Yan *et al. Cybersecurity* (2024) 7:45
https://doi.org/10.1186/s42400-024-00237-5

Cybersecurity

RESEARCH

Open Access

Phishing behavior detection on different blockchains via adversarial domain adaptation

Chuyi Yan[1,2], Xueying Han[1,2], Yan Zhu[1,2], Dan Du[1,2*], Zhigang Lu[1,2] and Yuling Liu[1,2]

Abstract

Despite the growing attention on blockchain, phishing activities have surged, particularly on newly established chains. Acknowledging the challenge of limited intelligence in the early stages of new chains, we propose ADA-Spear-an automatic phishing detection model utilizing *a*dversarial *d*omain *a*daptive learning which symbolizes the method's ability to penetrate various heterogeneous blockchains for phishing detection. The model effectively identifies phishing behavior in new chains with limited reliable labels, addressing challenges such as significant distribution drift, low attribute overlap, and limited inter-chain connections. Our approach includes a subgraph construction strategy to align heterogeneous chains, a layered deep learning encoder capturing both temporal and spatial information, and integrated adversarial domain adaptive learning in end-to-end model training. Validation in Ethereum, Bitcoin, and EOSIO environments demonstrates ADA-Spear's effectiveness, achieving an average F1 score of 77.41 on new chains after knowledge transfer, surpassing existing detection methods.

Keywords Blockchain, Phishing detection, Adversarial domain adaptation, Graph/network transfer learning, Hierarchical graph attention, Network security

Introduction

Since the introduction of Bitcoin (Nakamoto 2008) in 2008, blockchain and cryptocurrencies have flourished. According to CoinMarketCap (CoinMarketCap), there are now 25,853 different cryptocurrencies, with a market capitalization exceeding one hundred billion dollars. Typically, a blockchain gives rise to its own cryptocurrency, and this financial characteristic has resulted in a surge of phishing activities. Statistics from Chainalysis (Chainanalysis) reveal that since 2017, more than 50% of blockchain security incidents are linked to phishing. By 2022, the proportion of phishing incidents has consistently risen to over 80%. Consequently, there is an urgent need to research methods for detecting phishing activities across different cryptocurrencies.

Traditional phishing activities generally involve the use of fake websites to induce users to provide private information. Thus, traditional phishing detection focuses on identifying these counterfeit websites and promptly warning users against interacting with them (Jain et al. 2017; Zuraiq and Alkasassbeh 2019; Orunsolu et al. 2022). Phishing activities on the blockchain, however, have developed new patterns. Criminals have shifted their focus from stealing private information to cryptocurrencies, employing a combination of social engineering and technical methods. Upon successfully obtaining cryptocurrencies, they disguise their identities through multiple transactions, increasing the covert nature of their activities. Moreover, as different cryptocurrencies continually emerge, possessing reliable tagged data for each is extremely valuable. Newly emerged cryptocurrencies lack any tagged data, requiring a significant amount of time to accumulate relevant intelligence databases. At that point, the damage has already occurred, significantly

*Correspondence:
Dan Du
dudan@iie.ac.cn
[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

dampening user enthusiasm and hindering the development of new chains.

This paper aims to tackle the challenge of detecting phishing activities in diverse blockchain networks (target blockchain) by utilizing labeled data from source blockchains, as illustrated in Fig. 1. We addresses the issue of ineffective phishing detection on newly emerged chains during their early stages, characterized by limited on-chain annotations and differing data distributions across chains. Current detection methods cannot be directly generalized to effectively detect phishing activities on new chains, resulting in delays in halting phishing behavior during the initial stages of chain emergence. While mature chains like Ethereum possess more abundant data and established detection methods, the lack of information on target chain samples poses a challenge. To address this, we propose a adversarial domain adaptation-based method (Pan and Yang 2009; Ganin et al. 2016; Shen et al. 2018; Goodfellow et al. 2020) for phishing detection in small-sample public chains with limited annotations on the target chain named ADA-Spear.

