Neurocomputing 520 (2023) 115–128

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Reinforcement learning based web crawler detection for diversity and dynamics

Yang Gao [a], Zunlei Feng [b,*], Xiaoyang Wang [a], Mingli Song [a], Xingen Wang [a], Xinyu Wang [a], Chun Chen [a]

[a] College of Computer Science, Zhejiang university, Hangzhou, China
[b] College of Software Technology, Zhejiang University, Hangzhou, China

ARTICLE INFO

Article history:
Received 9 July 2022
Revised 26 October 2022
Accepted 15 November 2022
Available online 19 November 2022
Communicated by Zidong Wang

Keywords:
Web crawler detection
Reinforcement learning
Feature selection
Crawler diversity
Crawler dynamics

ABSTRACT

Crawler detection is always an important research topic in network security. With the development of web technology, crawlers are constantly updating and changing, and their types are becoming diverse. The diversity and dynamics of crawlers pose significant challenges for feature applicability and model robustness. Existing crawler detection methods can only detect a limited number of crawlers by predefined rules and can not cover all types of crawlers; worse, they can be completely invalidated by the emergence of new types of crawlers. In this paper, we propose a reinforcement learning based web crawler detection method for diversity and dynamics (WC3D), which is composed of a feature selector and a session classifier. The feature selector selects the appropriate feature set for different types of crawlers with deep deterministic policy gradient. The session classifier makes crawler detection and provides rewards to the feature selector. The two modules are trained jointly to optimize the feature selection and session classification processes. Extensive experiments demonstrate the existence of crawler diversity and that the proposed method is still highly robust against the new type of crawlers and achieves state-of-the-art performance even without considering the dynamics of the crawlers.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

Web crawlers (also known as web spiders, web robots) are programs or scripts that automatically crawl web information with certain rules [1]. According to statistics, more than half of today's network requests are made by crawlers [2,3]. These crawlers include many malicious crawlers, which ignore the constraints of robots.txt [4], infringe on user privacy, endanger network security, and cause network traffic overload. What they do seriously affects the user's online experience [5–7]. Therefore, how to detect crawlers in numerous network requests has become an important research topic in the network security field.

With the continuous development of Web technology, the enrichment of Web content, and the advancement of crawler detection research, crawlers are constantly being updated and changed. The types of crawlers have become diverse, including search engine crawlers that collect web content, topic crawlers that crawl relevant content according to a specified topic, and image crawlers that are only interested in images [8,9]. Crawlers' diversity and dynamics bring the following challenges to crawler detection:

1. It isn't easy to extract universal and stable features. Different types of crawlers have some differences in the distribution of feature space. Some features that apply to some crawlers' detection do not apply to other types of crawlers. For example, image crawlers are highly differentiated from users for the feature of Image-request-ratio, while non-image crawlers are consistent with users. Crawler detection requires valid features for different types of crawlers;
2. New types of crawlers make pre-trained models or rules invalid. Some new types of crawlers that are not distinguishable in the mode's features will decrease the detection accuracy, and the model is no longer robust.

Many scholars start from the perspective of features, hoping to design universal and stable features to characterize crawlers, such as Resource-request-patterns, which characterizes the pattern of web request file types [10]; Penalty, which characterizes the behavior of back-and-forward navigation or loop, and Maximum-rate-of-browser-file, which characterizes the properties of the session for resource requests, etc [11]. Such methods detect crawlers by using

* Corresponding author.
E-mail addresses: roygao@zju.edu.cn (Y. Gao), zunleifeng@zju.edu.cn (Z. Feng), skyoung@zju.edu.cn (X. Wang), brooksong@zju.edu.cn (M. Song), newroot@zju.edu.cn (X. Wang), wangxinyu@zju.edu.cn (X. Wang), chenc@zju.edu.cn (C. Chen).

https://doi.org/10.1016/j.neucom.2022.11.059
0925-2312/© 2022 Elsevier B.V. All rights reserved.

Neurocomputing 520 (2023) 115–128

内容列表可在 ScienceDirect Neurocomputing 期刊主页找到：
www.elsevier.com/locate/neucom

基于强化学习的针对多样性和动态性的网络爬虫检测

Yang Gao [a], Zunlei Feng [b,*], Xiaoyang Wang [a], Mingli Song [a], Xingen Wang [a], Xinyu Wang [a], Chun Chen [a]

[a] College of Computer Science, Zhejiang university, Hangzhou, China
[b] College of Software Technology, Zhejiang University, Hangzhou, China

ar ticl e i nfo （原文未进行翻译，因为输入看起来像是文本信息或代码。）
文章历史：收到日期 2022 年 7 月 9 日
修订日期 2022 年 10 月 26 日 接受日期
2022 年 11 月 15 日 网上发布日期
2022 年 11 月 19 日 通讯作者：王志东

关键词：网络爬虫检测
强化学习 特征选择 爬虫
多样性 爬虫动态性

摘要

网络爬虫检测一直是网络安全领域的重要研究课题。随着网络技术的不断发展，爬虫不断更新和变化，其类型也变得多样化。爬虫的多样性和动态性对特征适用性和模型鲁棒性提出了重大挑战。现有的爬虫检测方法只能通过预定义的规则检测有限数量的爬虫，并且无法涵盖所有类型的爬虫；更糟糕的是，它们可能会被新类型爬虫的出现完全失效。在本文中，我们提出了一种基于强化学习的针对多样性和动态性的网络爬虫检测方法（ WC3D ），该方法由特征选择器和会话分类器组成。特征选择器使用深度确定性策略梯度选择不同类型爬虫的适当特征集。会话分类器进行爬虫检测并向特征选择器提供奖励。这两个模块共同训练以优化特征选择和会话分类过程。大量的实验表明，存在爬虫多样性，并且所提出的方法对新型爬虫具有高度的鲁棒性，即使在未考虑爬虫动态性的情况下，也实现了最先进的性能。

版权所有 Elsevier B.V.

1. 引言

Web 爬虫（也称为网络蜘蛛、网络机器人）是按照一定规则自动爬取网络信息的程序或脚本 [1]。根据统计，今天超过一半的网络请求都是由爬虫完成的 [2,3]。这些爬虫包括许多恶意爬虫，它们无视 robots.txt [4]，的约束，侵犯用户隐私，危害网络安全，并导致网络流量过载。它们的行为严重影响了用户的在线体验 [5–7]。因此，如何在众多网络请求中检测爬虫已成为网络安全领域的一个重要研究课题 。

随着 Web 技术的不断发展，Web 内容的丰富以及爬虫检测研究的进步，爬虫不断更新和变化。爬虫的类型变得多样化，包括收集 Web 内容的搜索引擎爬虫、根据特定主题爬取相关内容的主题爬虫，以及只对图像感兴趣的图像爬虫 [8,9]。爬虫的多样性和动态性给爬虫检测带来了以下挑战：

多样性和动态性给爬虫检测带来了以下挑战：

1. 提取通用和稳定的特征并不容易。不同类型的爬虫在特征空间分布上存在一些差异。一些适用于某些爬虫检测的特征并不适用于其他类型的爬虫。例如，图像爬虫在 Image-request-ratio 特征上与用户高度区分，而非图像爬虫则与用户一致。爬虫检测需要针对不同类型的爬虫提取有效特征；

2. 新类型的爬虫使得预训练模型或规则失效。一些在新类型爬虫的特征中无法区分的爬虫将降低检测精度，模型不再稳健。

许多学者从特征的角度出发，希望设计出通用且稳定的特征来表征爬虫，例如资源请求模式，表征网页请求文件类型的模式；惩罚，表征前后导航或循环的行为，以及最大浏览器文件速率，表征资源请求会话的特性等。这些方法通过使用

通讯作者。电子邮件地址：roygao@zju.edu.cn (G. 高)，zunleifeng@zju.edu.cn (F. 风)，skyoung@zju.edu.cn (W. 王)，brooksong@zju.edu.cn (S. 宋)，newroot@zju.edu. cn (W. 王)，wangxinyu@zju.edu.cn (W. 王)，chenc@zju.edu.cn (C. 陈)。

https://doi.org/10.1016/j.neucom.2022.11.059 Elsevier B.V. 版权所有。

artificially defined feature sets, as shown in the first row of Fig. 1. As the number of features increases, some scholars have added feature selection methods, hoping to remove redundant features and keep the robust ones [12,13]. Such methods use feature selection to determine the feature set used for crawler detection, as shown in the second row of Fig. 1. These methods solve the problem of crawlers' diversity to a certain extent, but they also have some disadvantages:

1. the feature set is single, and all types of crawlers use the same feature set, and some features that can strongly characterize a specific type of crawlers are not selected;
2. the feature selection process is separated from the crawler detection task, and most of the selection methods using a greedy strategy, which easily causes the local optimal problem.

The last and the essential point is that the feature set used can not change dynamically. As the crawlers update and change, the selected features may become invalid, while the previously unselected features may become effective. And the robustness of the model will be significantly reduced when the used features are no longer suitable for distinguishing the new type of crawlers.

To address the challenges posed by the diversity and dynamics of crawlers, we want to propose a feature selection and detection method for the new types of crawlers that can sense the diversity of crawlers and dynamically select their characterization feature sets according to their types. We introduce a reinforcement learning method in which we model the feature selection as a sequential decision process, and we use the classification accuracy as the reward to optimize the screening process, which can avoid the local optimal problem; at the same time, we use the feature space of the crawler as the state space, and the agent not only learns the potential connection between features but also perceives more unknown feature distributions through exploration, which allows the model to better adapt to the updates and changes of crawlers and the emergence of the new type of crawlers. Our approach consists of two modules, a feature selection module based on deep reinforcement learning and a classification module composed of deep neural networks. The feature selection module optimizes the filtering process based on the results of the classification module, and the classification module uses the feature set provided by the feature selection module to detect crawlers. As shown in the third row of Fig. 1, our method can select different feature sets for different types of crawlers to better accommodate the diversity of crawlers and still achieve high robustness in detecting the new types of crawlers.

The contributions of this paper are as follows:

1. We propose a feature selection method for the diversity of crawlers, and the selected feature set has diversity and higher applicability.
2. We propose a crawlers detection method based on reinforcement learning, which can maintain high detection accuracy when crawlers have diversity and dynamics and have good robustness in the face of the new types of crawlers.
3. There are few public data sets in the field of crawler detection, which has caused certain obstacles to the development of research in this field. This paper publicizes a crawler data set for scholars' follow-up research convenience.

The content of this paper is organized as follows: In the second part, we introduce the recent work in the field of crawler detection. The third part introduces the feature selection and classification method for the new types of crawlers based on deep reinforcement learning. We give the experimental results in the fourth part. The fifth part is the summary of this article.

2. Related work

In this part, we will review some recent research in crawler detection, crawler diversity and feature selection.

A web crawler is a program or script that automatically crawls the World Wide Web for information according to specific rules. Well-intentioned crawlers have the functions of communicating with users, maintaining mirror sites, testing the validity of web pages, etc. However, with the development of web technology, more and more malicious crawlers have emerged, extorting users, illegally collecting information, DDos attacks, stealing users' privacy, and taking up network bandwidth to affect users' online experience. Due to the diversity of websites and the diversity of crawling rules, crawlers also have significant differences. Some of them crawl only specific resources while others collect all types of resources; some use breadth-first strategy to crawl content while others use depth-first; in addition, with the advancement of research on crawler detection, crawlers are constantly updated and changed to better mimic user behavior and thus evade website detection. The crawlers' diversity and dynamics pose a more significant challenge to the accuracy of crawler detection. Scholars have done a lot of research in this area, and they classify crawler detection methods into the following four categories [14]: syntactic log analysis, traffic pattern analysis, Turing test systems and analytical learning techniques. Syntactic log analysis: This method identifies crawlers based on log files by matching keywords in logs or matching IP and user agents with the whitelist. This method is relatively straightforward. Some initial research for crawler detection is based on this method [15,16]. Traffic pattern analysis: This method analyzes the differences in access patterns between crawlers and users and detects crawlers by comparing the access patterns obtained by simulating a specified algorithm (such as DFS or BFS) to crawlers and calculating the similarity between them [17–21]. Turing test systems: This method uses the Turing test for real-

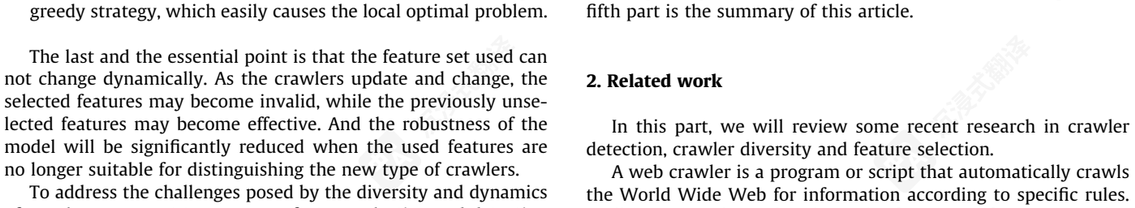


Fig. 1. Illustrative the challenges posed by the diversity and dynamics of crawlers. The diversity of crawlers makes the distribution of feature space different. Each type has its appropriate feature set for representing its characteristics, and the darker the color of the features in the graph, the more suitable for representing. The first row indicates the method with an artificially defined feature set, which uses the first and third features for crawler detection, the second row indicates the method with feature selection approach, which retains the first three features after filtering, and the third line indicates that our approach selects theirs appropriate feature set according to the different types of crawlers. When the new type of crawlers appear, the features used in the first two methods can not characterize them well and will lead to a decrease in detection accuracy.

如图 1 的第一行所示，人工定义的特征集。随着特征数量的增加，一些学者添加了特征选择方法，希望去除冗余特征并保留鲁棒的特征。这些方法使用特征选择来确定用于爬虫检测的特征集，如图 1 的第二行所示。这些方法在一定程度上解决了爬虫的多样性问题，但它们也存在一些缺点：

1. 特征集单一，所有类型的爬虫使用相同的特征集，并且一些可以强烈表征特定类型爬虫的特征没有被选中；2. 特征选择过程与爬虫检测任务分离，大多数使用贪婪策略的选择方法，容易导致局部最优问题。

最后和最关键的一点是，所使用的特征集不能动态变化。随着爬虫的更新和变化，选定的特征可能变得无效，而之前未选中的特征可能变得有效。当使用的特征不再适合区分新类型的爬虫时，模型的鲁棒性将显著降低。

为了解决爬虫多样性和动态性带来的挑战，我们希望提出一种针对新类型爬虫的特征选择和检测方法，该方法能够感知爬虫的多样性，并根据其类型动态选择其表征特征集。我们引入了一种强化学习方法，将特征选择建模为顺序决策过程，并使用分类准确率作为奖励来优化筛选过程，从而避免局部最优问题；同时，我们将爬虫的特征空间作为状态空间，代理不仅学习特征之间的潜在联系，而且通过探索感知更多未知特征分布，这使得模型能够更好地适应爬虫的更新和变化以及新类型爬虫的出现。我们的方法由两个模块组成，一个基于深度强化学习的特征选择模块和一个由深度神经网络组成的分类模块。特征选择模块根据分类模块的结果优化筛选过程，而分类模块使用特征选择模块提供的特征集来检测爬虫。如图 1 的第三行所示，我们的方法可以为不同类型的爬虫选择不同的特征集，以更好地适应爬虫的多样性，并在检测新类型爬虫时仍保持高鲁棒性。

本文的贡献如下：

1. 我们提出了一种用于爬虫多样性的特征选择方法，所选特征集具有多样性和更高的适用性。
2. 我们提出了一种基于强化学习的爬虫检测方法，该方法能够在爬虫具有多样性和动态性时保持高检测精度，并且面对新型爬虫具有良好的鲁棒性。3. 爬虫检测领域公开的数据集很少，这给该领域的研究发展带来了一定的障碍。本文公布了一个爬虫数据集，以方便学者们后续研究。

本文的内容组织如下：在第二部分，我们介绍了爬虫检测领域的近期工作。第三部分介绍了基于深度强化学习的新类型爬虫的特征选择和分类方法。第四部分给出了实验结果。第五部分是本文的总结。

2. 相关工作

在本部分，我们将回顾一些关于爬虫检测、爬虫多样性和特征选择的研究成果。

网络爬虫是一种根据特定规则自动爬取万维网信息的程序或脚本。善意爬虫具有与用户沟通、维护镜像站点、测试网页有效性等功能。然而，随着网络技术的发展，越来越多的恶意爬虫出现，勒索用户、非法收集信息、 DDoS 攻击、窃取用户隐私、占用网络带宽影响用户在线体验。由于网站多样性和爬取规则多样性，爬虫也存在显著差异。有些只爬取特定资源，而有些收集所有类型的资源；有些使用广度优先策略爬取内容，而有些使用深度优先；此外，随着爬虫检测研究的发展，爬虫不断更新和改变，以更好地模仿用户行为，从而规避网站检测。爬虫的多样性和动态性对爬虫检测的准确性提出了更大的挑战。学者们在这个领域进行了大量研究，并将爬虫检测方法分为以下四类 [14]：语法日志分析、流量模式分析、图灵测试系统和分析学习技术。语法日志分析：这种方法通过匹配日志中的关键词或匹配 IP 和用户代理与白名单来识别爬虫。这种方法相对直接。一些爬虫检测的初步研究基于这种方法 [15,16]。流量模式分析：这种方法通过分析爬虫和用户之间的访问模式差异，通过比较通过模拟特定算法（如 DFS 或 BFS ）获得的访问模式与爬虫之间的相似性来检测爬虫 [17–21]。图灵测试系统：这种方法使用图灵测试进行真实 -

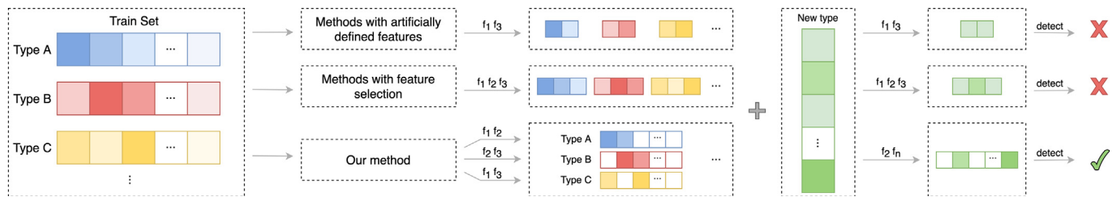


图 1. 阐述了爬虫多样性和动态性带来的挑战。爬虫的多样性使得特征空间的分布不同。每种类型都有其适当的特征集来表示其特征，图中特征的颜色越深，表示越适合表示。第一行表示具有人工定义特征集的方法，该方法使用第一和第三特征进行爬虫检测，第二行表示具有特征选择方法的方法，该方法在过滤后保留了前三个特征，第三行表示我们的方法根据不同类型的爬虫选择合适的特征集。当出现新的爬虫类型时，前两种方法使用的特征无法很好地表征它们，会导致检测精度下降。

time detection of crawlers. This method requires software-level modifications to the website. A more intuitive understanding is that the website distinguishes crawlers from users through verification codes. Some studies have adopted this method [22,23]. Analytical learning techniques: This method extracts features from sessions and uses machine learning to obtain detection results. With the continuous development of machine learning, many scholars have adopted different models and features for crawler detection. They have obtained relatively ideal results, which proves the effectiveness of this type of method. Our crawler detection model based on reinforcement learning also belongs to the category of analytical learning techniques, so we will focus on the related work of this method in the following text.

