的适宜长度，本小节在保证其他参数相同的情况下，对不同的预测序列长度进行实验，实验结果如图 11 所示。

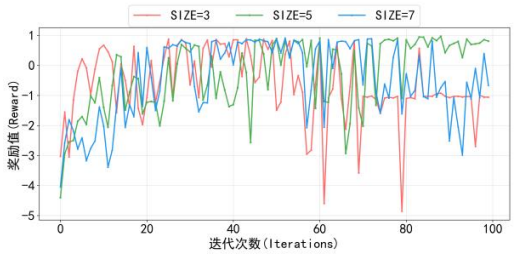


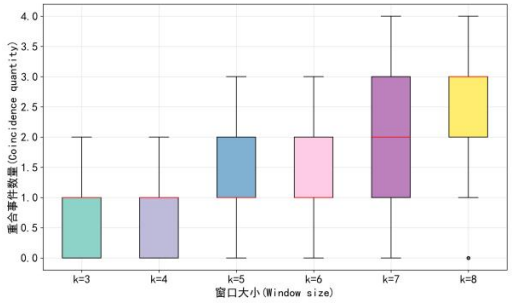
图 11 不同预测长度下奖励的变化情况

Fig.11 The variation of rewards under different prediction lengths

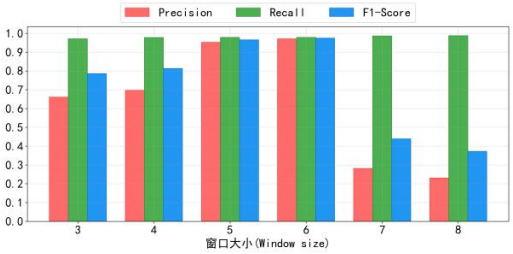
研究选择了三个预测序列的长度{3, 5, 7}进行实验探究。实验结果表明，预测长度等于 3 时，奖励会收敛在一个较低的值附近。奖励收于-1 附近是因为预测序列长度较短，在几个时间步内很难找到隐藏于正常序列下的异常模式。当将序列长度设定为 5 时，模型训练过程中，奖励能够稳定收敛于较高水平。这是因为该序列长度保障了模型训练的收敛性，引导模型参数在反复迭代过程中逐渐稳定，大幅降低模型陷入局部最优解的风险。同时也能够精准捕捉到日志数据里潜藏的异常模式。当序列长度设定为 7 时，模型训练过程中奖励难以顺利收敛。这主要是由于预测序列过长，致使奖励信号呈现稀疏状态。稀疏的奖励反馈，使得模型难以有效捕捉行为与奖励之间的关联，进而严重影响模型的学习效果。通过在 HDFS 数据集上开展的一系列对比实验发现，当预测序列长度为 5 时，DQNAF 方法能取得最佳的效果。

2.7.2 异常预测模型的预测窗口大小分析

在单个时间步的预测中，智能体会基于当前状态，估计下一个时间步可能出现的潜在异常事件。根据各动作的 Q 值排序，选择前 k 个事件作为该时间步的预测结果，其中预测窗口的大小即为 k。智能体在根据当前序列对潜在的异常进行预测时，会生成固定长度的预测序列。预测窗口的大小直接影响预测序列的数量：若预测窗口过大，生成的序列数量过多，不符合实际应用场景，容易导致误报现象；若预测窗口过小，生成的预测序列数量不足，又不能够最大程度地反映潜在的异常。因此，预测窗口大小的选择较为重要。为了探究预测窗口大小对模型性能的影响，在其他参数保持不变的情况下，本小节针对不同预测窗口大小开展了实验，并计算了召回率、精确率和 F1-Score 等指标，实验结果如图 12 所示。



(a) 不同预测窗口大小下的重合事件分布



(b) 不同预测窗口大小在三个评价指标上的结果

图 12 不同 DQNAF 预测窗口大小的实验结果

Fig.12 Experimental results of different DQNAF prediction window sizes

研究选择了 6 个窗口大小值{3, 4, 5, 6, 7, 8}进行参数分析实验。图 12(a)反映了不同预测窗口大小下重合事件的分布情况，随着 k 值的增大，重合事件的数目也随之增多。结合柱状图 12(b)进一步分析可以发现，预测窗口大小的变化并未对模型的召回率产生太大的影响。这说明模型在学习过程中，对于异常信息的提取是可靠的。当预测窗口设置过小时，异常序列生成器和正常序列生成器所预测产生的动作数量同步减少。这一变化致使重合事件集中事件数量锐减，部分本应在重合集中出现的事件，因窗口尺寸的限制未能纳入其中。这种情况致使本应被视作潜在异常前序的序列，错误地作为异常序列被输出。这类序列和正常序列在结构、特征上重合度较高，模型难以精确识别二者差异，在判别过程中，将部分正常序列误判为异常序列。随着这类误判情况不断增多，模型精确