The challenges are as follows: Firstly, current detection methods heavily depend on manual feature engineering, require substantial expert knowledge, and are unsuitable for mining deep patterns on the blockchain. Moreover, their generalizability is weak, making them challenging to apply to different chains. Secondly, there exists a substantial distribution drift (Wiles et al. 2021) between different chains, accompanied by low attribute overlap. This implies that there are differences in both the data distribution and feature space between the source and target blockchains. Therefore, the phishing address patterns on the source blockchain are difficult to directly apply to the target blockchain, leading to overfitting of the model to the features of the source chain and a subsequent decrease in the model's generalization ability. Additionally, the presence of coin mixing and other anonymity services prevents the cross-chain transfer of edge information from the source to the target chain, hindering knowledge transfer. Thirdly, in the source chain, trustworthy labels are sparse. Even in Ethereum, which has abundant labels, phishing activity labels still only account for about 0.2% of total addresses (Etherscan). This scarcity of usable information in the source chain leads in the inadequacy of full supervision learning robustness.

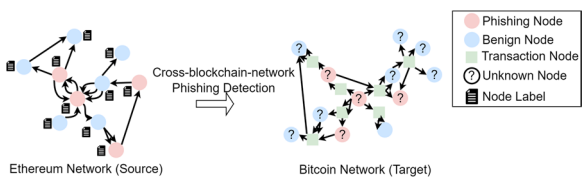

Fig. 1 Cross-blockchain-network phishing detection diagram: known features and labels of the source network are used to detect nodes with unknown labels in the target network

Building upon this, we introduce adversarial domain adaptation techniques from transfer learning and proposes a small-sample phishing detection method for public chains. Firstly, we propose a subgraph construction algorithm based on chain structure to transform heterogeneous graphs from different chains into homogeneous graphs, thereby alleviating the problem of significant data distribution drift between chains structurally. Secondly, we introduce a hierarchical representation encoder at both node and subgraph levels to capture spatial and temporal information of node behaviors, obtaining high-dimensional representations of node features. This encoder is better suited for mining deep patterns in blockchain data. Thirdly, we apply adversarial domain adaptation networks to the node representations across different chains to mitigate the low overlap of attributes and data distribution drift between the source and target chains. Simultaneously, the adversarial domain adaptation network effectively enhances knowledge transfer capability when there is no cross-chain edge for information propagation from the source to the target chain, enabling timely and effective phishing detection on new target chains with limited labels (Singh 2019; Sayadi et al. 2019; Chen et al. 2020a, b; Wu et al. 2020; Ao etal. 2021; Xu et al. 2018; Wu et al. 2020; Perozzi et al. 2014; Grover and Leskovec 2016; Wang et al. 2016, 2017; Ji et al. 2021; Heimann and Koutra 2017; Hamilton et al. 2017; Kipf and Welling 2016; Zheng et al. 2023; Yang et al. 2016; Fang et al. 2013; Ni et al. 2018; Xu et al. 2017; Pan and Yang 2009; Ganin et al. 2016; Shen et al. 2018; Goodfellow et al. 2020).

Specifically, ADA-Spear aims to characterize address behavior patterns and achieve knowledge transfer of behavior patterns from the source network to the target network, thereby facilitating the detection of phishing activities across different chains. The 'ADA' in ADA-Spear stands for **A**dversarial **D**omain **A**daptation, and 'Spear' symbolizes the method's ability to penetrate various heterogeneous blockchains for phishing detection. In the subgraph construction method, we treat the target address as the central node, and after obtaining its second-order neighbor nodes, propose a reduction strategy for adapting to the full-sample gradient descent training in neural networks. Subgraphs are established, abstracting the phishing detection into a subgraph classification task (Zhang et al. 2021; Narayanan et al. 2017). The main idea of the detection model is to learn different inter-class subgraph representations through a network encoder with multi-dimensional feature fusion and invariant subgraph representations across different chains

through adversarial domain adaptation. It should be noted that multidimensional features refer to time-based behavioral features, space-based behavioral features, and source data on blocks.Consequently, ADA-Spear consists of an encoder module that characterizes address behavior and a domain adaptation module. It should be noted that multidimensional features refer to temporal-based behavioral features, spatial-based behavioral features, and addresses' source data on blocks.