Some scholars hope to solve crawler diversity and dynamics by proposing stable and universal features for different types of crawlers. Tan, Kumar et al. were the first to propose 25 new features and use C4.5 decision trees to classify website visitors, which contain some statistical features such as the proportion of nightly visits, unassigned referrer, session time, etc. These statistical features were also frequently used in subsequent studies [1]. Doran, Gokhale et al. proposed a resource request pattern feature that represents the sequence of requests for different types of files, and the authors used a probabilistic model to analyze the differences between crawlers and users on this feature [10]. Lagopoulos et al. propose a crawler detection method for Linear Discriminant Analysis(LDA) based on the assumption that users are only interested in specific topics and crawlers will randomly crawl various topics. The authors also propose some novel features that can capture the semantics of the requested resource content, which improve the crawler detection accuracy [24]. Zabihi, Jahan et al. proposed two new features in their paper: Maximum rate of browser file request and penalty. The former characterizes the number property of all resource files bound to the visited page. The latter characterizes the behavior of web request back-and-forward navigation or loop. The authors concluded that these two features do not change over time, and they used the Density-Based Spatial Clustering of Applications with Noise(DBSCAN) algorithm for crawler detection [12]. Hiltunen et al. used the number and sequence of visited pages as features and used the Self-organizing Maps(SOM) method for crawler detection [25]. Zhu, Gao et al. proposed a machine learning-based approach that combines real-time monitoring with offline detection with two new heuristic rules. Eventually, they used nine features for crawler detection, and the model was able to achieve both high recognition accuracy and fast response time [26]. These methods use human-defined feature sets and combine different machine learning methods for crawler detection, which address the challenges to the applicability of features posed by the diversity of crawlers to some extent.

As the number of features increases, there are more and more redundant features. Some scholars believe that these redundant features increase the complexity of the model [27–29] and the feature set defined by humans has a solid subjective nature, which does not objectively judge the merits of the features, so they use feature selection to decide the feature set used for crawler detection. Many scholars use decision trees for feature filtering, Kwon et al. used decision trees to classify web sessions and proposed some new behavioral features of web crawlers based on features from previous studies [18]. Toni et al. proposed an intelligent system called Lino to identify whether a web crawler is malicious or not. They used support vector machine and C4.5 decision tree algorithm respectively, and compared the performance of these two algorithms in the Lino system [30]. Hamidzadeh et al. pointed out that website visitors have variability, crawlers' tasks have diversity, and especially the content of websites also has diversity. These three problems indicate the need for a crawler detection method that can dynamically select features according to the content of different websites for different tasks. They used the Fuzzy Rough Set(FRS) algorithm to select the features and classified them using the SOM method. The FRS algorithm was able to consider the similarity of all features simultaneously and deal with the ambiguity of the data effectively. Finally, it also achieved better experimental results [13]. Zabihi et al. used t-test to select the features with higher relevance among the 14 valid attributes. Although their proposed method has effective performance in distinguishing between crawlers and human users, the t-test is only based on the summary statistics of the compared attributes and has a more limited observation of the data [31]. The above study adds feature selection process to crawler detection, which reduces the complexity of the model by sifting out redundant features and makes the selected features more interpretable while solving the feature applicability problem.

The two types of methods mentioned above both start from the feature perspective and propose or select stable and universal features to address the challenges of the diversity of crawlers. Although these methods have achieved specific results, they still have some defects: 1. The feature set used is single; all types of crawlers use the same feature set, and some features that can well characterize a specific type of crawlers are not selected; 2. Some feature selection methods separated from the detection task tend to make the model fall into the local optimum problem. 3. Most importantly, they do not address the problem of dynamics of crawlers. As crawlers update and change, the feature set they use may not be suitable for the new types of crawlers, and the previously unselected features may become more effective. Doran, Gokhale et al. point out that crawler behavior patterns may change over time and that the features proposed by scholars are a weak reflection of crawler and user behavior, which may not reflect the essential differences between them [10].

To address these issues, we need a method that can perceive the diversity of crawlers and adjust the representation of crawler behavior patterns according to the dynamics of crawlers. In the problem of perceiving dynamic changes, reinforcement learning has always performed well [32,33], and in recent years some scholars have applied reinforcement learning to the problem of feature selection. Janisch et al. use Q-learning for feature selection and classification, and they design a reinforcement learning model with two types of actions in the action space. One is to add the unselected features to the feature set, and the other is to classify directly with the current set of selected features. They combined the classification task with the feature selection task, and the results of classification affect the feature selection process as a reward [34]. Xu, Wang et al. used reinforcement learning to extract the more important features from the temporal features, and the importance of their features was also influenced by the classification results [35]. Feng, Huang et al. also used reinforcement learning to improve the accuracy of the classification task; instead of using reinforcement learning to filter features, they filtered data used to train the model, but the principle is the same [36]. Inspired by the above studies, we wish to apply reinforcement learning to the crawler detection problem to solve the problem of sensing crawler diversity and dynamically adjusting the crawler's representational patterns. To this end, we propose a reinforcement learning-based crawler feature selection and detection method, which will be described in detail in the next section.

3. Method

To solve the problem of crawlers' diversity and dynamics mentioned above, we propose a new crawler detection method based

时间检测爬虫。这种方法需要对网站进行软件级别的修改。更直观的理解是，网站通过验证码来区分爬虫和用户。一些研究采用了这种方法[22,23]。分析学习技术：这种方法从会话中提取特征，并使用机器学习来获得检测结果。随着机器学习的不断发展，许多学者采用了不同的模型和特征进行爬虫检测。他们获得了相对理想的结果，这证明了这种方法的有效性。我们基于强化学习的爬虫检测模型也属于分析学习技术的范畴，因此我们将在下文中重点关注这种方法的相关工作。

一些学者希望通过为不同类型的爬虫提出稳定和通用的特征来解决爬虫的多样性和动态性。Tan 、 Kumaret 等人首先提出了 25 个新特征，并使用 C4.5 决策树对网站访客进行分类，这些特征包括夜间访问比例、未分配的引用者、会话时间等统计特征。这些统计特征也经常在后续研究中被使用 [1]。Doran 、 Gokhale 等人提出了一种资源请求模式特征，它表示对不同类型文件的请求序列，作者使用概率模型分析了爬虫和用户在这方面的差异 [10]。Lagopoulos 等人基于用户只对特定主题感兴趣，而爬虫会随机爬取各种主题的假设，提出了基于线性判别分析（ LDA ）的爬虫检测方法。作者还提出了一些可以捕获请求资源内容语义的新特征，这些特征提高了爬虫检测的准确性 [24]。Zabihi 、 Jahan 等人在他们的论文中提出了两个新特征：浏览器文件请求的最大速率和惩罚。前者描述了与访问页面绑定的所有资源文件的数量属性。后者描述了网页请求的回退导航或循环行为。作者得出结论，这两个特征不会随时间变化，他们使用基于密度的空间聚类应用噪声（ DBSCAN ）算法进行爬虫检测 [12]。Hiltunen 等人使用访问页面的数量和顺序作为特征，并使用自组织映射（ SOM ）方法进行爬虫检测 [25]。Zhu 、 Gao 等人提出了一种基于机器学习的方法，该方法结合了实时监控和离线检测，并引入了两个新的启发式规则。最终，他们使用了九个特征进行爬虫检测，该模型能够实现高识别准确率和快速响应时间 [26]。这些方法使用人类定义的特征集，并结合不同的机器学习方法进行爬虫检测，在一定程度上解决了爬虫多样性对特征适用性提出的挑战。

随着特征数量的增加，冗余特征越来越多。一些学者认为，这些冗余特征增加了模型的复杂性 [27–29]，而人类定义的特征集具有明显的主观性，不能客观地判断特征的优劣，因此他们使用特征选择来决定用于爬虫检测的特征集。许多学者使用决策树进行特征过滤，Kwon 等人使用决策树对 Web 会话进行分类，并基于先前研究提出了一些新的爬虫行为特征 [18]。Toni 等人提出了一种名为 Lino 的智能系统来识别 Web 爬虫是否恶意。他们分别使用了支持向量机和 C4.5 决策树算法，并比较了这两种算法在 Lino 系统中的性能 [30]。Hamidzadeh 等人指出，网站访问者具有可变性，爬虫的任务具有多样性，尤其是网站的内容也具有多样性。这三个问题表明了爬虫检测的必要性

一种可以根据不同网站的内容和不同任务动态选择特征的方法。他们使用模糊粗糙集（ FRS ）算法选择特征，并使用 SOM 方法进行分类。FRS 算法能够同时考虑所有特征的相似性，并有效地处理数据的模糊性。最后，它也实现了更好的实验结果 [13]。Zabihi 等人使用 t 检验从 14 个有效属性中选择相关性较高的特征。尽管他们提出的方法在区分爬虫和人类用户方面具有有效的性能，但 t 检验仅基于比较属性的汇总统计，对数据的观察更为有限 [31]。上述研究将特征选择过程添加到爬虫检测中，通过筛选冗余特征降低了模型的复杂性，同时解决了特征适用性问题，并使选定的特征更具可解释性。

上述提到的两种方法都是从特征角度出发，提出或选择稳定且通用的特征来解决爬虫多样性的挑战。尽管这些方法取得了特定成果，但它们仍存在一些缺陷：1. 使用的特征集单一；所有类型的爬虫都使用相同的特征集，而没有选择能够很好地表征特定类型爬虫的一些特征；2. 一些与检测任务分离的特征选择方法容易导致模型陷入局部最优问题；3. 最重要的是，它们没有解决爬虫动态变化的问题。随着爬虫的更新和变化，它们使用的特征集可能不再适合新的爬虫类型，而之前未选择的特征可能变得更加有效。Doran 、 Gokhale 等人指出，爬虫的行为模式可能会随时间而变化，而学者们提出的特征只是对爬虫和用户行为的弱反映，可能无法反映它们之间的本质差异。

为了解决这些问题，我们需要一种能够感知爬虫多样性的方法，并根据爬虫的动态变化调整爬虫行为模式的表示。在感知动态变化的问题上，强化学习一直表现良好，近年来一些学者将强化学习应用于特征选择问题。Janisch 等人使用 Q 学习进行特征选择和分类，他们在动作空间中设计了两种类型的动作。一种是向特征集中添加未选择的特征，另一种是直接使用当前选择的特征集进行分类。他们将分类任务与特征选择任务相结合，分类的结果会影响特征选择过程作为奖励。Xu 、 Wang 等人使用强化学习从时间特征中提取更重要的特征，他们特征的重要性也受到分类结果的影响。Feng 、 Huang 等人也使用强化学习来提高分类任务的准确性；他们不是使用强化学习来过滤特征，而是过滤用于训练模型的原始数据，但原理是相同的。受上述研究启发，我们希望将强化学习应用于爬虫检测问题，以解决感知爬虫多样性和动态调整爬虫表示模式的问题。为此，我们提出了一种基于强化学习的爬虫特征选择和检测方法，将在下一节中详细介绍。

3. 方法

为了解决上述提到的爬虫多样性和动态性问题，我们提出了一种新的爬虫检测方法。

on reinforcement learning, which can sense the diversity of crawlers and dynamically select their appropriate set of representative features for different types of crawlers. Our method avoids the problems of single and static feature sets, making it suitable for detecting the new type of crawlers.

Our model consists of two modules: the feature selector module, which is responsible for selecting a feature set suitable for characterizing the session according to its feature distribution, and a session classifier module, which is responsible for classifying a session using the feature set provided by the feature selector. These two modules interact during the training process to achieve the best detection results.

3.1. Problem definition

Formally, we decompose the task of web crawler detection with diversity and dynamics into two sub-problems in this paper: feature selection and session classification.

We formulate the feature selection problem as follows: given the known multi-type crawlers set as $C = \{c_1, c_2, \ldots, c_m\}$, where c_i is a type of crawler that we have observed and an unknown new type of crawler that we have not observed as $c\prime$. Different types of crawlers have different characteristics and feature distribution. The goal of feature selection is to determine which features can better represent its characteristic for different crawlers according to the known crawlers set C, and it still works when dealing with unknown crawler $c\prime$. The output of feature selection is the feature masks as $M = \{m_1, m_2, \ldots, m_m, m\prime\}$, where m_i is the feature mask of ith type of crawler that we have observed and $m\prime$ is the feature mask of the new type of crawler, and $m_i, m\prime = \{k_1, k_2, \ldots, k_n\}$, where k_i is the ith feature's mask and is 1 or 0, which means selecting or discarding this feature.

The session classification problem is formulated as follows: given the initial feature vector of a session as $s = \{f_1, f_2, \ldots f_n\}$ and its feature mask as $m = \{k_1, k_2, \ldots k_n\}$, we can get the final feature vector as $s\prime = \{f\prime_1, f\prime_2, \ldots, f\prime_n\}$ where $f\prime_i$ is the result of multiplying the ith feature f_i with its feature mask k_i, the goal is to predict the likelihood that it will be a crawler or user under this final feature vector.

3.2. Overview

The proposed model consists of the feature selector and the session classifier. The feature selector is based on deep deterministic policy gradient with sessions as input. Each session s_i belongs to a specific type of crawler or user and has a corresponding action a_i to indicate its feature mask. Different types of crawlers can obtain different feature masks. And the feature mask represents which features should be retained or discarded to represent its characteristics better. The session classifier adopts a deep neural network. It uses the initial feature vector and feature mask from the feature selector as input to classify sessions. Meanwhile, the session classifier gives feedback to the feature selector to refine its policy. Fig. 2 gives an illustration of how the proposed model works.

With the help of the feature selector based on reinforcement learning, our method can select the most suitable feature set for different types of crawlers, avoiding the problem that features screening results are simplified and static. And using reinforcement learning can avoid the problem of local optimization. All of our designs aim at making the model work well in complex network environments with crawlers' diversity and dynamics.

3.3. Feature selector

We cast feature selector as a reinforcement learning problem. As we all know, reinforcement learning needs an agent and environment. In our method, the agent is the feature selector, and the environment is the session classifier. The agent follows a policy to decide which features should be retained and output feature mask and then receive a reward depending on the accuracy of classification from the session classifier. We will introduce the state, action, state transition, terminal state, and reward as follows.

- **State**
  The state s_i represents the current feature distribution of a session, which can be formulated as: $s_i = \{f_1, f_2, \ldots, f_n\}$, where f_i is a specific value of a feature, and n is the number of features. The initial state of a session is its feature distribution in the dataset, which is extracted from a log file. With the screening of the feature selector, its feature distribution (state) will also change.
- **Action**
  First, we define the output of policy network as $p_i = \{w_1, w_2, \ldots, w_n\}$ to indicate the importance of each feature, where $w_i \in (0, 1)$, corresponding to the weight value assigned by the agent to the ith feature of the session, and a larger value of w_i indicates that the ith feature is more critical to this session. Then we get the action a_i by the following formula:

$$a_i = T(\mu(s_i|\theta^{\mu}) + \mathcal{N}_i) \tag{1}$$

  where μ is the policy network, $\mathcal{N}_i$ is Uhlenbeck-ornstein random process [37], θ^{μ} is the parameters of policy network and T is a mapping function that uses the predefined threshold t_1 to map the weights w_i to feature mask m_i. If w_i is greater than or equal to t_1, then the corresponding m_i is 1, otherwise 0. Thus the action can be formulated as $a_i = \{m_1, m_2, \ldots, m_n\}$, and $m_i \in \{0, 1\}$, which means discarding or retaining the ith feature.
- **State Transition**
  We define the state transition as $s_t \times a_t \rightarrow s_{t+1}$, which means we multiply the current feature distribution by its mask to obtain a new feature distribution (the next state).
- **Terminal State**
  The definition of the terminal state is required in the modeling of reinforcement learning, and no more state transfer will be performed when the agent reaches the terminal state. Our task is to classify the session correctly, so we use the logarithm of the output probability of the session classifier as a condition to determine whether the agent enters the terminal state. When $logP(c_g|s_i)$ is greater than our predefined threshold t_2, e.g., 0.8, we consider that the current feature distribution is sufficient to characterize the session, and the agent enters the terminal state, then its feature distribution will no longer change.
- **Reward**
  We use the logarithm of the output probability of session classifier to calculate the reward: $P(y = c_g|s_i)$, where c_g is the gold label of the input session s_i. In addition, to encourage the model to screen out more features that can not characterize the session, we include an additional term by computing the proportion of the number of discarded features to the number of all features. When the agent enters the terminal state, we use the classification result as the reward, giving a larger reward if the classification is correct and a negative reward if the result is wrong. The reward can be described as below:

$$r_i = \begin{cases} 1 & \text{if } s_i \in \mathcal{T}, y = c_g \\ logP(c_g|s_i) + \alpha N'/N & \text{if } s_i \notin \mathcal{T} \\ -1 & \text{if } s_i \in \mathcal{T}, y \neq c_g \end{cases}$$

基于强化学习的网络爬虫检测，能够感知爬虫的多样性并动态选择适合不同类型爬虫的代表特征集。我们的方法避免了单一和静态特征集的问题，使其适用于检测新型爬虫。

我们的模型由两个模块组成：特征选择模块，负责根据特征分布选择适合描述会话的特征集；会话分类模块，负责使用特征选择模块提供的特征集对会话进行分类。这两个模块在训练过程中相互交互，以实现最佳的检测效果。

3.1. 问题定义

在本文中，我们将具有多样性和动态性的网络爬虫检测任务分解为两个子问题：特征选择和会话分类。

我们将特征选择问题表述如下：给定已知的多种类型爬虫集合 C {v1}c{v2}1{v3}; c{v4}2{v5};{v6}; c{v7}，其中 c{v8} 是我们观察到的爬虫类型，以及我们尚未观察到的未知类型爬虫 c{v9}。不同类型的爬虫具有不同的特征和特征分布。特征选择的目标是根据已知的爬虫集合 C，确定哪些特征可以更好地代表不同爬虫的特征，并且当处理未知爬虫 c {v10} 时仍然有效。特征选择的输出是特征掩码 M {v11}m{v12}1{v13}; m{v14}2{v15};{v16}; m{v17}; m{v18}，其中 m{v19} 是我们观察到的第 i 种类型爬虫的特征掩码，m{v20} 是新型爬虫的特征掩码，k{v21}; k {v22}k{v23}1{v24}; k{v25}2{v26};{v27}; k{v28}，其中 k{v29} 是第 i 个特征的特征掩码，为 1 或 0，表示选择或丢弃此特征。