率持续下滑。预测窗口的值超过 6 时，随着预测窗口的增大，精确率也不断降低。一方面因为过大的窗口容易引入更多噪声数据，另一方面是因为预测窗口过大，在探索过程中贪求步数而未找到异常状态的动作，因为 Q 值相对其他动作较高，也出现在了重合集中。这些动作衍生出的潜在异常前序序列，在结构和特征层面与正常序列高度相似。模型在对序列进行判别时，难以有效甄别二者差异，进而将正常序列错误识别为异常序列。随着此类误判情况的增多，模型误判概率显著上升，精确率也随之下降。

2.7.3 正常序列生成器的预测窗口大小分析

在 DQNAF 方法中，异常序列生成器需要与正常序列生成器对抗，从而探索和学习环境中的异常模式。因此，正常序列生成器的表现就非常的重要。生成器对当前状态所生成的预测事件数量需要维持在合理区间：若预测事件数量过少，将导致生成器所能提供的正常行为模式样本不足，限制了预测模型在单次交互中所能获取的有效信息量，进而影响模型对异常模式的学习效果；反之，若预测事件数量过多，一方面可能超出了生成器的建模能力范畴，导致生成结果的可靠性下降，另一方面过量的事件容易引入冗余信息和噪声干扰，使得预测模型在判别过程中面临更高的误判风险。本节在保证其他条件相同的前提下，对不同的 DQNCG 预测的阈值进行了实验，实验结果如图 13 所示。

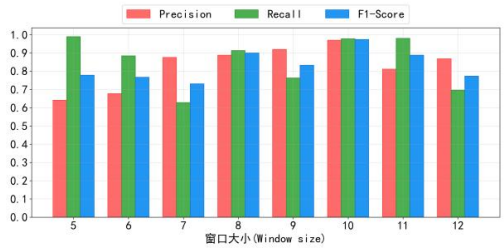


图 13 不同 DQNCG 预测窗口大小的实验结果

Fig.13 Experimental results of different DQNCG prediction window sizes

研究共选择了{5, 6, 7, 8, 9, 10, 11, 12}共 8 个窗口大小进行分析。由图 13 可以看出，DQNCG 的窗口大小为 10 时，模型的各项指标都较高，且 F1-Score 达到了最高。这与 2.6.3 节中的预测窗口大小分析的实验结果相吻合。若预测窗口过小，模型在对抗过程中所能学习到的历史经验信息较为匮乏，难以充分捕捉数据的长期依赖关系；而窗口过大时，模型对事件的估计容易引入冗余信息和噪声干扰，导致异常预测模型的训练过程受到负面影响。

2.8 基线方法

主成分分析法：PCA 方法能够对数据进行降维，经过降维处理的数据能够保留原始数据的特性。在日志异常检测任务中，PCA 方法构建了状态比率向量和事件计数向量，然后通过 PCA 模型对这些向量进行投影。通过计算投影点与中心点之间的距离，可以判断是否存在异常日志。

IM：IM 方法通过挖掘日志数据中的不变量实现日志异常检测。这些不变量反映了系统在运行期间始终存在的线性关系。IM 方法构建事件计数向量，利用事件计数向量来挖掘日志序列中的不变量。在日志异常检测阶段，只需要判断日志序列对应的事件计数向量是否满足不变量。满足所有不变量时，判断为正常，否则视为异常。

LogCluster：LogCluster 方法将日志序列构建为事件计数向量，经过 IDF 和归一化处理后，通过分层聚类分别对正常和异常的事件计数向量进行聚类，得到两组向量簇作为知识库，再计算簇的质心来代表该簇。异常检测阶段，通过计算到来的序列和质心的距离来判断日志序列是否异常。

Deeplog：DeepLog 方法忽略日志事件的数量，关注日志序列中日志事件之间的顺序关系。为了模拟正常事务流，DeepLog 选择了 LSTM 模型来获取日志数据之间的序列关系。该模型能够基于历史日志序列进行预测，输出下一时刻可能发生的日志事件。若预测的日志事件包含真实事件，则认为日志正常，否则视为异常。

其中，PCA、IM 和 LogCluster 都是通过聚类分析日志模式，且不同程度地利用了日志事件的数量关系。模型的输入是待检测的数据的特征向量，经过分析后得到对应数据在新的维度上的特征，并根据新的特征判断日志是否异常，不具备产生下一个时间步的日志事件的能力。DeepLog 方法使用了 LSTM 来捕获日志之间的依赖关系，具备生成下一个或多个时间步日志的能力，但预测的都是正常日志。上述几种方法不具备预测异常序列的能力，而 DQNAF 方法主要用于进行日志异常预测。