On one hand, the encoder module primarily aims to better characterize address behavior and learn class-level discriminative subgraph representations. It trains an effective classifier using existing label and considers the temporal evolution of behavior to characterize node-level feature representations and subgraph-level behavior patterns (Jiao et al. 2021). On the other hand, the adversarial domain adaptation module is designed to mitigate the distribution drift between the source and target networks so that transfer phishing pattern knowledge between chains. It employs adversarial learning to learn invariant subgraph representations across different chains, facilitating the transfer of behavior pattern knowledge from the source network to the target network. The training process is similar to that of Generative Adversarial Networks (GANs) (Goodfellow et al. 2020; Dai et al. 2018; Pan et al. 2018; Dai et al. 2019), where the encoder learns inter-chain invariance, and the domain discriminator distinguishes whether subgraph representations originate from the source or target domain. By combining these two parts, ADA-Spear can learn class-level discriminative subgraph representations and inter-chain invariant representations, facilitating the transfer of class information across different chains.

We choose Ethereum, Bitcoin, and EOSIO as three blockchain platforms, serving respectively as the source and target domains, to research cross-blockchain-network phishing behavior detection. In summary, our contributions are:

- We propose a novel approach ADA-Spear, a neural network detection model with adversarial domain adaptation and multi-dimensional feature fusion, tackling the challenge of detecting phishing activities in diverse blockchain networks.
- The proposed model ADA-Spear comprises an encoder that effectively characterizes on-chain phishing activities and an adversarial domain adaptation module. It considers the temporal-based of behavior features, spatial-based behavioral features and addresses' source data on blocks, effectively characterizing phishing activities at both node and subgraph levels. By learning high-dimensional behavioral patterns and alleviating the issue of distribution drift between different chains, ADA-Spear can detect phishing activity effectively on new target chains with few or no labels.
- We conduct detailed and extensive experiments in real Ethereum, Bitcoin, and EOSIO environments to demonstrate the effectiveness of ADA-Spear. Our method achieves an average F1-score exceeding 76% in the target domain with no labels and a label rate of 5%, surpassing existing advanced detection methods. Furthermore, we analyze phishing activities across diverse chains, highlighting the model's exceptional performance and scalability. This proves its robustness, even in instances of distribution differences between chains.

Related work

The detection of phishing activities on blockchain primarily employs feature engineering and graph analysis methods, where graph analysis can further be divided into single-network learning, graph-based semi-supervised learning, and cross-network learning methods.

Feature engineering based method

Feature engineering methods (Sayadi et al. 2019; Singh 2019) rely heavily on expert knowledge of blockchain phishing behavior, making it time-consuming and labor-intensive. Toyoda et al. (2018) use seven types of statistical information on Bitcoin to perform multi-classification of Bitcoin services, including the detection of anomalous services. Chen et al. (2020a) extract 219 features in Ethereum to identify phishing activities, representing a comprehensive feature extraction work in early feature engineering methods. Jourdan et al. (2018) build repeat patterns by extracting features from Bitcoin and use LightGBM for subsequent classification. Ostapowicz (?) and others directly utilize information recorded on the blockchain as features, exploring the performance of machine learning methods like Random Forest, Support Vector Machine (SVM), and XGBoost in fraud detection. While feature engineering-based methods can detect certain behaviors, with the advancement of blockchain technology, anomalous behaviors (including phishing activities) are becoming increasingly complex. Simple extraction of block features fails to accurately capture these more intricate behaviors. Additionally, relying on feature engineering methods is highly dependent on expert knowledge, making it difficult to obtain higher-order information about behavioral features. Additionally, the extracted features are limited to the current chain, posing challenges in applying them to new chains and resulting in low generalizability.

Graph analysis based method

Single network learning method

This is an unsupervised learning method that can learn node representations based on the network's topological structure or other information for subsequent tasks like node classification, as seen in algorithms like DeepWalk. Generally, these methods use unsupervised learning to embed network structural information, followed by training classifiers for node or subgraph classification (Wang et al. 2016, 2017; Ji et al. 2021). Yuan et al. (2020) are the first to use DeepWalk (Perozzi et al. 2014) and Node2vec (Grover and Leskovec 2016) on blockchain to learn topological representations of addresses, later using machine learning for address classification. As Etheruem phishing itself also exhibits its own behavioral characteristics, so Wu et al. (2020) and Lin et al. (2020) combine Ethereum transaction amounts and timestamps to extract node representations through biased walks, using single-class SVM and other machine learning methods for phishing node detection. Ao etal. (2021) refine time-based repeat patterns in Ethereum and propose a community detection method to extract phishing node pattern representations. However, such methods lack the ability to generalize to other networks because they do not maintain a similarity structure that "brings closer" the embeddings of nodes or subgraphs of the same category across different networks (Heimann and Koutra 2017). Additionally, their classification effectiveness is not as robust as graph-based semi-supervised learning methods. Therefore, these methods are also not applicable for detecting cross-chain phishing behaviors.