会话分类问题表述如下：给定会话的初始特征向量 s {v1}f{v2}1{v3}; f {v4}2{v5}; {v6}f {v7} 及其特征掩码 m {v8}k{v9}1{v10}; k{v11}2{v12}; {v13}k{v14}，我们可以得到最终的特征向量 s{v15}f0{v16}1{v17}; f 0 {v18}2{v19}; {v20}; f {v21}，其中 f {v22} 是第 i 个特征 f{v23} 与其特征掩码 k{v24} 相乘的结果，目标是预测在最终特征向量下它将是爬虫还是用户的可能性。

3.2. 概述

所提出的模型由特征选择器和会话分类器组成。特征选择器基于深度确定性策略梯度，以会话作为输入。每个会话属于特定类型的爬虫或用户，并有一个相应的动作来指示其特征掩码。不同类型的爬虫可以获得不同的特征掩码。特征掩码表示应该保留或丢弃哪些特征以更好地表示其特征。会话分类器采用深度神经网络。它使用特征选择器提供的初始特征向量和特征掩码来分类会话。同时，会话分类器向特征选择器提供反馈以优化其策略。图 2 展示了所提出的模型的工作原理。

借助基于强化学习的特征选择器，我们的方法可以为不同类型的爬虫选择最合适的特征集，避免特征筛选结果简化且静态的问题。使用强化学习可以避免局部优化的问题。我们所有的设计都是为了使模型在具有爬虫多样性和动态性的复杂网络环境中工作得更好。

3.3. 特征选择器

我们将特征选择器视为一个强化学习问题。众所周知，强化学习需要一个代理和环境。在我们的方法中，代理是特征选择器，环境是会话分类器。代理遵循策略来决定应该保留哪些特征，并输出特征掩码，然后根据会话分类器的分类准确性接收奖励。以下我们将介绍状态、动作、状态转移、终止状态和奖励。

- 状态
  状态 s_i 表示会话当前的特征分布，可以表示为：$s_i = \{f_1; f_2; \ldots; f_n\}$，其中 f_i 是特征的一个特定值，n 是特征的数量。会话的初始状态是其在数据集中的特征分布，该分布是从日志文件中提取的。经过特征选择器的筛选，其特征分布（状态）也会发生变化。

- 动作
  首先，我们将策略网络的输出定义为 $p_i = \{w_1; w_2; \ldots; w_n\}$，以表示每个特征的重要性，其中 $w_i \in (0; 1)$ 对应于智能体分配给会话第 i 个特征的权重值，w_i 的值越大，表示第 i 个特征对这个会话越关键。然后我们通过以下公式得到动作 a_i：

$$a_i = T(\mu(s_i|\theta^{\mu}) + \mathcal{N}_i) \tag{1}$$

  其中 μ 是策略网络，N_i 是 Uhlenbeck-Ornstein 随机过程 [37]，h_{μ} 是策略网络的参数，T 是一个映射函数，它使用预定义的阈值 t_1 将权重 w_i 映射到特征掩码 m_i。如果 w_i 大于或等于 t_1，则相应的 m_i 为 1，否则为 0。因此，动作可以表示为 $a_i = \{m_1; m_2; \ldots; m_n\}$，以及 $m_i \in \{0; 1\}$，这意味着丢弃或保留第 i 个特征。• St

ate Transition
我们将状态转移定义为 $s_t \times a_t \rightarrow s_{t+1}$，这意味着我们将当前特征分布与其掩码相乘以获得新的特征分布（下一个状态）。

- 终止状态
  在强化学习的建模中，需要定义终止状态，当智能体达到终止状态时，将不再执行状态转移。我们的任务是正确分类会话，因此我们使用会话分类器的输出概率的对数作为条件来判断智能体是否进入终止状态。当 $\log P(c_g|s_i)$ 大于我们预先定义的阈值 t_2，例如，0.8，我们认为当前特征分布足以表征会话，智能体进入终止状态，然后其特征分布将不再改变。

- 奖励
  我们使用会话分类器的输出概率的对数来计算奖励：$P(y = c_g|s_i)$，其中 c_g 是输入会话 s_i 的真实标签。此外，为了鼓励模型筛选出更多不能表征会话的特征，我们通过计算被丢弃的特征数与所有特征数的比例来包含一个额外的项。当智能体进入终止状态时，我们使用分类结果作为奖励，如果分类正确则给予更高的奖励，如果结果错误则给予负奖励。奖励可以描述如下：

$$r_i = \begin{cases} 1 & \text{if } s_i \in \mathcal{T}, y = c_g \\ logP(c_g|s_i) + \alpha N'/N & \text{if } s_i \notin \mathcal{T} \\ -1 & \text{if } s_i \in \mathcal{T}, y \neq c_g \end{cases}$$

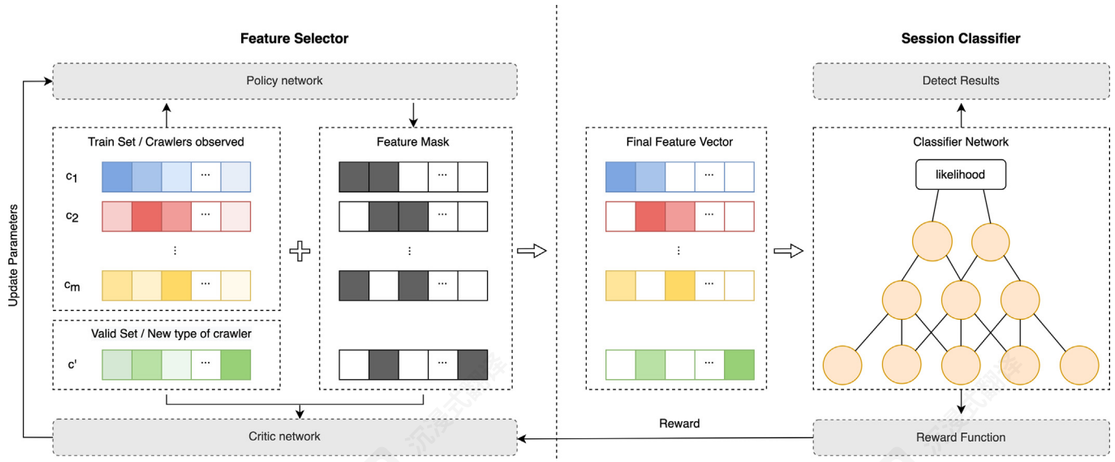


Fig. 2. Overview of model. The training set contains several types of crawlers, and these crawlers will get different feature masks after the policy network and then get different feature sets. The processed feature vectors are input to the session classifier to get the detection results. The reward function calculates the reward feedback to the critic network and then optimizes the policy network to produce a better feature set. After training, the new type of crawler, which is not included in the training set, is used to detect the robustness of the model.

where $\mathcal{T}$ is the terminal state, $N\prime$ denotes the number of discarded features, N denotes the number of all features, and α is a hyper-parameter to balance the two terms.

Based on the above definition of reinforcement learning model elements, our method uses the architecture of Deep Deterministic Policy Gradient(DDPG), which is an actor-critic based algorithm in reinforcement learning [37]. DDPG use a policy network to generate the actions of the agent. The Critic network can judge the quality of actions and guide the updating direction of policy function. We will introduce the policy network and the critic network in the following.

- **Policy Network**
  In our method, the policy network consists of several fully connected layers and the policy function is formulated as follows:

$$\mu(s_i|\theta^{\mu}) = \sigma(W_{\mu} * s_i + b_{\mu}) \tag{2}$$

  where $\sigma(\cdot)$ is sigmoid function with the parameter of policy network$\theta^{\mu} = \{W_{\mu}, b_{\mu}\}$. The policy network takes the feature distribution of the session (state) as input and outputs the feature mask (action) corresponding to the input. We use the sigmoid function to deflate the mask to (0, 1), representing the weights assigned to each feature. To balance the exploration and exploitation problem in reinforcement learning, we use the Uhlenbeck-Ornstein stochastic process as the random noise.
- **Critic Network**
  DDPG uses a neural network to model the Q function like Deep Q Network (DQN), so the Q function can be formulated as follows:

$$Q(s_i, a_i|\theta^{Q}) = W_Q * F(s_i, a_i) + b_Q \tag{3}$$

  where $F(s_i, a_i)$ is a vector that combines state s_i and action a_i with the parameter of critic network $\theta^{Q} = \{W_Q, b_Q\}$. The critic network takes the combined vector of states and actions as input and outputs the Q value of the action corresponding to the current state.
- **Optimization**
  The previous practice has shown that if only a single Q neural network algorithm is used, the learning process is volatile because the parameters of the Q network are used to calculate the gradient of the critic network and the policy network, while the gradient update is frequent, based on this, DDPG creates two copies of neural networks for the policy network and critic network respectively, called target network. Thus, our feature selector consists of 4 networks: policy network, critic network, target policy network, and target critic network.
  For the policy network, we use a function J to measure the performance of the policy $\mu(s_i|\theta^{\mu})$, which we call the performance objective, defined as follows:

$$J_{\beta}(\mu) = \int_S \rho^{\beta}(s) Q^{\mu}(s, \mu(s)) ds \tag{4}$$

  where s is the state which has the distribution function $\rho^{\beta}, Q^{\mu}(s, \mu(s))$ is the Q value that generated in each state if acton $\mu(s)$ is selected in accordance with the policy $\mu(s_i|\theta^{\mu})$. The goal of our training is to maximize $J_{\beta}(\mu)$. According to the mathematical derivation of the Deterministic Policy Gradient(DPG) [38], the policy gradient can be formulated as follows:

$$\bigtriangledown_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \bigtriangledown_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \cdot \bigtriangledown_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_i} \tag{5}$$

  where N is the number of mini-batch data, and this is an unbiased estimate of policy gradient with the method of Monte-Carlo.
  For the critic network, in order to get a more accurate Q value, we use a supervised learning-like approach to calculate its gradient with Mean Square Error(MSE) loss:

$$L_Q = \frac{1}{N} \sum_i (r_i + \gamma Q\prime(s\prime_i, \mu\prime(s\prime_i|\theta^{\mu\prime})|\theta^{Q\prime}) - Q(s_i, a_i|\theta^{Q}))^2 \tag{6}$$

  where r_i is the reward, $Q\prime$ is the target critic network, $\mu\prime$ is the target policy network, $s\prime_i$ is the next state of s_i and $\gamma \in [0, 1]$ is discounted rate.
  For the update of the target policy network and target critic network, we use a soft update algorithm as follows:

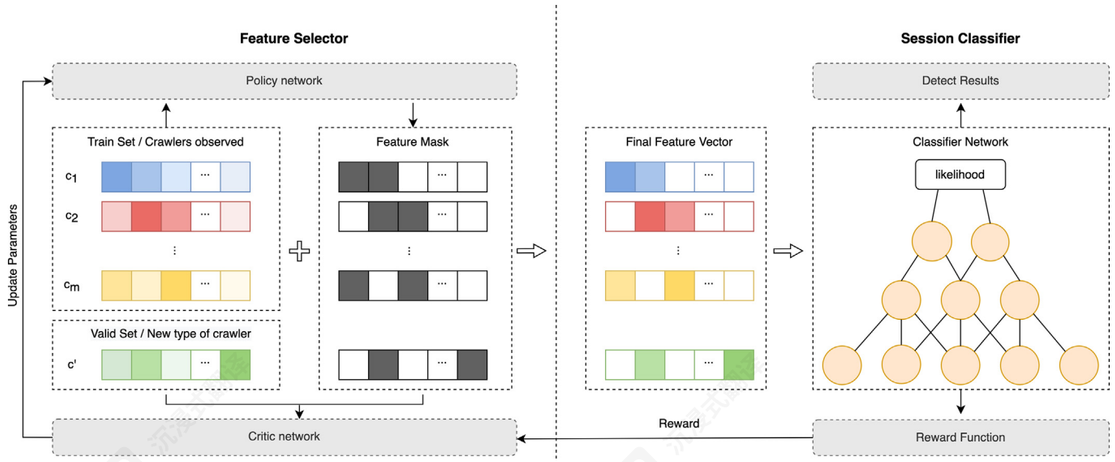


图 2. 模型概述。训练集包含几种类型的爬虫，这些爬虫在策略网络之后将获得不同的特征掩码，然后获得不同的特征集。处理后的特征向量输入到会话分类器以获得检测结果。奖励函数计算对评论家网络的奖励反馈，然后优化策略网络以产生更好的特征集。训练后，使用训练集中未包含的新类型爬虫来检测模型的鲁棒性。

其中 T 是终止状态，N\prime 表示丢弃的特征数量，N 表示所有特征的数量，而 α 是一个超参数，用于平衡两个项。

基于上述强化学习模型元素的定义，我们的方法使用深度确定性策略梯度（ DDPG ）的架构，这是一种基于演员 - 评论家的强化学习算法 [37]。DDPG 使用策略网络来生成智能体的动作。评论家网络可以判断动作的质量并指导策略函数的更新方向。以下我们将介绍策略网络和评论家网络。

- 策略网络
  在我们的方法中，策略网络由几个全连接层组成，策略函数如下所示：

$$\mu(s_i|\theta^{\mu}) = \sigma(W_{\mu} * s_i + b_{\mu}) \tag{2}$$

  其中 $\sigma(\cdot)$ 是具有参数 policy networkh$_{\mu}$ = {W$_{\mu}$;b$_{\mu}$} 的 sigmoid 函数。策略网络以会话（状态）的特征分布作为输入，并输出与输入相对应的特征掩码（动作）。我们使用 sigmoid 函数将掩码压缩到（ 0，1 ），表示分配给每个特征的权重。为了平衡强化学习中的探索和利用问题，我们使用 Uhlenbeck-Ornstein 随机过程作为随机噪声。
- 评论家网络
  DDPG 使用神经网络来模拟 Q 函数，类似于深度 Q 网络（ DQN ），因此 Q 函数可以表示如下：

$$Q(s_i, a_i|\theta^{Q}) = W_Q * F(s_i, a_i) + b_Q \tag{3}$$

  其中 F(s$_i$;a$_i$) 是一个向量，它结合了状态 s$_i$ 和动作 a$_i$ 以及评论网络参数 h^{Q} = {W$_Q$; b$_Q$}。评论网络将状态和动作的组合向量作为输入，并输出对应于当前状态的 Q 值。
- 优化
  之前的实践表明，如果只使用单个 Q 神经网络算法，学习过程是波动的，因为 Q 网络的参数用于计算评论网络和政策网络的梯度，而梯度更新是频繁的。基于此，DDPG 为策略网络和评论网络分别创建了两个副本，称为目标网络。因此，我们的特征选择器由 4 个网络组成：策略网络、评论网络、目标策略网络和目标评论网络。

对于策略网络，我们使用一个函数 J 来衡量策略 μ(s$_i$|h$_{\mu}$) 的性能，我们称之为性能目标，定义如下：

$$J_{\beta}(\mu) = \int_S \rho^{\beta}(s) Q^{\mu}(s, \mu(s)) ds \tag{4}$$

其中 s 是状态，它具有分布函数 ρ^{β};Q$_{\mu}$(s; μ(s)) 是在每个状态下如果根据策略 μ(s) h$_{\mu}$(s$_i$|h$_{\mu}$) 选择动作 $_{\beta}$(l) 时生成的 Q 值。我们训练的目标是最大化 J[38], {{v27}}{{v28}}{{v29}}。根据确定性策略梯度（ DPG ）的数学推导，策略梯度可以表示如下：

$$\bigtriangledown_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \bigtriangledown_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \cdot \bigtriangledown_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_i} \tag{5}$$

其中 N 是迷你批数据数量，这是一个基于蒙特卡洛方法的非偏政策梯度估计。

对于评论家网络，为了获得更准确的 Q 值，我们采用类似于监督学习的方法，使用均方误差（ MSE ）损失来计算其梯度：

$$L_Q = \frac{1}{N} \sum_i (r_i + \gamma Q\prime(s\prime_i, \mu\prime(s\prime_i|\theta^{\mu\prime})|\theta^{Q\prime}) - Q(s_i, a_i|\theta^{Q}))^2 \tag{6}$$

其中 r$_i$ 是奖励，Q\prime 是目标评论家网络，$\mu\prime$ 是目标策略网络，s\prime$_i$ 是 s$_i$ 和 $\gamma \in [0; 1]$ 是折扣率。

对于目标策略网络和目标评论家网络的更新，我们使用以下软更新算法：

$$\theta^{\mu\prime} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu\prime} \tag{7}$$

$$\theta^{Q\prime} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q\prime} \tag{8}$$

where τ is the soft update rate.

3.4. Session classifier

In the session classifier, we adopt a Deep Neural Network(DNN) architecture to predict the class of session. The DNN network has an input layer, some fully connected layers and a non-liner layer from which the representation is used for session classification.

Input Layer
When a session is input to the feature selector, we will get two vectors: one is the current feature distribution vector of the session formulated as $s_i = \{f_1, f_2, \ldots, f_n\}$, the other is its feature mask formulated as $a_i = \{m_1, m_2, \ldots, m_n\}$. We multiply the corresponding positions of these two vectors to obtain a new feature distribution vector $s\prime_i = \{f\prime_1, f\prime_2, \ldots, f\prime_n\}$, where the value of retained features will remain unchanged, and the value of discarded features will become 0. We use this new feature distribution $s\prime_i$ as the input to DNN for classification.

Deep Neural Network
We found that few scholars have used DNNs directly for crawler detection through our research. Here we use five fully connected layers to map the feature distribution to a 2-dimensional vector and use softmax activation to represent the probability that a session is a crawler or a user. The probability for session classification $P(c|s\prime_i, \theta^C)$ is given as follows:

$$P(c|s\prime_i, \theta^C) = softmax(W_C * s\prime_i + b_C) \tag{9}$$

where $\theta^C = \{W_C, b_C\}$ is the parameter of session classifier.