2.9 DQN 变体算法对比

为了阐释 DDQN 算法的优势，本节聚焦于不同 DQN 变体，在统一参数设置的前提下，分析它们在训练过程中奖励值的动态变化，呈现它们的性能差异。对比算法主要包括 DQN、DDQN 和 Dueling DQN 方法，三种方法的奖励变化情况如图 14 所示。

由图可知，DQN 方法较 DDQN 方法和 Dueling DQN 方法而言，在奖励收敛的过程中，获得的平均奖励几乎都较低，即使最后奖励收敛，DQN 方法获得的奖励依然低于二者。这是因为 DQN 的单网络架构导致目标值震荡，使其在复杂日志序列决策中难以捕捉细微正常模式；DDQN 的目标网络延迟更新机制和 Double Q 策略，减少了因最大化操作导致的奖励估计偏差；Dueling DQN 的价值函数分解，更高效地学习日志状态的基础价值与动作优势。在表现相同的情况下，由于 Dueling DQN 的网络更为复杂，DDQN 算法的计算开销比 Dueling DQN 更低，更具有应用价值。

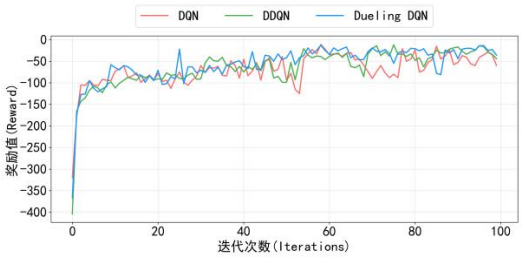


图 14 不同强化学习方法的奖励收敛过程

Fig.14 Reward convergence process of different reinforcement learning methods

2.10 HDFS 与 OpenStack 数据集对比

表 6 描述了在日志滑动窗口尺寸为 7 的情况下异常日志包含的事件序列在训练集与测试集中的重叠关系。从表中的统计数据可以得到以下结论。首先，异常日志所独有的事件序列（即滑动窗口尺寸为 7 的事件序列）无论在总数比例还是种类比例上，数据集 OpenStack （ 0.39%/20.99% ） 都 远 低 于 HDFS（19.21%/61.25%）。这就意味着 OpenStack 中有大量的异常日志所包含的事件序列与正常日志所包含的事件序列是一致的，换句话说，需要提升判断 OpenStack 中的异常日志，仅靠事件序列性是完全不够的。其次，在我们的方法中，训练集仅包含正常日志数据。而在两个数据集的训练集与测试集中，异常日志所包含的事件序列在训练集中出现的总数是 4300。换句话说，OpenStack 在训练的过程中会出现大量的“异常日志事件序列”按照“正常日志事件序列”的标签进行训练，从而导致依赖事件序列进行判断日志异常的方法在性能上有所影响。DQNCG 和 DQNAF 在训练的过程中虽然均依赖事件序列进行异常日志的判断，但是两者在训练方式上有所区别。前者是通过训练集中的正常日志事件序列构建经验缓冲池，然后再通过随机采样的方式从经验缓冲池中选择样本进行训练。这在一定程度上减少了 DQNCG 的过拟合问题，也就减少了 DQNCG 在过度学习所有既正常又异常的事件序列模式。而 DQNCG 本身的机制是只要日志中包含一个异常事件序列，就将整个日志预测为异常日志。所以尽管 OpenStack 数据集中有大量重叠的既正常又异常的事件序列，DQNCG 会更容易朝着异常日志的方向进行预测。由于 OpenStack 的数据量相较于 HDFS 而言小太多，导致 DQNCG 难以充分训练而使得产生的结果相对不稳定（即无法完全拟合真实数据集的分布），这就导致 DQNAF 在训练的过程中会将 DQNCG 的不稳定结果视为正常日志事件序列进行学习。而 DQNAF 在拟合上比起 DQNCG 更加强，从而将 OpenStack 中大量的异常日志预测为正常日志。同样的问题在 HDFS 中则弱化了很多，因为 HDFS 中异常日志的事件序列性更强（即异常日志独有事件序列种类比例大）。

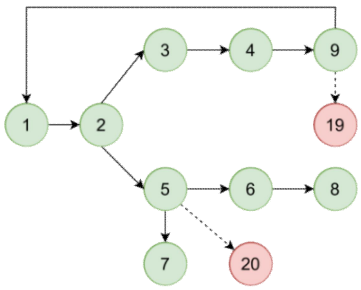
表 6 HDFS 与 OpenStack 数据集的区别

Table 6 The Difference Between HDFS and OpenStack Data Sets

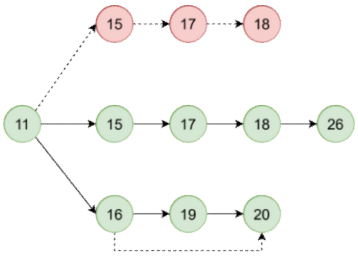
数据集统计指标	HDFS	OpenStack
异常日志独有事件序列数	39430	17
异常日志所有事件序列数	205285	4311
异常日志独有事件序列占比	19.21%	0.39%
异常日志独有事件序列种类数	5208	17
异常日志所有事件序列种类数	8503	81
异常日志独有事件序列种类占比	61.25%	20.99%
测试集与训练集重叠的异常日志事件序列总数	134031	4300
测试集中异常日志事件序列总数	205285	4311
测试集与训练集重叠的异常日志事件序列比例	65.29%	99.74%