Graph-based semi-supervised learning method

To further improve the classification of phishing activities, later methods often use graph-based semi-supervised learning (Hamilton et al. 2017; Kipf and Welling 2016; Zheng et al. 2023). Since blockchain's data structure and massive transaction data are best modeled as a graph structure, they are trained in a supervised manner from a graph perspective. Patel et al. (2020) use graph convolution network (GCN) to extract structural information in Ethereum and classify fraud nodes using a single-class SVM. This method enhances the accuracy of phishing detection on the blockchain. To further enhance the extraction of structural information, Chen et al. (2020a) modify the initial features of GCN, effectively improving classification accuracy. Li et al. (2020) consider the topological structural differences between fraudulent and benign addresses in the Bitcoin network and use XGBoost for address classification. Zhang et al. (2021) use a hierarchical approach to extract Ethereum's topological structure for subgraph representation, training neural networks end-to-end for phishing subgraph identification. This method extends the dimensionality of address analysis. Subsequent studies further consider temporal behavior, with Li et al. (2022) constructing Ethereum as a multi-graph, transforming multiple edges into sequences, and embedding them into LSTM to obtain temporal representations of phishing nodes, effectively improving detection accuracy. Graph-based semi-supervised learning methods adopt an end-to-end approach, integrating network structure, node attributes, and labels, demonstrating effective classification results in single networks. However, these methods face challenges in generalizing to other networks experiencing significant distribution drift (Yang et al. 2016).

Cross-network learning method

Currently, blockchain phishing detection methods lack generalizability and do not employ cross-blockchain-networklearning models. However, to promptly detect phishing activities on new chains, research on cross-blockchain-network phishing detection methods is necessary. Existing methods utilizing source network information to assist in identifying target network behavior include: Fang et al. (2013) propose the NetTr method, which only transfers topological information in networked data, but it is computationally expensive and transfers limited information. Shen et al. (2020) propose CDNE, learning representations of nodes within multiple networks and minimizing the Maximum Mean Discrepancy (MMD) loss function to mitigate distribution drift between nodes in different networks. To further alleviate domain drift during knowledge transfer, Ganin et al. propose the domain adaptation method DANN (Ganin and Lempitsky 2015), learning inter-domain invariance through a min-max game approach. Building on DANN, Shen et al. (2018) use Wasserstein distance to quantify inter-domain differences for better gradient properties, thereby improving transfer effectiveness. Current methods that leverage edge information across networks to improve classification typically involve learning embeddings across multiple networks, but heavily rely on inter-network connections. Although cross-chain transactions can occur between different blockchains, establishing inter-network edges becomes challenging after passing through cross-chain bridges or mixing services, rendering such methods unsuitable for the problem addressed in our study. For the issue addressed in this paper, which involves phishing detection on new chains with limited annotations, cross-network transfer learning algorithms are appropriate. These algorithms primarily address the adverse effects on models caused by domain shift during knowledge transfer from the source domain to the target domain. This paper will use adversarial domain

adaptation based on graph networks to generalize the model to different chains and promptly detect phishing activities.

Preliminaries

This section mainly introduces the definition and feature construction of on-chain interaction graphs and the definition of the issues studied in this paper. Table 1 shows the main symbols used in our framework.

On-chain interaction graph

This subsection defines the Ethereum, Bitcoin and EOSIO networks modeled as graph structures and gives the Graph definition after subtraction.