Loss Function
Given the mini-batch training set S with N sessions, we define the objective function of the session classifier using cross-entropy as follows:

$$L_C = -\frac{1}{N}\sum_i logP(c_g|s\prime_i, \theta^C) \tag{10}$$

Model Train
In our approach, the result of the session classifier will affect the policy of the feature selector as a reward, and the result of the selector will affect the classification accuracy of the classifier, so the two modules should be trained jointly. The training process is described in Algorithm 1. To make the model converge as soon as possible and avoid unnecessary exploration, we need to pre-train the session classifier first, which is viewed as the environment of reinforcement learning, so that it can give relatively reliable rewards. We use a random process to generate some feature masks to retain and discard features. To make the training data more reliable and ensure a wide range of selection, we generate ten feature masks for each session. Each feature mask discards features according to different ratios, satisfying 0.0, 0.1, …, and 0.9 incremental rules. For example, ratio 0.0 means retaining all the features, and 0.9 means discarding 90 percent of the features. Experiments show that using such a pre-training approach, the session classifier can return a relatively reliable reward for each level of feature selection. Then, we fix the parameters of the session classifier and train the feature selector. At last, when the feature selector performs stably in the current environment, we jointly train the two modules.

Algorithm 1: Overall Training Process

1: Initialize the parameters of feature selector and session classifier with random weights respectively.
2: Pre-train the session classifier with random feature mask by minimizing Eq. (10)
3: Fix the parameters of session classifier and pre-train the feature selector by running Algorithm 2.
4: Train the two modules jointly until convergence.

The training process of the reinforcement learning model is shown in Algorithm 2. For each session in the training set, if the number of feature selection steps does not reach the upper limit N_t or the current feature distribution is insufficient to distinguish the session, which means it does not enter the terminal state, we will keep modifying its feature mask. When there is enough data in the replay buffer, we will sample a random mini-batch data to train the feature selector each time, and we will also train the session classifier if it is joint training. At the same time, we update the target network using soft update.

Algorithm 2: Reinforcement Learning Training Process

Input: Episode number N_e. Training data $S = \{s_1, s_2, \ldots, s_n\}$. A session classifier model C with parameters θ^C. A policy network μ with parameters θ^{μ} and a critic network Q with parameters θ^{Q}.
1: Initialize the parameters θ^{μ} and θ^{Q}.
2: Initialize the replay buffer M
3: Copy to get two target networks $\mu\prime$ and $Q\prime$ and initialize their parameters as: $\theta^{\mu\prime} = \theta^{\mu}, \theta^{Q\prime} = \theta^{Q}$.
4: **for** *episode* $e = 1$ *to* N_e **do**
5: **for** $s_i \in S$ **do**
6: **for** *step* $t = 1$ *to* N_t **do**
7: Get an action a_i with policy network μ according to Eq. (1)
8: State Transition: $s\prime_i \leftarrow s_i \times a_i$
9: Get reward from session classifier: $r_i \leftarrow C(s\prime_i)$
10: Add transition $(s_i, a_i, r_i, s\prime_i)$ into replay buffer M
11: **if** $s\prime_i$ is terminal state **then**
12: **break**
13: **end if**
14: $s_i = s\prime_i$ 15: **end for**
16: Sample a random mini-batch B from M
17: **for all** $(s_i, a_i, r_i, s\prime_i) \in B$ **do**
18: Compute the loss of policy network and critic network respectively according to Eq. (5) and Eq. (6)
19: Update the parameters θ^{μ} and θ^{Q}
20: **if** *training jointly* **then**
21: Compute the loss of session classifier according to Eq. (10).
22: Update the parameters θ^{C}
23: **end if**
24: Soft update the parameters $\theta^{\mu\prime}$ and $\theta^{Q\prime}$ according to Eq. (7) and Eq. (8)
25: **end for**
26: **end for**
27: **end for**

$$\theta^{\mu\prime} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu\prime} \tag{7}$$

$$\theta^{Q\prime} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q\prime} \tag{8}$$

其中是软更新率。

3.4. 会话分类器

在会话分类器中，我们采用深度神经网络（ DNN ）架构来预测会话的类别。DNN 网络包含一个输入层，一些全连接层以及一个非线性层，该层的表示用于会话分类。

输入层
当会话输入到特征选择器时，我们将得到两个向量：一个是会话当前特征分布向量，表示为 $s_i = \{f_1; f_2; \ldots; f_n\}$；另一个是其特征掩码，表示为 $a_i = \{m_1; m_2; \ldots; m_n\}$。我们将这两个向量的对应位置相乘，得到一个新的特征分布向量 $s\prime_i = \{f0_1; f0_2; \ldots; f\prime_n\}$，其中保留的特征值将保持不变，而丢弃的特征值将变为 0。我们使用这个新的特征分布 $s\prime_i$ 作为 DNN 分类的输入。

深度神经网络
我们发现，通过我们的研究，很少有学者直接使用 DNNs 进行爬虫检测。在这里，我们使用五个全连接层将特征分布映射到二维向量，并使用 softmax 激活函数来表示一个会话是爬虫还是用户的概率。会话分类概率 $P(c|s\prime_i; h^C)$ 如下给出：

$$P(c|s\prime_i, \theta^C) = softmax(W_C * s\prime_i + b_C) \tag{9}$$

$h^C = \{W_C; b_C\}$ 是会话分类器的参数。
损失函数 **ion**
给定具有 N 个会话的迷你批训练集 S，我们使用交叉熵定义会话分类器的目标函数如下：

$$L_C = -\frac{1}{N}\sum_i logP(c_g|s\prime_i, \theta^C) \tag{10}$$

模型火车
在我们的方法中，会话分类器的结果将作为奖励影响特征选择器的策略，而选择器的结果将影响分类器的分类准确率，因此这两个模块应该联合训练。训练过程在算法 1 中描述。为了使模型尽快收敛并避免不必要的探索，我们首先需要预训练会话分类器，这被视为强化学习环境，以便它能给出相对可靠的奖励。我们使用随机过程生成一些特征掩码来保留和丢弃特征。为了使训练数据更可靠并确保广泛的选取范围，我们为每个会话生成十个特征掩码。每个特征掩码根据不同的比率丢弃特征，满足 0.0 、 0.1 、 …和 0.9 的增量规则。例如，比率 0.0 表示保留所有特征，而 0.9 表示丢弃 90% 的特征。实验表明，使用这种预训练方法，会话分类器可以为每个特征选择级别返回相对可靠的奖励。然后，我们固定会话分类器的参数并训练特征选择器。最后，当特征选择器在当前环境中表现稳定时，我们联合训练这两个模块。

算法 1：整体训练过程

1: 分别用随机权重初始化特征选择器和会话分类器的参数。sion 分类器。
2: 使用随机特征预训练会话分类器。 mask 通过最小化等式 (10)
3: 修复会话分类器和预训练的参数 in the 通过运行算法 2 进行特征选择。
4: 联合训练两个模块，直到收敛。

强化学习模型的训练过程如图 2 所示。对于训练集中的每个会话，如果特征选择步骤的数量没有达到上限 N_t 或者当前特征分布不足以区分会话，这意味着它没有进入终止状态，我们将继续修改其特征掩码。当重放缓冲区中有足够的数据时，我们将每次随机采样一个迷你批数据来训练特征选择器，如果进行联合训练，我们还将训练会话分类器。同时，我们使用软更新来更新目标网络。

算法 2：强化学习训练过程

输入：第 N_e 集。训练数据 S = {s1； s2； … ,s_n}. A 会话分类模型 C，参数为 h_c。一个 icy 网络版本 μ，具有参数 h_μ 和批评网络 Q with 参数 h_Q。
1: 初始化参数 h_μ 和 h_Q。
2: 初始化重放缓冲区 M
3: 复制以获取两个目标网络 $\mu\prime$ 和 $Q\prime$ 以及初始化 ialize 他们的参数为：$h_{\mu\prime} = h_\mu$； $h^{Q\prime} = h_Q$。
4: 对于第 e = 1 到 N_e 个片段
5: 对于 s_i ∈S
6: 对于步骤 t = 1 到 N_t 执行
7: 使用策略网络 μ 获取动作 a_i ding to 等式 (1)
8: 状态转移：$s\prime_i \leftarrow s_i \times a_i$
9: 从会话分类器获取奖励：$r_i \leftarrow C(s\prime_i$)
10: 添加过渡（s_i； a_i； r_i； $s\prime_i$）到回放缓冲区 r M
11: 如果 if$s\prime_i$ 是终止状态，则
12: **break**
13: **end if**
14: $s_i = s\prime_i$ 15: 结束循环
16: 从 M 中采样一个随机的小批量 B
17: 对所有（s_i； a_i； r_i； $s\prime_i$）∈ B 执行
18: 计算策略网络的损失和 cri tic 网络分别根据公式 (5) 和公式 (6))
19: 更新参数 h_μ 和 h^Q
20: 如果联合训练，则
21: 计算会话分类器的损失 acc ording 到方程式 (10)。
22: 更新参数 h^C
23: **end if**
24: Soft update the parameters $\theta^{\mu\prime}$ and $\theta^{Q\prime}$ according to Eq. (7) and Eq. (8)
25: **end for**
26: **end for**
27: **end for**

4. Experiments

In this section, some experiments have been conducted over two real datasets to validate the proposed approach. The first is our own real dataset, which will be described in detail later, and the second is a public dataset from the search engine of the library and information center of the Aristotle University of Thessaloniki in Greece. We first demonstrated the existence of the diversity of crawlers in these two datasets experimentally. Then, to prove that our method can deal with the dynamic change of crawler, we compared our method with some baselines regarding the accuracy of the new type of crawler detection. We also did experiments on the accuracy of all types of crawlers detection. Finally, we looked at the effect of the number of the selected feature on the detection accuracy, and the experimental results and analysis will be given in the following text.

4.1. Datasets

4.1.1. Private dataset

Our dataset comes from the faculty personal home page portal of Zhejiang University. The site allows users to search faculty members' homepages, which contain personal information, photos, research directions, academic achievements, etc., based on their names or departments. We use the access log data of this website, and the time spans from June 22, 2020, to June 28, 2020, with a total of 2809702 requests.

- **Session Identification**

Our session identification process works as follows. First, we group requests with the same IP address and user-agent string. In addition, we remove some requests with empty IP or empty user-agent. Then, we apply a timeout threshold to break the groups into sessions. In this paper, we set the timeout threshold to 30 min as in most studies. This process identified 56544 sessions which have an average of 49.69 requests.

With a lot of research in the field of crawler detection, we selected 24 previously published features based on the characteristics of our dataset, the names and descriptions of the features are shown in Table 1.

- **Session Labeling**

The session labeling process is a very complex task. To achieve a higher quality of labeling, we use multiple dimensions to label the session. First, we use a dataset from GitHub, which summarizes the common crawler user-agent [39]. It is a JSON file, and if the user-agent of our data is included in the JSON file, it will be labeled as a crawler; otherwise, it is temporarily labeled as user data.

Second, we checked if the session contains a request to the *robots.txt* and a session will be labeled as a crawler if it contains such a request. This method is a standard labeling process because there are no external or internal links of *robots.txt* and only the crawler can access it.

Third, We define some regular expressions to detect the user-agent, if the user-agent contains keywords such as a bot, spiders, spam, etc., it will be marked as a crawler.

Finally, we define some features to label the crawlers regarding previous studies. For sessions where all requests do not access images, do not have a referrer, fail, or all requests are of HEAD type, we label them as crawlers. For the sessions that are not labeled as crawlers in any of the above dimensions, we label them as a user. We ended up with a private dataset containing 39581 user data and 16963 crawler data.

Table 1
Universal extracted features of private dataset.

ID	Feature name	Description
1	IS_TRAP_FILE	Whether to access trap file such as robots. txt.
2	NIGHT_RATIO	Percentage of requests made between 12am and 7am.
3	IMAGE_RATIO	Percentage of image file requests.
4	HTML_RATIO	Percentage of html file requests.
5	REFERRER_RATIO	Percentage of requests with unassigned referrer.
6	HEAD_RATIO	Percentage of requests of type HEAD.
7	304_RATIO	Percentage of requests with status code 304.
8	ERROR_RATIO	Percentage of erroneous requests.
9	ERROR_UPSTREAM_RATIO	Percentage of requests with empty upstream status.
10	SESSION_TIME	Total time elapsed between the first and the last request.
11	AVERAGE_INTERVAL	Average time between two consecutive requests.
12	DEVIATION_INTERVAL	Standard deviation of the time between two consecutive requests.
13	REQUESTS_NUMBER	The total number of requests.
14	UNIQUE_TYPE	The total number of file type of requests.
15	MAX_BROWSER_FILE_RATE	Maximum number of embedded resources in a web page.
16	PENALTY	Penalty for each backward and forward navigation or loop.
17	SD_RPD	Standard deviation of the page depth across all requests.
18	CSR	Percentage of requests with continuous access to the page belong to the same directory.
19	RES	Average response time to requests.
20	SF-FILE_TYPE	Switching factor of file types for each session.
21	SF-REFERRER	Switching factor on unassigned referrer.
22	WIDTH	The number of leaf nodes generated in the graph of all requests.
23	DEPTH	The maximum depth of the tree within the graph of all requests.
24	TOTA_PAGE	The total number of pages requested.

4.1.2. Public dataset

This dataset contains the server logs of the search engine of the Library and Information Center of the Aristotle University of Thessaloniki, Greece (http://search.lib.auth.gr/). Users can use this search engine to check the availability of books and other written works and search for digitized materials and scientific publications. This dataset contains access logs for an entire month from March 1 to March 31, 2018, including 4,091,155 requests. Through the publisher's session identification process and the labeling process, the access logs were divided into 67,352 sessions containing 53,858 user data and 13,494 crawler data.

4.2. Multi-type crawler analysis

This paper is based on the assumption that there are multi-types crawlers in the web environment and different types of crawlers have different characteristics and distributions in the feature space. To prove that our method can select the appropriate feature set for different types of crawlers, we should first verify whether the dataset satisfies the assumption of multi-types crawlers. Since both private and public datasets have only two types of labels, user or crawler, and there is no multi-type label for crawlers, we first use the K-means method to cluster the datasets and then use shap-value to analyze the differences in feature distributions between the clustering results. In addition, to present the

4. 实验

在本节中，我们在两个真实数据集上进行了实验，以验证所提出的方法。第一个是我们自己的真实数据集，将在后面详细描述，第二个是来自希腊萨洛尼卡亚里士多德大学图书馆和信息中心搜索引擎的公共数据集。我们首先通过实验证明了这两个数据集中爬虫的多样性。然后，为了证明我们的方法可以处理爬虫的动态变化，我们将我们的方法与一些基线方法在新型爬虫检测的准确性方面进行了比较。我们还对各种类型爬虫检测的准确性进行了实验。最后，我们考察了所选特征数量对检测准确性的影响，实验结果和分析将在下文中给出。

4.1. 数据集

4.1.1. 私有数据集

我们的数据集来自浙江大学教师个人主页门户。该网站允许用户根据姓名或部门搜索教师主页，这些主页包含个人信息、照片、研究方向、学术成就等。我们使用该网站的访问日志数据，时间跨度从 2020 年 6 月 22 日到 2020 年 6 月 28 日，共计 2809702 次请求。

5. 会话识别

我们的会话识别过程如下。首先，我们将具有相同 IP 地址和用户代理字符串的请求分组。此外，我们删除了一些 IP 地址或用户代理为空的请求。然后，我们应用超时阈值将组划分为会话。在本文中，我们将超时阈值设置为 30 分钟，这与大多数研究一致。这个过程识别了 56544 个会话，平均每个会话有 49.69 个请求。

在爬虫检测领域进行了大量研究后，我们根据我们的数据集的特征、特征名称和描述选择了 24 个先前发布的特征，这些特征名称和描述如表 1 所示。

会话标注

会话标注过程是一个非常复杂的任务。为了实现更高的标注质量，我们使用多个维度来标注会话。首先，我们使用 GitHub 上的数据集，该数据集总结了常见的爬虫用户代理。它是一个 JSON 文件，如果我们的数据中的用户代理包含在 JSON 文件中，它将被标注为爬虫；否则，它将暂时被标注为用户数据。

其次，我们检查会话中是否包含对 robots.txt 的请求，如果包含此类请求，则将此会话标注为爬虫。这种方法是一种标准的标注过程，因为没有外部或内部链接指向 robots.txt，只有爬虫可以访问它。

第三，我们定义了一些正则表达式来检测用户代理，如果用户代理包含诸如 bot 、 spiders 、 spam 等关键词，它将被标记为爬虫。

最后，我们根据以往的研究定义了一些特征来标记爬虫。对于所有请求都不访问图像、没有引用、失败或所有请求都是 HEAD 类型的会话，我们将其标记为爬虫。对于在任何上述维度上未标记为爬虫的会话，我们将其标记为用户。我们最终得到了一个包含 39581 个用户数据和 16963 个爬虫数据的私有数据集。

表 1 私有数据集的通用提取特征。

ID	特征名称	描述
1	IS_TRAP_FILE	是否访问此类陷阱文件 s robots. txt.
2	NIGHT_比率	请求百分比 ween 凌晨 12 点和早上 7 点。
3	图片 _比例	图像文件请求百分比 s.
4	HTML 文件请求比例	HTML 文件请求百分比
5	引用者 _比例	带有 una 请求的百分比 ssigned 引用者。
6	HEAD_比例	类型 H 请求的百分比 EAD.
7	304_ 比例	带有统计信息的请求百分比 us code 304.
8	错误 _比率	错误请求的百分比 s.
9	错误 _上游 _比率	具有 emp 的请求数百分比 ty 上游状态。
10	会话 _时间	两个事件之间的总时间流逝 first and 最后的请求。
11	平均 _间隔	两次连续之间的平均时间 ecutive 请求。
12	偏差 _间隔	时间 b 的标准差 etween 两次连续请求
13	请求 _数量	The total number of 请求。
14	唯一 _类型	文件类型的总数为 requests.
15	MAX_浏览器 _文件 _速率	嵌入的最大数量 网页中的资源。
16	罚款	每次后退和 f 的罚款 orward 导航或循环。
17	SD_RPD	页面长度的标准差 epth 对所有请求而言。
18	CSR	请求百分比 sts with continuous 访问页面属于 same 目录。
19	RES	请求的平均响应时间 ts.
20	SF-FILE_类型	针对文件类型的切换因子 each 会话。
21	SF- 引用者	未分配 r 上的切换因子 eferrer.
22	宽度	生成的叶节点数量 ed in the 所有请求的图
23	深度	树的深度最大值 within 所有请求的图
24	总页数	总页数 requested.

4.1.2. 公共数据集

该数据集包含希腊萨洛尼卡亚里士多德大学图书馆和信息中心搜索引擎的服务器日志（ http://search.lib.auth.gr/ ）。用户可以使用此搜索引擎检查书籍和其他书面作品的可用性，并搜索数字化材料和科学出版物。该数据集包含 2018 年 3 月 1 日至 3 月 31 日的整个月的访问日志，包括 4,091,155 个请求。通过出版商的会话识别过程和标记过程，访问日志被划分为包含 53,858 个用户数据和 13,494 个爬虫数据的 67,352 个会话。

4.2. 多类型爬虫分析

本文基于网络环境中存在多类型爬虫的假设，并且不同类型的爬虫在特征空间中具有不同的特征和分布。为了证明我们的方法可以为不同类型的爬虫选择合适的特征集，我们首先需要验证数据集是否满足多类型爬虫的假设。由于私有和公共数据集都只有两种标签，即用户或爬虫，并且没有为爬虫设置多类型标签，我们首先使用 K-means 方法对数据集进行聚类，然后使用 shap-value 分析聚类结果之间的特征分布差异。此外，为了展示

clustering results intuitively, we use the TSNE method to downscale the clustering results so that they have only two-dimensional features.