2.11 DQNCG 案例分析

利用日志序列图进行异常检测的实际案例如图15所示。



(a)异常操作产生的序列异常



(b)操作不完整产生的序列异常

图15 案例分析

Fig.15 Case analysis

图15(a)反映正常工作流中出现异常操作或事件时的异常序列情况，图中共有三个正常的工作流，{1−2−3−4−9−1}路径代表的是完整数据的写入流程，系统接受客户端的请求(1)，然后对数据进行校验(2)，随后系统分配数据块(3)，之后建立数据管道(4)，最后通过数据节点的确认(9)，再次重新接受客户端的数据(1)。{1−2−5−6−8}路径代表的是元数据更新流程，数据校验完毕后，获取元数据锁(5)，然后执行Fslmage合并(6)，最终完成检查点(8)。{1－2－5－7}路径代表的是快速元数据操作，对于不需要持久化的元数据变更，系统在获取元数据锁(5)后，直接更新内存缓存(7)。正常序列图生成模型在训练完成后，输入初始的状态1，模型经过计算分析根据设定的阈值可以生成图15(a)，对于到来的日志序列，只有符合完整的路径才会被判定为正常。如图所示，如果到来的状态轨迹为{1－2－3－4－9－19}，说明系统通过数据节点的确认(9)后未能返回有效的响应，进入了写入失败状态(19)，此时模型将会将该条序列判定为异常。如果系统在获取元数据锁(5)时发现版本冲突，就会跳转到元数据异常状态，从而产生{1－2－5－20}路径，此路径在同正常序列子图进行比较时也找不到一条完整的路径，也会被视为异常。

图15(b)反映的是正常工作流中出现不完整序列时的异常情况。图中共有两个正常的工作流，{11－15－17－18－26}表示的是完整的数据读取流程，从接受客户端请求开始(11)，系统首先定位数据块副本(15)，然后与Data Node建立数据流链接(17)。在传输过程中，数据被分片送达(18)，最终完成校验并确认读取成功(26)。{11－16－19－20}展示了元数据批量操作的完整周期。系统在获取元数据批量锁(16)后，将多个Edit Log变更批量写入磁盘(19)，最后完成批量数据的检查点提交(20)。在客户端主动取消下载、Data Node节点突然宕机或网络连接不稳定的情况下可能发生数据读取意外终止，此时会产生路径{11－15－17－18}，此路径虽然前序和图中的路径完全匹配，但缺少最后的状态，也会被判定为异常。在磁盘写满、内存不足或出现文件系统权限问题时，会产生路径{11－16－20}，此时不会进行批量写入过程，直接进行检查点提交，并在提交时失败，此路径也和图中的子序列部分匹配，并未出现其他的状态，但依然会因为路径未完全匹配而被识别为异常。

2.12 DQNAF 案例分析

图16展示了输出状态为1时，模型会产生潜在的异常序列，序列长度受到异常预测模型和正常序列生成模型的共同影响，在异常预测模型中而不在正常序列模型中的事件不进行进一步的预测，保留预测结果，直到生成所有序列后一同输出；对重合的事件进行进一步的预测，直到达到最大预测序列长度，这样模型就能生成不同长度的异常序列，并对异常进行分级。路径{1－2}预测的是在任务提交以后(1)，进行数据块定位时(2)，出现了权限不足或Name Node不可达等问题。路径{1－3－5－6－7}的流程为在任务提交以后(1)，进行数据分片(3)，然后执行Map任务(5)，写入中间结果(6)，最后进行结果提交(7)，在结果提交时，发生了检验和冲突、副本写入失败等问题。路径{1－4－8－9－10－11}的流程为任务提交以后(1)，系统进行小文件合并(4)，然后进行跨集群复制(8)，再实现数据压缩(9)，然后更新元数据(10)，最后清理临时文件(11)，在清理临时文件时发生垃圾回收延迟的问题。

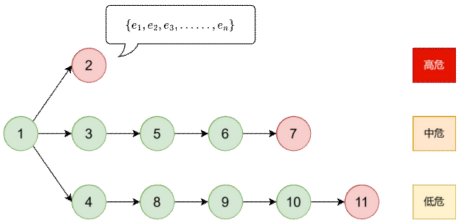


图 16 异常预测案例分析

Fig.16 Case analysis of anomaly prediction