Ethereum Attribution Graph (E-AIG) A directed, weighted, homogeneous multigraph $G = (V^e, E^e, \mathbf{A}^e, \mathbf{F}_v^e, \mathbf{F}_e^e, Y^e)$, where V^e, E^e, and $\mathbf{A}^e$ respectively represent the sets of nodes, directed edges, and the adjacency matrix. The edge set E^e is defined as $E^e = \{(v_i, v_j, \mathbf{F}_e^e) | v_i, v_j \in V^e\}$. $\mathbf{F}_v^e$ and $\mathbf{F}_e^e$ respectively represent the feature matrices for Ethereum nodes and edges, as detailed in Sect. 3.2. Some nodes in the E-AIG are labeled with $y_i \in Y^e$, where $Y^e = \{(v_i, y_i) | v_i \in V^e\}$.

EOSIO Address Interaction Graph (EO-AIG) The definition is the same as for the E-AIG and will not be repeated.

Bitcoin Address-Transaction Interaction Graph (B-ATIG) A directed, weighted, heterogeneous multigraph $G = (V_a^b, V_t^b, E^b, \boldsymbol{F}_v^b, \boldsymbol{F}_e^b, Y^b)$, where V_a^b and V_t^b respectively represent the sets of address nodes and transaction nodes, E^b represents the set of edges, defined as $E^b = \{(v_i, v_j, \boldsymbol{F}_e^b) | v_i \in V_a^b \cup V_t^b, v_j \in V_a^b \cup V_t^b\}$. $\boldsymbol{F}_v^b$ and $\boldsymbol{F}_e^b$ respectively represent the feature matrices for Bitcoin address nodes and edges, as detailed in Sect. 3.2. Some address nodes in the B-ATIG are labeled with $y_i \in Y^b$, where $Y^b = \{(v_i, y_i) | v_i \in V_a^b\}$.

As the original B-ATIG is a heterogeneous graph, to align with the graph network of E-AIG and reduce computational complexity, the strategy proposed in Sect. 4.1 is used to remove transaction nodes from B-ATIG, converting it into a homogeneous graph B-AIG.

Bitcoin Address Interaction Graph (B-AIG) A directed, weighted, homogeneous multigraph $G = (V^b, \widetilde{E}^b, \boldsymbol{A}^b, \boldsymbol{F}_v^b, \widetilde{\boldsymbol{F}}_e^b, Y^b)$, where V^b, $\widetilde{E}^b$, and $\boldsymbol{A}^b$ are the sets of address nodes, transformed set of directed edges, and adjacency matrix, respectively. The transformed edge set $\widetilde{E}^b$ is defined as $\widetilde{E}^b = \{(v_i, v_j, \widetilde{\boldsymbol{F}}_e^b) | v_i, v_j \in V_a^b\}$, and $\widetilde{\boldsymbol{F}}_e^b$ is the edge feature matrix after the removal of transaction nodes.

E-AIG, EO-AIG, and B-AIG are all dense connected homogeneous multigraphs, which increases the complexity during subsequent knowledge transfer. Thus, through feature reconstruction, interaction aggregation, and node reduction as described in Sect. 4.1, the directed graphs r-AIG (E-AIG→rE-AIG, EO-AIG→rEO-AIG, B-AIG→rB-AIG) are obtained.

Reduced Address Interaction Graph (r-AIG) A directed, weighted, homogeneous graph $G = (V^r, E^r, \boldsymbol{A}^r, \boldsymbol{F}_v^r, \boldsymbol{F}_e^r, Y^r)$, where V^r is the set of nodes after the TopK reduction described in Sect. 4.1, $E^r = \{(v_i, v_j, \boldsymbol{F}_e^r) | v_i, v_j \in V^r\}$ is the remaining set of edges, and $\boldsymbol{A}^r$ is the adjacency matrix. $\boldsymbol{F}_v^r$ and $\boldsymbol{F}_e^r$ are the feature matrices for the nodes V^r and edges E^r, respectively.