Parameter Setting: Through extensive experiments, for the parameters of the private dataset, we set the cluster number as 7, random state as 12; for the parameters of the public dataset, we set the cluster number as 6, random state as 12.

Result: The clustering results of the private dataset are shown in Fig. 3, where the green part is the user data, and the other colors are the crawler data. Although using the TSNE method to reduce the 24-dimensional feature space to a 2-dimensional space will cause an inevitable error, the individuals with similar feature spaces are still close together in the figure, and here we set the number of clusters to 7 to make each class basically contain only one kind of label and minimize the number of clusters.

We take out the six clustering results belonging to the crawler and perform a binary classification task with the user data respectively. To analyze the importance of different features in the classification process, here we used the xgboost classifier and analyzed the shap-value of different features using the shap toolkit. The analysis results are shown in Fig. 4. From the results, we can see that different clustering results have different feature importance when classifying with user data. Take the second and fourth categories for comparison, where the image ratio of the second category crawler has a significant difference with the user data, so the IMAGE_RATIO feature has a high shap-value, and the use of this feature can better distinguish crawler from user data. The fourth type of crawler has a significant difference between the TOTAL_PAGE with the user data, but on the contrary, there is basically no difference in the IMAGE_RATIO, so the shap-value of the TOTAL_PAGE is high, while the shap-value of the IMAGE_RATIO is very low. For this type of crawler, using the feature of TOTAL_-PAGE can better distinguish crawler from user data.

The clustering results of the public dataset are shown in Fig. 5, where the blue part is the user data, and the other colors are the crawler data. From the clustering results, we can still see that different types of crawlers have apparent differences in feature distribution. Similarly, we analyzed the different crawler clustering results with users for feature importance, and the results are shown in Fig. 6. The results show that the features that have the highest shap-value vary for each type of crawler. The first type of crawler differs from the user data in the TOTAL_HTML feature, and it should be a crawler that only visits html files. In contrast, the second type of crawler differs from the user data in the IMAGES feature, and it should be an image crawler that only crawls images as mentioned before. The fourth type of crawler differs from the user data in the NIGHT feature, and it should be a crawler that only visits the website at night.

From the above experiments and analysis, we can conclude the following: firstly, the datasets we used have different types of crawlers. Secondly, different crawlers have different characteristics and distributions in the feature space.

Therefore, we can improve the accuracy of crawler detection by selecting different feature sets for multi-type crawlers.

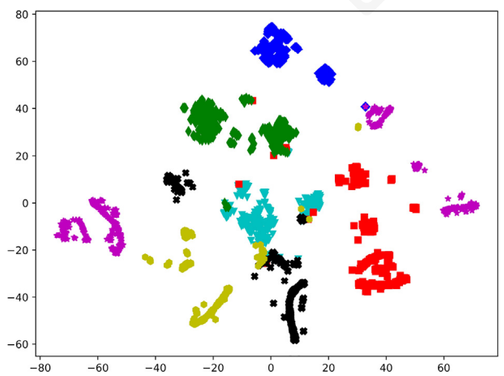


Fig. 3. Clustering results of the private dataset. The green part is the user data, and the other colors are the crawler data.

4.3. Evaluation metrics

To show the effectiveness of the proposed method, we use the following metrics for evaluation. In addition to the commonly used recall, precision, F1-score and accuracy for classification tasks, we also add an evaluation metric for the white sample mishap, because the crawler detection problem has to ensure a low error rate in the actual business, which is responsible for seriously affecting the user's online experience.

- Recall: Given by $\frac{TP}{TP+FN}$, which represents the ratio of true positive to true positive plus false negative.
- Precision: Given by $\frac{TP}{TP+FP}$, which represents the ratio of true positive to true positive plus false positive.
- Error: Given by $\frac{FP}{FP+TN}$, which represents the ratio of false positive to false positive plus true negative.
- Accuracy: Given by $\frac{TP+TN}{TP+FP+TN+FN}$, which represents the ratio of true prediction to all the data.
- F1-score: Given by $\frac{2*P*R}{P+R}$, which considers both precision and recall in a single metric by taking their harmonic mean, where P and R are precision and recall respectively.

4.4. New type crawler detection

As mentioned before, crawlers have problems with diversity and dynamic change, the web environment is complex and volatile, and the emergence of new types of crawlers can make pre-trained models less robust. To check the robustness of our method in the face of new types of crawlers, we do the following experiments. Each time, we extract one type of crawler from the clustering results and mark it as a new type crawler. This part of data is not involved in the training process. After the model is trained, we test the robustness of the model by forming a test data set with user data and new type crawler, and observe the crawler detection effect of the model on this test set. For the private dataset, we have 6 types of crawlers, and we did experiments for each type separately, for a total of 6 sets of experiments. Similarly, for the public dataset we did 5 sets of experiments. We compare our method with some previous models that perform well.

Parameter Setting: For the parameters of the feature selector, we set the train episodes as 10, the step number of the agent as 10, learning rate of the policy network as 5e-6, learning rate of the critic network as 1e-5, gamma as 0.9, soft update rate as 0.01, feature selecting threshold as 0.4/0.6, terminal state threshold as 0.9, size of replay buffer as 5000, alpha as 10, batch size as 32, the delay coefficient τ as 0.01. For the parameters of the session classifier, we set the learning rate as 1e-3, and batch size as 32.

Baselines:

- DTMC: This method is based on a first-order discrete time Markov Chain model, which represents robot and human resource request patterns. They use this model to compute the probability a request pattern is more likely generated by a crawler or a user [10].

通过直观地聚类结果，我们使用 TSNE 方法对聚类结果进行降维，使其只有二维特征。

参数设置：通过大量实验，对于私有数据集的参数，我们将聚类数设置为 7，随机状态设置为 12；对于公共数据集的参数，我们将聚类数设置为 6，随机状态设置为 12。

结果：如图 3 所示，私有数据集的聚类结果，其中绿色部分是用户数据，其他颜色是爬虫数据。虽然使用 TSNE 方法将 24 维特征空间降低到二维空间会导致不可避免的误差，但具有相似特征空间的个体在图中仍然彼此靠近，因此我们设置聚类数为 7，以便每个类别基本上只包含一种标签，并最小化聚类数。

我们取出属于爬虫的六个聚类结果，分别与用户数据进行二元分类任务。为了分析不同特征在分类过程中的重要性，这里我们使用了 xgboost 分类器，并使用 shap 工具包分析了不同特征的 shap 值。分析结果如图 4 所示。从结果中我们可以看到，不同的聚类结果在用用户数据进行分类时具有不同的特征重要性。以第二类和第四类爬虫进行比较，第二类爬虫的图像比率与用户数据有显著差异，因此 IMAGE _RATIO 特征具有高 shap 值，使用此特征可以更好地区分爬虫和用户数据。第四类爬虫与用户数据的 TOTAL_PAGE 有显著差异，但相反，在 IMAGE_RATIO 上基本上没有差异，因此 TOTAL_PAGE 的 shap 值很高，而 IMAGE_RATIO 的 shap 值非常低。对于此类爬虫，使用 TOTAL_- PAGE 的特征可以更好地区分爬虫和用户数据。

公共数据集的聚类结果如图 5 所示，其中蓝色部分是用户数据，其他颜色是爬虫数据。从聚类结果中，我们仍然可以看出不同类型的爬虫在特征分布上有明显的差异。同样，我们分析了不同爬虫聚类结果与用户特征的重要性，结果如图 6 所示。结果显示，具有最高 shap-value 的特征因爬虫类型而异。第一种爬虫与用户数据在 TOTALHTML 特征上有所不同，它应该是一种只访问 HTML 文件的爬虫。相比之下，第二种爬虫与用户数据在 IMAGES 特征上有所不同，它应该是一种如前所述只爬取图像的图像爬虫。第四种爬虫与用户数据在 NIGHT 特征上有所不同，它应该是一种只在夜间访问网站的爬虫。

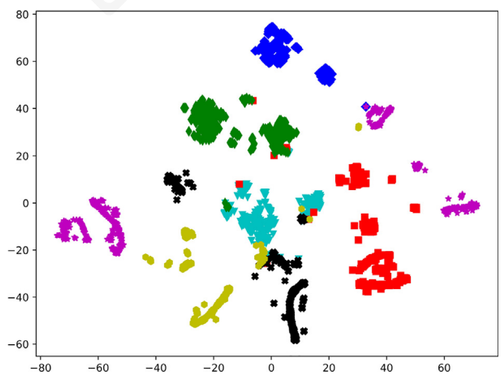


图 3. 私有数据集的聚类结果。绿色部分是用户数据，其他颜色是爬虫数据。

特征，它应该是一种只爬取图像的图像爬虫。第四种爬虫与用户数据在 NIGHT 特征上有所不同，它应该是一种只在夜间访问网站的爬虫。

从上述实验和分析中，我们可以得出以下结论：首先，我们使用的数据集中有不同类型的爬虫。其次，不同爬虫在特征空间中具有不同的特征和分布。

因此，我们可以通过为多类型爬虫选择不同的特征集来提高爬虫检测的准确性。

4.3. 评估指标

为了展示所提出方法的有效性，我们使用以下指标进行评估。除了分类任务中常用的召回率、精确率、 F1 分数和准确率之外，我们还增加了一个对白样本误报的评估指标，因为爬虫检测问题必须确保在实际业务中低错误率，这会严重影响用户的在线体验。

召回率：由 $\frac{TP}{TP+FN}$ 给出，表示真实正例与真实正例加假负例的比率。

- 精确率：由 $\frac{TP}{TP+FP}$ 给出，表示真实正例与真实正例加假正例的比率。
- 错误率：由 $\frac{FP}{FP+TN}$ 给出，表示假正例与假正例加真负例的比率。
- 准确率：由 $\frac{TP+TN}{TP+FP+TN+FN}$ 给出，表示真实预测与所有数据的比率。
- F1 分数：由 $\frac{2*P*R}{P+R}$ 给出，通过取精确率和召回率的调和平均数，将两者结合在一个指标中，其中 P 和 R 分别是精确率和召回率。

4.4. 新型爬虫检测

如前所述，爬虫在多样性和动态变化方面存在问题，网络环境复杂且易变，新型爬虫的出现可能会使预训练模型变得不那么鲁棒。为了检查我们的方法在面对新型爬虫时的鲁棒性，我们进行了以下实验。每次，我们从聚类结果中提取一种类型的爬虫，并将其标记为新型爬虫。这部分数据没有参与训练过程。模型训练完成后，我们通过使用用户数据和新型爬虫形成测试数据集来测试模型的鲁棒性，并观察模型在此测试集上的爬虫检测效果。对于私有数据集，我们有 6 种类型的爬虫，我们对每种类型分别进行了实验，共进行了 6 组实验。同样，对于公共数据集，我们进行了 5 组实验。我们将我们的方法与一些表现良好的先前模型进行了比较。

参数设置：对于特征选择器的参数，我们将训练回合数设置为 10，代理的步数设置为 10，策略网络的学习率为 5e-6，评论员网络的学习率为 1e-5，gamma 为 0.9，软更新率为 0.01，特征选择阈值设置为 0.4/0.6，终止状态阈值为 0.9，重放缓冲区大小为 5000，alpha 为 10，批量大小为 32，延迟系数 τ 为 0.01。对于会话分类器的参数，我们将学习率设置为 1e-3，批量大小设置为 32。

基线：

DTMC：该方法基于一阶离散时间马尔可夫链模型，该模型表示机器人和人类资源请求模式。他们使用此模型来计算请求模式更有可能是由爬虫还是用户生成的概率。

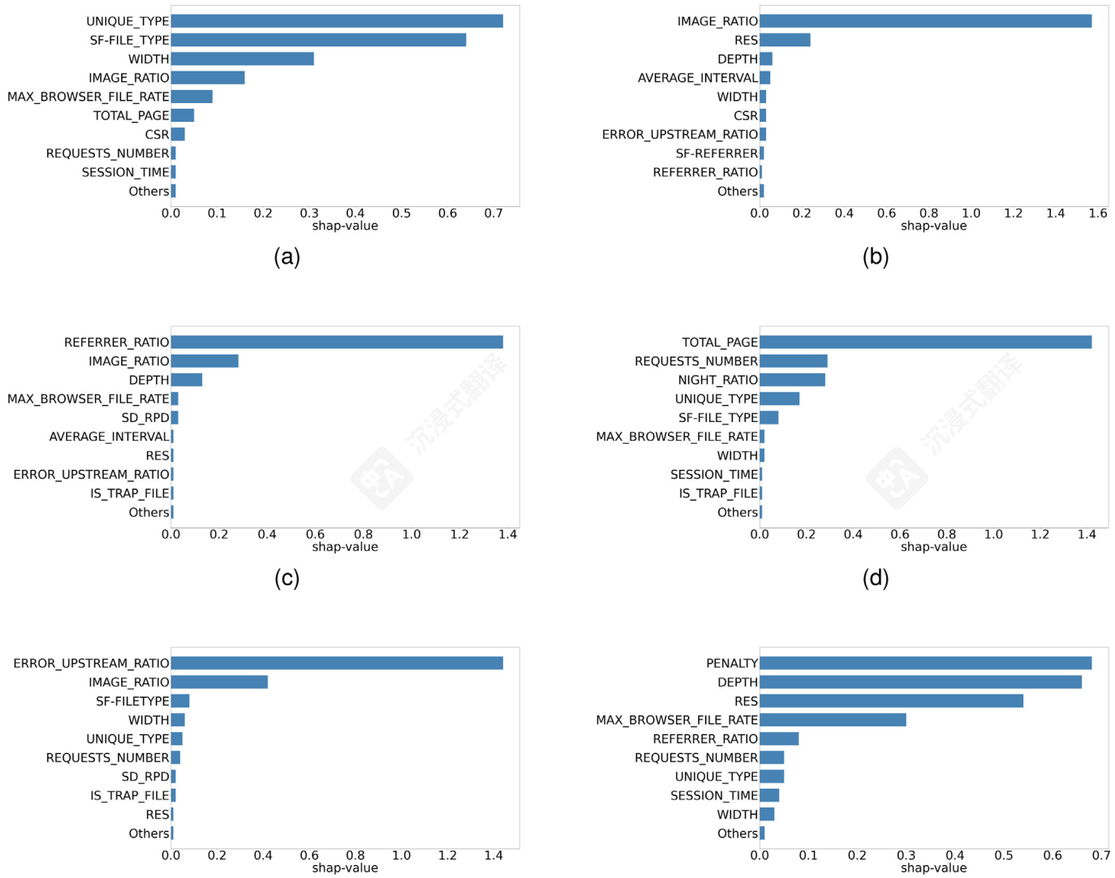


Fig. 4. The feature importance of each clustering result to user data in the private dataset. Each subgraph represents a comparison between a crawler clustering result with user data.

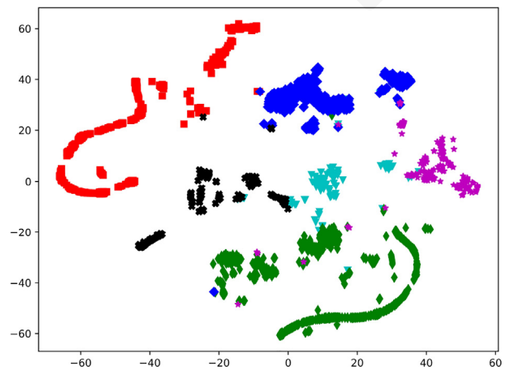


Fig. 5. Clustering results of the public dataset. The blue part is the user data, and the other colors are the crawler data.

- FRS_SOM: This model consists of two parts, the feature selecting module using the fuzzy rough set algorithm and the classification module using the SOM algorithm [13].
- MLP_SPRT: This approach uses deep neural networks combined with Wald's Sequential Probability Ratio Test to express the relationship between subsequent HTTP requests in an ongoing session and to assess the likelihood of each session being generated by a bot or human before it ends [40].
- BNC: This method constructs a Bayesian network that automatically classifies access log sessions as crawlers or users. They apply machine learning techniques to determine the parameters of the probabilistic model. The resulting classification is based on the maximum posterior probability of two classes [41].

Result: After a series of tuning parameters, we made each model show its best results. We used Recall, Precision, F1-score and Error rate as the judges of model effectiveness, and we first conducted experiments on the private dataset. The experimental results are shown in Fig. 7. The experimental results show that our model has the best performance in all metrics. DTMC has the worst per-

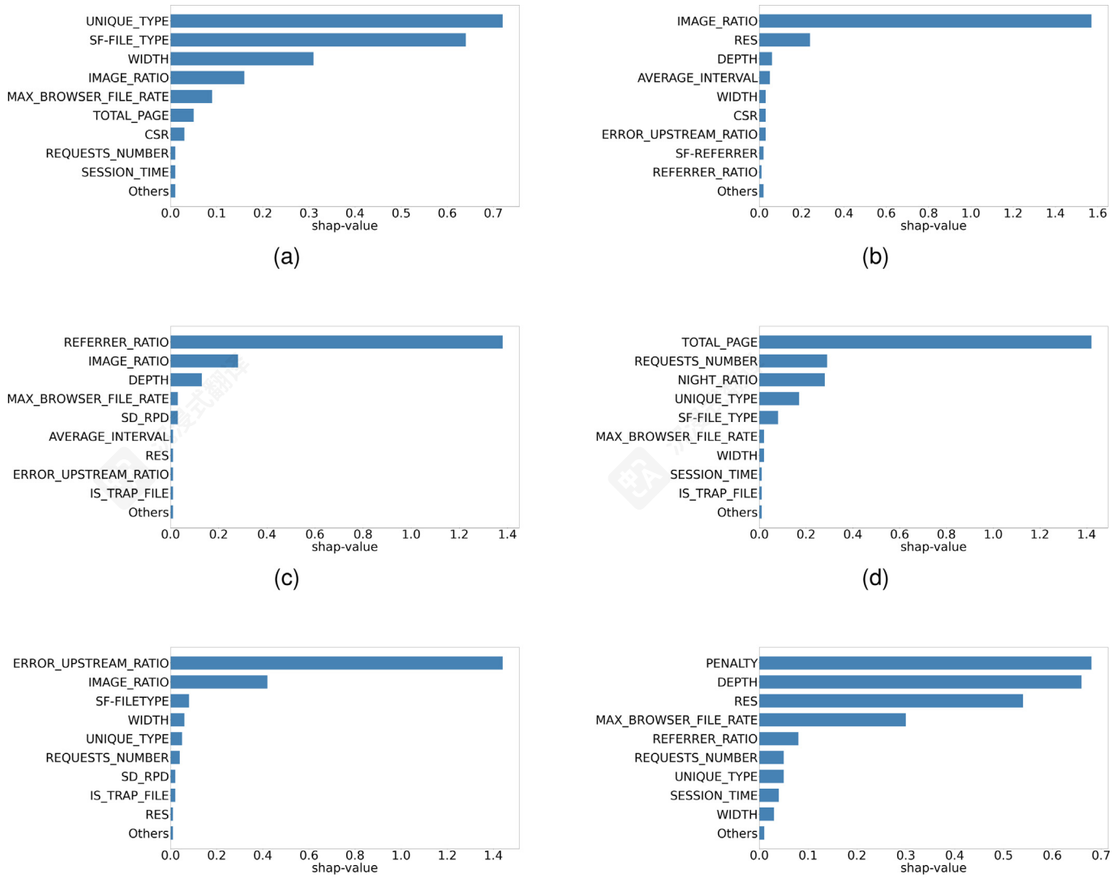


Fig. 4. 每个聚类结果对私有数据集中用户数据的特征重要性。每个子图代表爬虫聚类结果与用户数据的比较。 esult with user

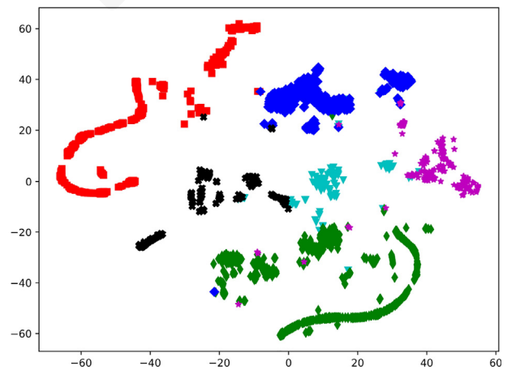


图 5. 公共数据集的聚类结果。蓝色部分是用户数据，其他颜色是爬虫数据。

• FRS_SOM：此模型由两部分组成，即使用模糊粗糙集算法的特征选择模块和使用 SOM 算法的分类模块 [13]。• MLP_SPRT：该方法使用深度神经网络与 Wald 的顺序概率比测试相结合，以表达持续会话中后续 HTTP 请求之间的关系，并在会话结束时评估每个会话是由机器人还是人类生成的可能性 [40]。

BNC：该方法构建了一个贝叶斯网络，可以自动将访问日志会话分类为爬虫或用户。他们应用机器学习技术来确定概率模型参数。最终的分类基于两个类别的最大后验概率

结果：经过一系列参数调整，我们使每个模型都表现出最佳效果。我们使用召回率、精确率、 F1 分数和错误率作为模型有效性的评判标准，并首先在私有数据集上进行了实验。实验结果如图 7 所示。实验结果表明，我们的模型在所有指标上均表现最佳。DTMC 表现最差。

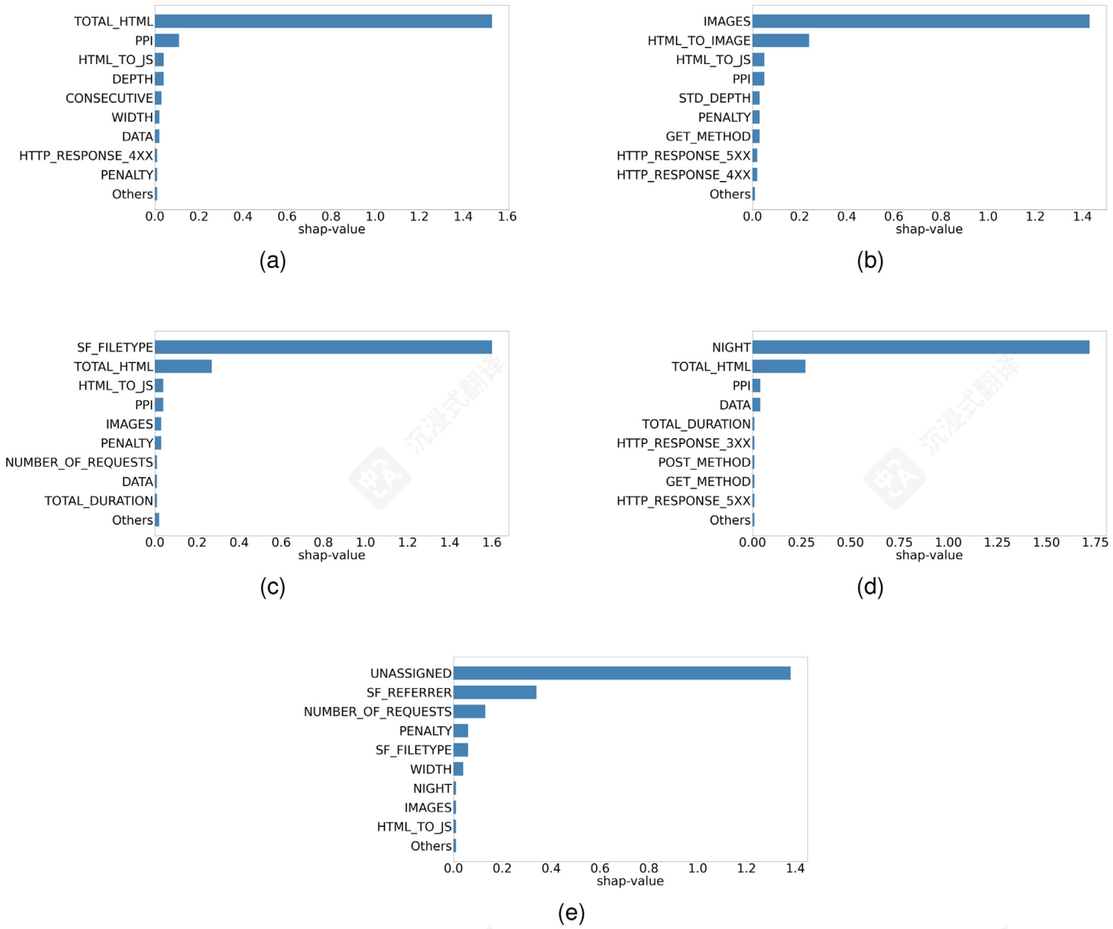


Fig. 6. The characteristic importance of each clustering result to user data in public dataset.

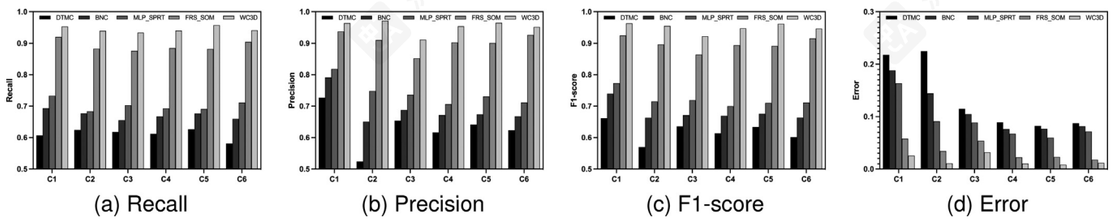


Fig. 7. The result of new type crawler detection for private dataset.

formance, considering that the private dataset does not have a complex file structure, and there are no frequent requests for different files on the website, so this method, based on the resource request patterns, does not perform well. BNC also performs poorly because it uses only six features, which can not represent the characteristics of the new type of crawler. MLP_SPRT performs relatively well because all the features are used, and the characteristics of the new crawler can be retained, but the recognition effect is not particularly good on account of a large number of redundant features and interference features. FRS_SOM performs well, which indicates that its feature filtering method filters out certain features that do not vary much across crawler types, allowing the model to detect some of the new types of crawlers. Our method performs better than FRS_SOM and can detect more new types of crawlers, thanks to the exploration process of reinforcement learning, where the feature selection module generates some

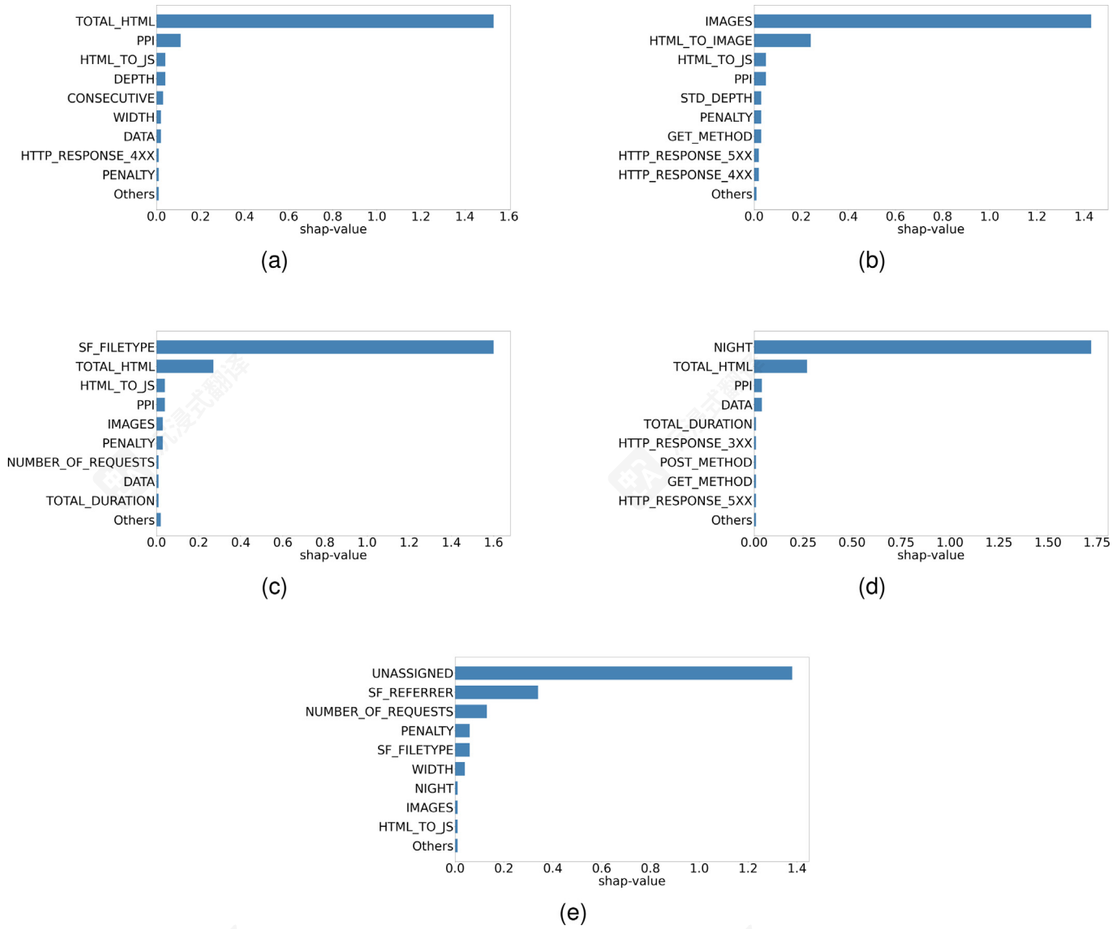


Fig. 6. The characteristic importance of each clustering result to user data in public dataset.

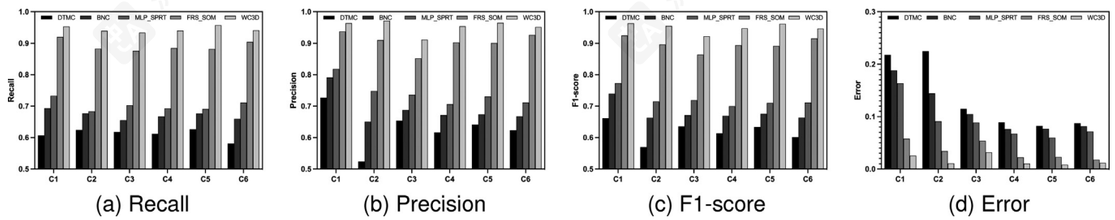


图 7. 对私有数据集进行的新型爬虫检测结果。

性能表现不佳，考虑到私有数据集没有复杂的文件结构，并且网站上没有频繁请求不同文件，所以这种方法，基于资源请求模式，表现不佳。BNC 也表现不佳，因为它只使用了六个特征，这不能代表新型爬虫的特征。MLP_SPRT 表现相对较好，因为使用了所有特征，并且可以保留新型爬虫的特征，但识别

由于大量冗余特征和干扰特征，分类效果并不特别好。FRS_SOM 表现良好，这表明其特征过滤方法过滤掉了一些在不同爬虫类型中变化不大的特征，使得模型能够检测到一些新的爬虫类型。我们的方法比 FRS_SOM 表现更好，并且能够检测到更多的新爬虫类型，这得益于强化学习过程中的探索过程，其中特征选择模块生成了一些

new feature combination patterns through exploration, enabling the model to adapt to the unknown environment to a certain extent, in other words, it still has some detection ability when facing the new type of crawlers. Similarly, we have done experiments on the public dataset, and the results are shown in Fig. 8. From the experimental results, it can be seen that our method achieves the best results. Considering that the websites used in the public dataset are more complex and the performance of each model is somewhat degraded compared to the private dataset, we do not experiment with DTMC because we do not have access to the characteristics of the resource request patterns of the public dataset.

4.5. Crawler detection accuracy

We have proved that our model has a more robust performance on the new type of crawler detection problem through experiments. We also want to verify the inherent detection accuracy of the model without considering the crawler dynamic change problem. We add an unsupervised baseline: DBC_WRD. The four features they used were Trap file request, Maximum rate of browser file request, Penalty and Percentage of 304 response codes, where Penalty and Maximum rate of browser file request are newly proposed. The experimental results show that the method in our paper has a more outstanding performance on both datasets.

Result: The results for the private and public datasets are shown in Table 2 and Table 3.

The experimental results show that our model has the best performance, with an accuracy rate of 0.99. DTMC still has the worst performance. In addition to our method, the FRS_SOM method and the MLP_SPRT method also perform well. The former uses a feature selecting method to filter out low-importance features to achieve better classification results, proving the effectiveness of feature selection. The latter uses all features and a neural network as the classifier. Only the FRS_SOM method uses the feature selecting method among these five baselines, but compared with the uniqueness of the selecting results of the fuzzy rough set method, our method will select different results that are suitable for different types of crawlers, which is one of the reasons why our method can get a higher accuracy rate. The DBC_WRD method does not perform well on the public dataset, considering that the method uses only four features for classification, and these four features on this dataset are less critical, so the classification result is not good, which again illustrates the need for feature selection.

4.6. Feature selection baseline

To demonstrate the effectiveness of our proposed feature selection method using reinforcement learning, we compared it with other feature selection methods. We conducted experiments using DNN as downstream classifiers in combination with different feature selection methods. For those methods that require a specified number of features, we uniformly set the number to the average of the number of features selected by our method.

Baselines:

- K-Best-Selection: This method uses label information and $\chi^2$to rank features and selects the top k features to form a feature set [42].
- mRMR: This method first ranks the features by minimizing their redundancy, while maximizing their relevance to the labels, and then selects the top k features [43].
- Recursive Feature Elimination (RFE): This method discards the features step by step in a recursive manner. First, the predictor is trained through all the features and the predictor scores the importance of each feature. After that, the least important features are deselected. This process is cycled until the required number of features are selected [44].
- FRS: this method selects features by FRS algorithm, which can consider the similarity of all features simultaneously and deal with the ambiguity of the data effectively [13].

Result: The results for the private and public datasets are shown in Table 4 and Table 5. The experimental results show that the feature set selected by our feature selection method is more effective in characterizing different types of crawlers, while other methods perform poorly because the selected feature set is single and separated from the downstream classifier

4.7. Screening ratio analysis

As mentioned before, we set threshold t_1 to map the weight w_i of each feature output from the policy network to feature mask m_i, where $w_i \in (0, 1)$ and $m_i \in \{0, 1\}$. If w_i is greater than t_1, the corresponding feature mask m_i is 1, otherwise it is 0. The setting of this parameter affects the number of retained features, which affects the accuracy of crawler detection. Therefore, we conducted a detailed experiment on this parameter to analyze the relationship between the number of retained features and the accuracy of crawler detection.

The results of the private dataset are shown in Fig. 9(a), where the red line shows the variation of classification accuracy with t_1, and the green line shows the variation of the average number of retained features of all sessions with t_1. From the figure, we can see that if we set t_1 to 0, the model will retain all features, at this time, the feature selector will no longer work, the model only has the session classifier module, which is equivalent to direct use of DNN for crawler detection, and the accuracy can reach 0.91. With the increase of t_1, more and more features are screened out, and the classification accuracy gradually increases. In this process, some features that will interfere with crawler detection are screened out, while those features that are more distinct between users and crawlers are retained. When the number of screened features reaches 6, the crawler detection accuracy reaches the highest 0.99. After that, as t_1 continues to increase, more features are screened out, and some features that help classify crawlers and users are also screened out, and the accuracy of the model starts

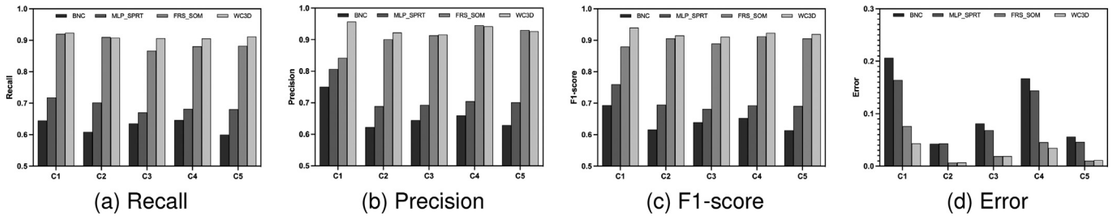


Fig. 8. The result of new type crawler detection for public dataset.