Table 1 Main symbols used in our framework

Symbols	Expression
G^s, G^t	Graph in the source and target domain
V^s, V^t	Nodes in the source and target domain
E^s, E^t	Edges in the source and target domain
A^s, A^t	Adjacency matrix in the source and target domain
${F_v}^s, {F_v}^t$	Nodes' feature set in the source and target domain
${F_e}^s, {F_e}^t$	Edges' feature set in the source and target domain
Y^s	Source domain label set
s_{disc}	Domain discriminator training set
γ, λ	Coeffeicients in the loss function
$\alpha_{disc}, \alpha_{encod}$	Learning rates of discriminator and encoder
N^s, N^t	The number of subgraphs in the source and target domian
$f_{encod}, f_{pred}, f_{disc}$	Optimal encoder, semi-supervised learning classifier and domain discriminator
$\theta_{encod}, \theta_{pred}, \theta_{disc}$	Parameters for encoder, clssifier and discriminator
H^s, H^t	Addresses representation in the source and target domian
$\mathcal{L}_{pred}, \mathcal{L}_{disc}, \mathcal{L}_{penal}$	The loss function of classifier, domain discriminator and penalty factor

Some nodes in r-AIG are labeled with $y_i \in Y^r$, where $Y^r = \{(v_i, y_i)|v_i \in V^r\}$.

Feature construction

Node features and edge features in Ethereum, Bitcoin, and EOSIO are categorized into three types: transaction time, transaction amount, and transaction count. Each type is differentiated by direction and further classified into incoming and outgoing transactions. In multigraphs, node features primarily include overall subgraph information like lifespan, total amount, degree, and the number of active nodes. Edge features mainly encompass information for each transaction, including block number, timestamp, transaction amount, and transaction fees. For transaction amount-type features, the maximum, minimum, and average functions will be used. Transaction amount-type features will utilize the maximum, minimum, and average functions. Similarly, transaction count-type features, based on the same approach, are combined with transaction amount-type features to derive per-transaction average features.

Problem definition

The primary focus of this paper is on detecting cross-blockchain-network phishing behavior through domain adaptation. Specifically, it leverages knowledge of phishing behavior in the source domain to assist in recognizing phishing behavior in a new target domain. This process transforms the phishing behavior recognition challenge into a graph classification problem.

The source network is denoted as $G^s = (V^s, E^s, \boldsymbol{A}^s, \boldsymbol{F}_v^s, \boldsymbol{F}_e^s, Y^s)$, where $V^s = \{v_1, v_2, \ldots, v_{n^s}\}$ represents the set of address nodes, $E^s \subseteq \{(v_i, v_j)|v_i, v_j \in V^s\}$ represents the set of interaction edges, $\boldsymbol{A}^s \in \mathbb{R}^{n^s \times n^s}$ represents the adjacency matrix, $\boldsymbol{F}_v^s \in \mathbb{R}^{n^s \times c_n^s}$ represents the node feature matrix, with c_n^s being the number of features for each node, $\boldsymbol{F}_e^s \in \mathbb{R}^{m^s \times c_e^s}$ represents the edge feature matrix, with $m^s = |E^s|$ and c_e^s being the number of features for each edge, $Y^s = \{(v_i, y_i)|v_i \in V^s\}$ represents the label set for some labeled address nodes, where if $y_i = 1$, the node is a phishing node, otherwise $y_i = 0$. Similarly, the target network is denoted as $G^t = (V^t, E^t, \boldsymbol{A}^t, \boldsymbol{F}_v^t, \boldsymbol{F}_e^t)$. In this approach, r-AIG is used as both the source and target networks.

A subgraph centered around v^r is defined as $g_{v^r} \subset G^r$, where $r \in \{s, t\}$, the source domain is $D^s = (G^s, f(g_{v^r}^s))$, and the target domain is $D^t = (G^t, f(g_{v^r}^t))$, with $D^s \neq D^t$, and $f(g)$ being the subgraph classification task. The problem studied in this paper is, given a distributional difference but identical label sets between source and target domains, to learn a classification function $f(g) \mapsto y_i$ with the aid of source domain information, in order to accurately identify the category of $g_{v^r}^t$ in the target domain.