通过探索，引入了新的特征组合模式，使模型在一定程度上能够适应未知环境，换句话说，在面对新型爬虫时仍具有一定的检测能力。同样，我们在公共数据集上进行了实验，结果如图 8 所示。从实验结果可以看出，我们的方法取得了最佳效果。考虑到公共数据集中使用的网站更加复杂，与私有数据集相比，每个模型的性能都有所下降，因此我们没有对 DTMC 进行实验，因为我们无法获取公共数据集中资源请求模式的特点。

4.5. 爬虫检测准确率

我们通过实验证明了我们的模型在新型爬虫检测问题上具有更鲁棒的性能。我们还想验证模型固有的检测准确率，不考虑爬虫动态变化问题。我们添加了一个无监督基线：DBCWRD。他们使用的四个特征是陷阱文件请求、浏览器文件请求的最大速率、惩罚和 304 响应代码的百分比，其中惩罚和浏览器文件请求的最大速率是新的提议。实验结果表明，我们论文中的方法在两个数据集上都有更突出的性能。

结果：私有和公共数据集的结果分别如表 2 和表 3 所示。

实验结果表明，我们的模型性能最佳，准确率达到 0.99。DTMC 仍然表现最差。除了我们的方法外，FRS-SOM 方法和 MLP-SPRT 方法也表现良好。前者使用特征选择方法过滤掉低重要性的特征，以实现更好的分类结果，证明了特征选择的有效性。后者使用所有特征和一个神经网络作为分类器。在这五个基线中，只有 FRS-SOM 方法使用了特征选择方法，但与模糊粗糙集方法选择结果的独特性相比，我们的方法会选择适合不同类型爬虫的不同结果，这也是我们的方法能够获得更高准确率的原因之一。DBC-WRD 方法在公共数据集上表现不佳，考虑到该方法仅使用四个特征进行分类，而这四个特征在此数据集上不太关键，因此分类结果不佳，这再次说明了特征选择的需求。

4.6. 基于强化学习的特征选择基准

为了展示我们提出的基于强化学习的特征选择方法的有效性，我们将其与其他特征选择方法进行了比较。我们使用 DNN 作为下游分类器，结合不同的特征选择方法进行了实验。对于需要指定特征数量的方法，我们将特征数量统一设置为我们的方法选择的特征数量的平均值。

基准：

K-Best-Selection：这种方法使用标签信息和 χ^2 对特征进行排序，并选择前 k 个特征形成特征集 [42]。• mRMR：这种方法首先通过最小化冗余来对特征进行排序，同时最大化其与标签的相关性，然后选择前 k 个特征 [43]。• 递归特征消除（ RFE ）：这种方法以递归的方式逐步丢弃特征。首先，通过所有特征训练预测器，并评估每个特征的重要性。然后，选择最不重要的特征进行 deselection。这个过程循环进行，直到选择所需数量的特征 [44]。• FRS：这种方法通过 FRS 算法选择特征，该算法可以同时考虑所有特征之间的相似性，并有效地处理数据的模糊性 [13]。

结果：私有和公共数据集的结果分别显示在表 4 和表 5 中。实验结果表明，我们特征选择方法选择的特征集在表征不同类型的爬虫方面更为有效，而其他方法表现不佳，因为选择的特征集单一且与下游分类器分离。

4.7. 筛选比例分析

如前所述，我们将阈值 t_1 设置为将策略网络输出的每个特征的权重 w_i 映射到特征掩码 m_i，其中 $w_i \in (0; 1)$ 和 $m_i \in \{0; 1\}$。如果 w_i 大于 t_1，则相应的特征掩码 m_i 为 1，否则为 0。此参数的设置会影响保留特征的数量，从而影响爬虫检测的准确性。因此，我们对这个参数进行了详细的实验，以分析保留特征数量与爬虫检测准确率之间的关系。

私有数据集的结果如图 9(a) 所示，其中红色曲线表示分类准确率随 t_1 的变化，绿色曲线表示所有会话保留特征平均数量的变化随 t_1 的变化。从图中可以看出，如果我们把 t_1 设为 0，模型将保留所有特征，此时特征选择器将不再工作，模型只有会话分类模块，这相当于直接使用 DNN 进行爬虫检测，准确率可以达到 0.91。随着 t_1 的增加，越来越多的特征被筛选出来，分类准确率逐渐提高。在这个过程中，一些会干扰爬虫检测的特征被筛选出来，而那些在用户和爬虫之间更明显的特征被保留。当筛选出的特征数量达到 6 时，爬虫检测的准确率达到最高 0.99。之后，随着 t_1 的继续增加，更多的特征被筛选出来，一些有助于分类爬虫和用户的特点也被筛选出来，模型的准确率开始

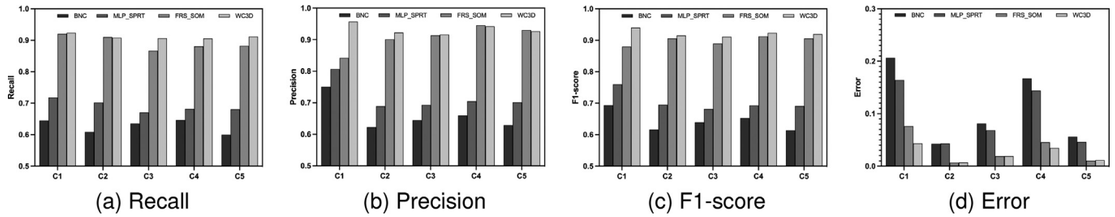


图 8. 公共数据集新型爬虫检测的结果。

Table 2
The result of crawler detection for private dataset.

Method	Recall	Precision	F1	Acc	Error
DBC_WRD	0.9269	0.9660	0.9461	0.9683	0.0192
DTMC	0.6835	0.8634	0.7630	0.9244	0.0475
FRS_SOM	0.9647	0.9926	0.9784	0.9870	0.0109
MLP_SPRT	0.9621	0.9867	0.9742	0.9847	0.0113
BNC	0.9014	0.9942	0.9455	0.9684	0.0192
WC3D	**0.9982**	**0.9991**	**0.9987**	**0.9992**	**0.0034**

Table 3
The result of crawler detection for private dataset.

Method	Recall	Precision	F1	Acc	Error
DBC_WRD	0.9105	0.4034	0.5591	0.7301	0.2543
DTMC	/	/	/	/	/
FRS_SOM	0.8971	0.8519	0.8730	0.9490	0.0367
MLP_SPRT	0.8984	0.8785	0.8883	0.9550	0.0295
BNC	0.8851	0.6146	0.7254	0.8632	0.0845
WC3D	**0.9282**	**0.8788**	**0.9028**	**0.9607**	**0.0234**

Table 4
The result of feature selection analysis for private dataset.

Method	Recall	Precision	F1	Acc	Error
KBS	0.8117	0.8023	0.8069	0.8835	0.0857
mRMR	0.8439	0.8472	0.8456	0.9075	0.0652
RFE	0.8991	0.9107	0.9048	0.9433	0.0378
FRS	0.9332	0.9436	0.9384	0.9632	0.0239
WC3D	**0.9982**	**0.9991**	**0.9987**	**0.9992**	**0.0034**

Table 5
The result of feature selection analysis for public dataset.

Method	Recall	Precision	F1	Acc	Error
KBS	0.7743	0.7693	0.7718	0.9083	0.0582
mRMR	0.8142	0.8371	0.8255	0.9310	0.0397
RFE	0.8688	0.8840	0.8764	0.9509	0.0286
FRS	0.9178	0.8762	0.9019	0.9545	0.0238
WC3D	**0.9282**	**0.8788**	**0.9028**	**0.9607**	**0.0234**

to decrease. Until the number of retained features reaches 11, the model's accuracy drops to 0.87. The results of the public dataset are shown in Fig. 9(b). As with the private dataset, in the public dataset, the number of retained features decreases as t_1 increases, while the accuracy rate first increases and then decreases. When t_1 is 0, the model retains all features in the dataset, and the crawler

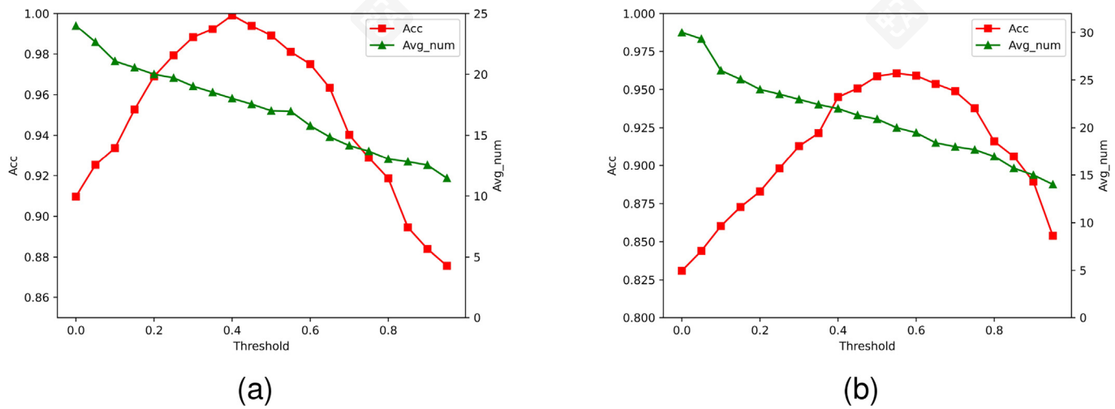


Fig. 9. The relationship between model accuracy and number of retained features. The result of the private dataset is shown in (a) and the result of the public dataset is shown in (b).

表 2 私有数据集爬虫检测的结果。

方法	回忆	精度	F1	Acc	Error
DBC_WRD	0.9269	0.9660	0.9461	0.9683	0.0192
DTMC	0.6835	0.8634	0.7630	0.9244	0.0475
FRS_SOM	0.9647	0.9926	0.9784	0.9870	0.0109
MLP_SPRT	0.9621	0.9867	0.9742	0.9847	0.0113
BNC	0.9014	0.9942	0.9455	0.9684	0.0192
WC3D	**0.9982**	**0.9991**	**0.9987**	**0.9992**	**0.0034**

表 3 私有数据集爬虫检测的结果。

方法	回忆	精度	F1	Acc	Error
DBC_WRD	0.9105	0.4034	0.5591	0.7301	0.2543
DTMC	/	/	/	/	/
FRS_SOM	0.8971	0.8519	0.8730	0.9490	0.0367
MLP_SPRT	0.8984	0.8785	0.8883	0.9550	0.0295
BNC	0.8851	0.6146	0.7254	0.8632	0.0845
WC3D	**0.9282**	**0.8788**	**0.9028**	**0.9607**	**0.0234**

表 4 私有数据集特征选择分析结果。

方法	回忆	精度	F1	Acc	Error
KBS	0.8117	0.8023	0.8069	0.8835	0.0857
mRMR	0.8439	0.8472	0.8456	0.9075	0.0652
RFE	0.8991	0.9107	0.9048	0.9433	0.0378
FRS	0.9332	0.9436	0.9384	0.9632	0.0239
WC3D	**0.9982**	**0.9991**	**0.9987**	**0.9992**	**0.0034**

表 5 公共数据集特征选择分析结果。

Method	回忆	Precision	F1	Acc	Error
KBS	0.7743	0.7693	0.7718	0.9083	0.0582
mRMR	0.8142	0.8371	0.8255	0.9310	0.0397
RFE	0.8688	0.8840	0.8764	0.9509	0.0286
FRS	0.9178	0.8762	0.9019	0.9545	0.0238
WC3D	**0.9282**	**0.8788**	**0.9028**	**0.9607**	**0.0234**

随着保留特征数量的减少，模型的准确率降至 0.87。公开数据集的结果显示在图 9(b)。与私有数据集一样，在公开数据集中

数据集中，随着 t1 的增加，保留特征的数量减少，而准确率先上升后下降。当 t1 为 0 时，模型保留数据集中的所有特征，而爬虫

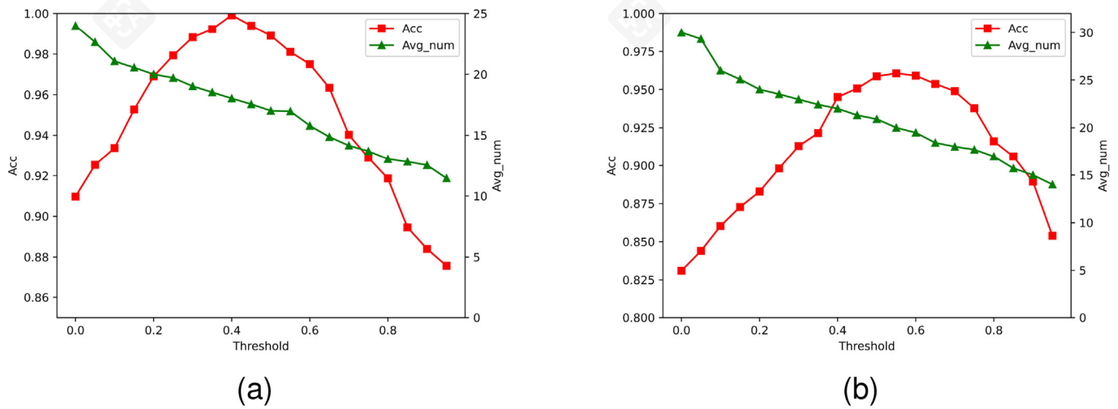


图 9. 模型准确率与保留特征数的关系。私有数据集的结果显示在 (a) 中，公共数据集的结果显示在 (b) 中。

detection accuracy is 0.83. When t_1 is 0.55, the model filters out 10 features, and the accuracy reaches the highest 0.96. And when t_1 reaches the maximum, the model screened out 16 features, and the accuracy is only 0.85.

Based on the above experimental results, we can conclude that there are features in the dataset that can negatively affect the accuracy of crawler detection, and screening out these features will help distinguish crawlers from users. Second, by setting a threshold t_1, we can make the model screen out insufficient features to characterize a specific type of crawler and keep the good features to maximize the model's accuracy.

5. Conclusion

This paper proposes a new crawler detection method based on reinforcement learning for diversity and dynamics. The model consists of two modules: feature selector and session classifier. The feature selector module uses the DDPG architecture, consisting of a policy network and a critic network. This module is responsible for maximizing the reward by selecting suitable feature sets for different types of crawlers. It uses the feature distribution of the session as the state, the feature selection result as the action, and the session classification result as the reward. The session classification module uses DNN architecture. This module takes the retained features as input, outputs its classification results, and provides reward feedback to the feature selector based on the accuracy of the classification results. The two modules are trained jointly to achieve the best classification results.

Through experiments, we first demonstrate the existence of crawlers' diversity in the two datasets we used. Then we compare our method with state-of-the-art methods for the model's robustness when the new type of crawler appears, and our method achieves the best experimental results on both datasets. The results demonstrate the importance of feature selection and also that different types of crawlers have different feature distributions, and our method can select the appropriate set of representational features for crawlers based on their types. To sum up, our method has better performance in the complex network environment with crawlers' diversity and dynamics.

CRediT authorship contribution statement

Yang Gao: Conceptualization, Methodology, Software, Writing – original draft. **Xiaoyang Wang:** Data curation, Investigation. **Zunlei Feng:** Writing - review & editing. **Mingli Song:** Supervision. **Xingen Wang:** Project administration. **Xinyu Wang:** Formal analysis, Visualization. **Chun Chen:** Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P.-N. Tan, V. Kumar, Discovery of web robot sessions based on their navigational patterns, in: Intelligent Technologies for Information Analysis, Springer, 2004, pp. 193–222.
[2] H.N. Rude, D. Doran, Request type prediction for web robot and internet of things traffic, in: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, 2015, pp. 995–1000.
[3] I. Zeifman, Bot traffic report 2016. in Imperva Incapsula, 2017.
[4] C.L. Giles, Y. Sun, and I.G. Councill, Measuring the web crawler ethics, in Proceedings of the 19th international conference on World wide web, 2010, pp. 1101–1102.
[5] V. Almeida, D. Menascé, R. Riedi, F. Peligrinelli, R. Fonseca, and W. Meira Jr, Analyzing web robots and their impact on caching, in Proc. Sixth Workshop on Web Caching and Content Distribution, 2001, pp. 20–22.
[6] M.D. Dikaiakos, A. Stassopoulou, L. Papageorgiou, An investigation of web crawler behavior: characterization and metrics, Comput. Commun. 28 (8) (2005) 880–897.
[7] S. Ye, G. Lu, and X. Li, Workload-aware web crawling and server workload detection, in Proceedings of the second Asia-Pacific advanced network research workshop. Citeseer, 2004, pp. 263–269.
[8] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer networks and ISDN systems 30 (1–7) (1998) 107–117.
[9] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, Searching the web, ACM Transactions on Internet Technology (TOIT) 1 (1) (2001) 2–43.
[10] D. Doran, S.S. Gokhale, An integrated method for real time and offline web robot detection, Expert Syst. 33 (6) (2016) 592–606.
[11] G. Suchacka, I. Motyka, Efficiency analysis of resource request patterns in classification of web robots and humans, ECMS (2018) 475–481.
[12] M. Zabihi, M.V. Jahan, J. Hamidzadeh, A density based clustering approach for web robot detection, in: 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), IEEE, 2014, pp. 23–28.
[13] J. Hamidzadeh, M. Zabihimayvan, R. Sadeghi, Detection of web site visitors based on fuzzy rough sets, Soft. Comput. 22 (7) (2018) 2175–2188.
[14] D. Doran, S.S. Gokhale, Web robot detection techniques: overview and limitations, Data Min. Knowl. Disc. 22 (1) (2011) 183–210.
[15] T. Kabe, M. Miyazaki, Determining www user agents from server access log, in: Proceedings Seventh International Conference on Parallel and Distributed Systems: Workshops, IEEE, 2000, pp. 173–178.
[16] P. Huntington, D. Nicholas, H.R. Jamali, Web robot detection in the scholarly information environment, J. Inf. Sci. 34 (5) (2008) 726–741.
[17] S. Kwon, Y.-G. Kim, S. Cha, Web robot detection based on pattern-matching technique, J. Inf. Sci. 38 (2) (2012) 118–126.
[18] S. Kwon, M. Oh, D. Kim, J. Lee, Y.-G. Kim, S. Cha, Web robot detection based on monotonous behavior, Proc. Inf. Sci. Ind. Appl. 4 (2012) 43–48.
[19] Q. Bai, G. Xiong, Y. Zhao, L. He, Analysis and detection of bogus behavior in web crawler measurement, Proc. Comput. Sci. 31 (2014) 1084–1091.
[20] F. Quan-Long, Y. Bin, Y. Zhou-Hua, X. Lei, Spider detection based on trap techniques, J. Comput. Appl. 30 (07) (2010) 1782.
[21] D. Doran, K. Morillo, and S.S. Gokhale, A comparison of web robot and human requests, in Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, 2013, pp. 1374–1380.
[22] M. Motoyama, B. Meeder, K. Levchenko, G.M. Voelker, and S. Savage, Measuring online service availability using twitter, in 3rd Workshop on Online Social Networks (WOSN 2010), 2010.
[23] G. Jacob, E. Kirda, C. Kruegel, and G. Vigna, {PUBCRAWL}: Protecting users and businesses from {CRAWLers}, in 21st USENIX Security Symposium (USENIX Security 12), 2012, pp. 507–522.
[24] A. Lagopoulos, G. Tsoumakas, G. Papadopoulos, Web robot detection: A semantic approach, in: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2018, pp. 968–974.
[25] Y. Hiltunen, M. Lappalainen, Automated personalisation of internet users using self-organising maps, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2002, pp. 31–34.
[26] W. Zhu, H. Gao, Z. He, J. Qin, B. Han, A hybrid approach for recognizing web crawlers, in: International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2019, pp. 507–519.
[27] X. Li, J. Ren, MICQ-IPSO: an effective two-stage hybrid feature selection algorithm for high-dimensional data, Neurocomputing 501 (2022) 328–342, https://doi.org/10.1016/j.neucom.2022.05.048 [Online]. Available:.
[28] A. Tan, J. Liang, W. Wu, J. Zhang, L. Sun, C. Chen, Fuzzy rough discrimination and label weighting for multi-label feature selection, Neurocomputing 465 (2021) 128–140, https://doi.org/10.1016/j.neucom.2021.09.007 [Online]. Available:.
[29] H.E. Kiziloz, Classifier ensemble methods in feature selection, Neurocomputing 419 (2021) 97–107, https://doi.org/10.1016/j.neucom.2020.07.113 [Online]. Available:.
[30] T. Gržinić, L. Mršić, J. Šaban, Lino-an intelligent system for detecting malicious web-robots, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2015, pp. 559–568.
[31] M. Zabihi, M. Vafaei Jahan, and J. Hamidzadeh, A density based clustering approach to distinguish between web robot and human requests to a web server, The ISC International Journal of Information Security, vol. 6, no. 1, pp. 77–89.
[32] S. Fan, X. Zhang, Z. Song, Reinforced knowledge distillation: Multi-class imbalanced classifier based on policy gradient reinforcement learning, Neurocomputing 463 (2021) 422–436, https://doi.org/10.1016/j.neucom.2021.08.040 [Online]. Available:.
[33] Y. Li, Y. Fang, Z. Akhtar, Accelerating deep reinforcement learning model for game strategy, Neurocomputing 408 (2020) 157–168, https://doi.org/10.1016/j.neucom.2019.06.110 [Online]. Available:.
[34] J. Janisch, T. Pevný, and V. Lisý, Classification with costly features using deep reinforcement learning, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, 2019, pp. 3959–3966.
[35] Z. Xu, Y. Wang, J. Jiang, J. Yao, L. Li, Adaptive feature selection with reinforcement learning for skeleton-based action recognition, IEEE Access 8 (2020) 213038–213051.

检测准确率为 0.83。当 t₁ 为 0.55 时，模型筛选出 10 个特征，准确率最高达到 0.96。而当 t₁ 达到最大值时，模型筛选出 16 个特征，准确率仅为 0.85。

基于上述实验结果，我们可以得出结论：数据集中存在一些特征会负面影响爬虫检测的准确率，筛选出这些特征有助于区分爬虫和用户。其次，通过设置阈值 t₁，我们可以使模型筛选出不足以表征特定类型爬虫的特征，同时保留良好特征以最大化模型的准确率。

5. 结论

本文提出了一种基于强化学习的针对多样性和动态性的网络爬虫检测新方法。模型由两个模块组成：特征选择器和会话分类器。特征选择器模块使用 DDPG 架构，包括策略网络和评价网络。该模块负责通过为不同类型的爬虫选择合适的特征集来最大化奖励。它使用会话的特征分布作为状态，特征选择结果作为动作，会话分类结果作为奖励。会话分类器模块使用 DNN 架构。该模块以保留的特征作为输入，输出其分类结果，并根据分类结果的准确率向特征选择器提供奖励反馈。这两个模块联合训练以实现最佳的分类结果。

通过实验，我们首先证明了我们在使用的两个数据集中爬虫的多样性存在。然后，我们比较了我们的方法与最先进的方法在出现新型爬虫时的模型鲁棒性，我们的方法在两个数据集上都取得了最佳的实验结果。结果表明，特征选择的重要性，以及不同类型的爬虫具有不同的特征分布，我们的方法可以根据爬虫的类型选择适当的代表性特征集。总之，我们的方法在具有爬虫多样性和动态性的复杂网络环境中具有更好的性能。

基于强化学习的针对多样性和动态性的网络爬虫检测t

高扬：概念化、方法论、软件、写作 - 初稿。王晓阳：数据整理、调查。冯尊磊：写作 - 审稿与编辑。宋明丽：监督。王新根：项目管理。王欣宇：形式分析、可视化。陈春：资金获取。

利益冲突声明

作者声明，他们没有已知的可能影响本文报道工作的财务利益或个人关系。

参考文献

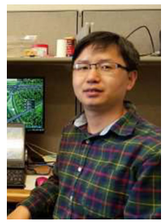
P.-N. Tan, V. Kumar, 基于导航模式发现网络爬虫会话，载于《信息分析智能技术》，Springer 出版社，2004 年，第 193–222 页。H.N. Rude, D. Doran, 针对网络爬虫和物联网流量的请求类型预测，载于《 2015 年第 14 届国际机器学习与应用会议（ ICMLA ）》，IEEE 出版社，2015 年，第 995–1000 页。I. Zeifman, 2016 年网络爬虫流量报告，载于 Imperva Incapsula，2017 年。C.L. Giles, Y. Sun, 和 I.G. Councill, 测量网络爬虫伦理，载于《第 19 届国际万维网会议论文集》，2010 年，第 1101–1102 页。
[5] V. Almeida, D. Menascé, R. Riedi, F. Peligrinelli, R. Fonseca, 和 W. Meira Jr, 分析网络爬虫及其对缓存的影响，载于《第六届网络缓存与内容分发研讨会论文集》，2001 年，第 20–22 页。
[6] M.D. Dikaiakos, A.Stassopoulou, L. Papageorgiou, 网络爬虫行为的调查：特征和度量，Comput. Commun. 28 (8) (2005) 880–897。
[7] S. Ye, G. Lu, 和 X. Li, 基于工作负载的网络爬虫和服务器工作负载检测，载于《第二届亚太先进网络研究研讨会论文集》，Citeseer，2004 年，第 263–269 页。
[8] S.Brin, L. Page, 大规模超文本搜索引擎的解剖结构，Computer networks and ISDN systems 30 (1–7) (1998) 107–117。
[9] . rasu, . o, . arc a- o na, . aepc e, . ag avan, earc ng t e we , ACM Transactions on Internet Technology (TOIT) 1 (1) (2001) 2–43.
[10] D. Doran, S.S. Gokhale, An integrated method for real time and offline web robot detection, Expert Syst. 33 (6) (2016) 592–606.
[11] G. Suchacka, I. Motyka, Efficiency analysis of resource request patterns in classification of web robots and humans, ECMS (2018) 475–481.
[12] M. Zabihi, M.V. Jahan, J. Hamidzadeh, 一种基于密度的网络爬虫检测聚类方法，载于 2014 年第 4 届计算机与知识工程国际会议（ ICCKE ），IEEE，2014 年，第 23–28 页
[13] J. Hamidzadeh, M. Zabihimayvan, R. Sadeghi, Detection of web site visitors based on fuzzy rough sets, Soft. Comput. 22 (7) (2018) 2175–2188.
[14] D. Doran, S.S. Gokhale, Web robot detection techniques: overview and limitations, Data Min. Knowl. Disc. 22 (1) (2011) 183–210.
[15] T. Kabe, M. Miyazaki, 从服务器访问日志中确定 www 用户代理，载于第七届并行与分布式系统国际会议：研讨会，IEEE，2000 年，第 173–178 页
[16] P. Huntington, D. Nicholas, H.R. Jamali, 学术信息环境中的网络爬虫检测，信息科学杂志，第 34 卷第 5 期（ 2008 年），第 726–741 页
[17] S. Kwon, Y.-G. Kim, S. Cha, 基于模式匹配技术的网络爬虫检测，信息科学杂志，第 38 卷第 2 期（ 2012 年），第 118–126 页
[18] S. Kwon, M. O , D. Kim, J. Lee, Y.-G. Kim, S. C a, We ro ot etection ase on monotonous behavior, Proc. Inf. Sci. Ind. Appl. 4 (2012) 43–48.
[19] Q. Bai, G. Xiong, Y. Zhao, L. He, Analysis and detection of bogus behavior in web crawler measurement, Proc. Comput. Sci. 31 (2014) 1084–1091.
[20] F. Quan-Long, Y. Bin, Y. Zhou-Hua, X. Lei, 基于陷阱技术的蜘蛛检测，计算机应用杂志，第 30 卷第 7 期（ 2010 年），第 1782 页
[21] D. Doran, K. Morillo, S.S. Gokhale，比较网络爬虫和人类请求，载于 2013 年 IEEE/ACM 国际社交网络分析与挖掘会议论文集，2013 年，第 1374–1380 页
[22] M. Motoyama, B. Meeder, K. Levchenko, G.M. Voelker, 和 S. Savage, 使用 Twitter 测量在线服务可用性，在 2010 年第 3 届在线社交网络研讨会（ WOSN 2010 ）上，2010 年。
[23] G.Jacob, E. Kirda, C. Kruegel, 和 G. Vigna, {PUBCRAWL}: 保护用户和企业免受 {CRAWLers} 的侵害，在 2012 年第 21 届 USENIX 安全研讨会（ USENIX Security 12 ）上，pp. 507–522。
[24] A. Lagopoulos, G. Tsoumakas, G. Papadopoulos, 网络机器人检测：一种语义方法，在 2018 年第 30 届 IEEE 工具与人工智能国际会议（ ICTAI ）上，IEEE，2018 年，pp. 968–974。
[25] Y. Hiltunen, M. Lappalainen, 使用自组织图自动个性化互联网用户，在 2002 年国际智能数据工程与自动学习会议，Springer，2002 年，pp. 31–34。
[26] W. Zhu, H. Gao, Z. He, J. Qin, B. Han, 一种识别网络爬虫的混合方法，在 2019 年无线算法、系统和应用国际会议，Springer，2019 年，pp. 507–519。
[27] X. Li, J. Ren, MICQ-IPSO：一种用于高维数据的有效两阶段混合特征选择算法，Neurocomputing 501 (2022) 328–342，https://doi.org/10.1016/j.neucom.2022.05。
[28] 基于模糊粗糙区分和标签加权的多标签特征选择，神经计算 465 （ 2021 ） 128–140，[链接](https://doi.org/10.1016/j.neucom.2021.09.Online)。可供查阅：
[29] H.E. Kiziloz，特征选择中的分类集成方法，神经计算 419 （ 2021 ） 97–107，[链接](https://doi.org/10.1016/j.neucom.2020.07.Online)。可供查阅：
[30] T. Grzinic，L. Mršic，J. Šaban，Lino-an 智能系统用于检测恶意网络爬虫，在《亚洲智能信息与数据库系统会议》 Springer，2015 年，第 559–568 页。[链接](https://)。
[31] M. Zabihi，M. Vafaei Jahan，J. Hamidzadeh，基于密度的聚类方法区分网络爬虫和人类对 Web 服务器的请求，国际信息安全信息科学杂志，第 6 卷，第 1 期，第 77–89 页。
[32] S. Fan，X. Zhang，Z. Song，强化知识蒸馏：基于策略梯度强化学习的多类别不平衡分类器，神经计算 463 （ 2021 ） 422–436，[链接](https://doi.org/10.1016/j.neucom.2021.08.Online)。可供查阅：
[33] Y. Li，Y. Fang，Z. Akhtar，加速深度强化学习游戏策略模型，神经计算 408 （ 2020 ） 157–168，[链接](https://doi.org/10.1016/j.neucom.2019.06.Online)。可供查阅：
[34] J. Janisch，T. Pevný，和 V. Lisý，使用深度强化学习进行具有高昂特征的分类，在 2019 年 AAAI 人工智能会议论文集，第 33 卷，第 01 期，第 3959–3966 页。
[35] Z. Xu，Y. Wang，J. Jiang，J. Yao，L. Li，基于强化学习的自适应特征选择用于骨骼动作识别，IEEE Access 8 (2020) 213038–213051.

[36] J. Feng, M. Huang, L. Zhao, Y. Yang, and X. Zhu, Reinforcement learning for relation classification from noisy data, in: Proceedings of the aaai conference on artificial intelligence, vol. 32, no. 1, 2018.
[37] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971, 2015.
[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in International conference on machine learning, PMLR (2014) 387–395.
[39] OzzyCzech, crawler-user-agents, https://github.com/monperrus/crawler-user-agents, 2021.
[40] A. Cabri, G. Suchacka, S. Rovetta, F. Masulli, Online web bot detection using a sequential classification approach, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2018, pp. 1536–1540.
[41] A. Stassopoulou, M.D. Dikaiakos, Web robot detection: A probabilistic reasoning approach, Comput. Netw. 53 (3) (2009) 265–278.
[42] Y. Yang and J.O. Pedersen, A comparative study on feature selection in text categorization, 1997.
[43] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238.
[44] P.M. Granitto, C. Furlanello, F. Biasioli, F. Gasperi, Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products, Chemometrics Intell. Lab. Syst. 83 (2) (2006) 83–90.

Yang Gao received the master degree in computer science from Zhejiang University of China, in 2017. He is currently pursuing the Ph.D. degree with the College of Computer Science, Zhejiang University. His research interests include anomaly detection and timeseries data mining.

Xiaoyang Wang is a master student in Computer Technology from College of Computer Science, Zhejiang University, and received his B.Eng. Degree in Computer Science and Technology from Zhejiang University. His research interests mainly include reinforcement learning, deep learning, machine learning.

Zunlei Feng is an assistant research fellow in College of Software Technology, Zhejiang University. He received his Ph.D degree in Computer Science and Technology from College of Computer Science, Zhejiang University, and B. Eng. Degree from Soochow University. His research interests mainly include computer vision, image information processing, representation learning, medical image analysis. He has authored and co-authored many scientific articles at top venues including IJCV, NeurIPS, AAAI, TVCG, ACM TOMM, and ECCV. He has served with international conferences including AAAI and PKDD, and international journals including IEEE Transactions on Circuits and Systems for Video TechnologyInformation Sciences, Neurocomputing, Journal of Visual Communication and Image Representation and Neural Processing Letters.



Mingli Song (M'06-SM'13) received the Ph.D. degree in computer science from Zhejiang University, China, in 2006. He is currently a Professor with the Microsoft Visual Perception Laboratory, Zhejiang University. His research interests include face modeling and facial expression analysis. He received the Microsoft Research Fellowship in 2004.

Xingen Wang received the graduate and PhD degrees in computer science from Zhejiang University of China, in 2005 and 2013, respectively. He is currently a research assistant in the College of Computer Science, Zhejiang University. His research interests include distributed computing and software performance.

Xinyu Wang received the graduate and PhD degrees in computer science from Zhejiang University of China, in 2002 and 2007, respectively. He was a research assistant at the Zhejiang University, from 2002 to 2007. He is currently a professor in the College of Computer Science, Zhejiang University. His research interests include streaming data analysis, formal methods, very large information systems, and software engineering.

Chun Chen is currently a Professor with the College of Computer Science, Zhejiang University. His research interests include computer vision, computer graphics, and embedded technology.

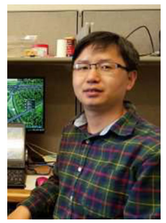
[36] J. Feng, M. Huang, L. Zhao, Y. Yang, 和 X. Zhu, 基于强化学习的从噪声数据中进行关系分类，载于《AAAI 人工智能会议论文集》，第 32 卷，第 1 期，2018 年。
[37] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, 和 D. Wierstra, 基于深度强化学习的连续控制，arXiv 预印本 arXiv:1509.02971，2015 年。
[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, 确定性策略梯度算法，载于《国际机器学习会议》，PMLR （2014） 387–395。
[39] OzzyCzech, crawler-user-agents, https://github.com/monperrus/crawler-user-agents，2021 年。
[40] A. Cabri, G. Suchacka, S. Rovetta, F. Masulli, 基于序列分类方法的在线网络爬虫检测，载于《2018 IEEE 第 20 届高性能计算与通信国际会议；IEEE 第 16 届智能城市国际会议；IEEE 第 4 届数据科学和系统国际会议（HPCC/SmartCity/DSS）》，IEEE，2018 年，第 1536–1540 页。
[41] A. Stassopoulou, M.D. Dikaiakos, 网络爬虫检测：一种概率推理方法，Comput. Netw. 53 (3) (2009) 265–278。
[42] Y. 杨和 J.O. Pedersen，关于文本分类中特征选择的比较研究，1997 年。
[43] H. 彭，F. 龙和 C. 丁，基于最大依赖性、最大相关性和最小冗余度的互信息特征选择，IEEETrans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238。
[44] P.M. Granitto，C. Furlanello，F. Biasioli，F. Gasperi，使用随机森林进行 ptr-ms 分析的特征递归消除法，Chemometrics Intell. Lab. Syst. 83 (2) (2006) 83–90。

高扬于 2017 年在中国浙江大学获得计算机科学硕士学位。他目前在浙江大学计算机学院攻读博士学位。他的研究兴趣包括异常检测和时序数据挖掘。

王晓阳是浙江大学计算机学院计算机技术专业的研究生，并获得了浙江大学计算机科学与技术专业的工学学士学位。他的研究兴趣主要包括强化学习、深度学习和机器学习。

冯尊雷是浙江大学软件技术学院助理研究员。他从浙江大学计算机学院获得计算机科学与技术博士学位，从苏州大学获得工学学士学位。他的研究兴趣主要包括计算机视觉、图像信息处理、表示学习和医学图像分析。他在 IJCV 、NeurIPS 、 AAAI 、 TVCG 、 ACM TOMM 和 ECCV 等顶级会议和期刊上发表了多篇科学论文。他曾在 AAAI 和 PKDD 等国际会议以及国际期刊上担任过职务。

《 IEEE 电路与系统视频技术》、《信息科学》、《神经计算》、《视觉通信与图像表示》和《神经处理信函》



宋明理（ M’ 06-SM’ 13 ）于 2006 年在中国浙江大学获得计算机科学博士学位。他目前是浙江大学微软视觉感知实验室的教授。他的研究兴趣包括人脸建模和面部表情分析。他于 2004 年获得了微软研究奖学金。

王兴根于 2005 年和 2013 年分别在中国浙江大学获得计算机科学硕士和博士学位。他目前是浙江大学计算机学院的研究助理。他的研究兴趣包括分布式计算和软件性能。

王新宇于 2002 年和 2007 年分别在中国浙江大学获得计算机科学硕士和博士学位。他曾在浙江大学担任研究助理（ 2002 年至 2007 年）。他目前是浙江大学计算机学院的教授。他的研究兴趣包括流数据分析、形式方法、超大型信息系统和软件工程。

陈春目前是浙江大学计算机学院的教授。他的研究兴趣包括计算机视觉、计算机图形学和嵌入式技术。