Cross-blockchain-network phishing behavior detection method

In this section, we offer a comprehensive overview of the ADA-Spear phishing detection method's structure, as illustrated in Fig. 2. ADA-Spear is applicable both for phishing behavior recognition and mining within the same blockchain, as well as for cross-blockchain-network phishing detection. When examining a node v, ADA-Spear takes the r-AIG g_v centered around v as input and outputs the node's label. A label of 1 indicates a phishing node, while 0 represents a benign node. ADA-Spear comprises the following components: Firstly, to characterize on-chain interaction behavior, an r-AIG centered around v is constructed. Secondly, to depict the complex behavior of node v, an r-AIG encoder is designed, incorporating temporal feature analysis and attention mechanisms. Thirdly, an adversarial domain adaptation transfer

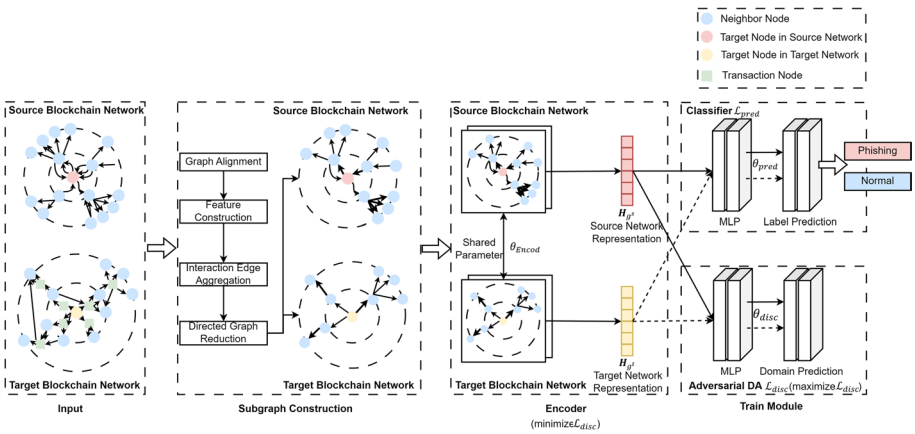

Fig. 2 The framework of cross-blockchain-network phishing behavior detection method

learning module is developed, adapting the encoder to the address behavior representation on the target chain and identifying phishing behavior on the target chain. Finally, a training module is designed, integrating domain adaptive transfer learning with binary classification tasks for training.

Subgraph construction

Ethereum and Bitcoin data both reach million-level scales, resulting in excessively large graph data for their respective AIGs. This poses a challenge for full-batch training in graph nerual network (GNN)-based neural networks. Furthermore, an abundance of redundant data hinders effective transfer learning, making the migration of source domain features to the target domain challenging. This may result in issues like overfitting, impacting the effectiveness of phishing detection. Consequently, this subsection aligns and simplifies E-AIG/EO-AIG and B-ATIG, creating a unified r-AIG. The subgraph construction process is illustrated in Fig. 3.

Sampling original data into multigraphs

In this subsection, the second-order nodes of the target address are sampled from the original block data of Ethereum and EOSIO to create the subgraph representing the target address, resulting in the multigraph E-AIG. Given Bitcoin's distinct organizational structure from Ethereum, the original block data for Bitcoin involves sampling the second-order transaction nodes of the target address and the next-order addresses corresponding to these transactions. This process results in the creation of a heterogeneous graph containing transaction nodes and address nodes, forming the multigraph B-ATIG.

The design of the sampling method outlined above is driven by two key considerations: a. Phishing and benign nodes exhibit distinct behavioral patterns, manifested in their interactions with neighboring nodes, i.e., local structural features on the graph; b. Subgraphs, in comparison to the full graph, undergo a significant reduction in scale and are adaptable to the training method of the model network.

To facilitate adaptation to the later-designed encoder and adversarial domain adaptive learning, aligning EAIG/EO-AIG with B-ATIG is essential. This involves transforming B-ATIG into B-AIG. The transformation process is shown in Fig. 4. In the heterogeneous graph B-ATIG, all nodes of type "transaction" $n_{tx} \in N_{tx}$ and their corresponding edge pairs $(n_{tx_{pre}}, n_{tx})$, $(n_{tx}, n_{tx_{next}})$ are extracted. By removing n_{tx}, the edge $(n_{tx_{pre}}, n_{tx_{next}})$ is formed. The attributes of the edge $(n_{tx}, n_{tx_{next}})$ are retained as new edge attributes. This process concludes with the construction of the homogeneous multigraph B-AIG.

Node and edge feature construction

Following the analysis of phishing behaviors, features exhibit strong correlations with amount, time, frequency, and degrees. The feature construction

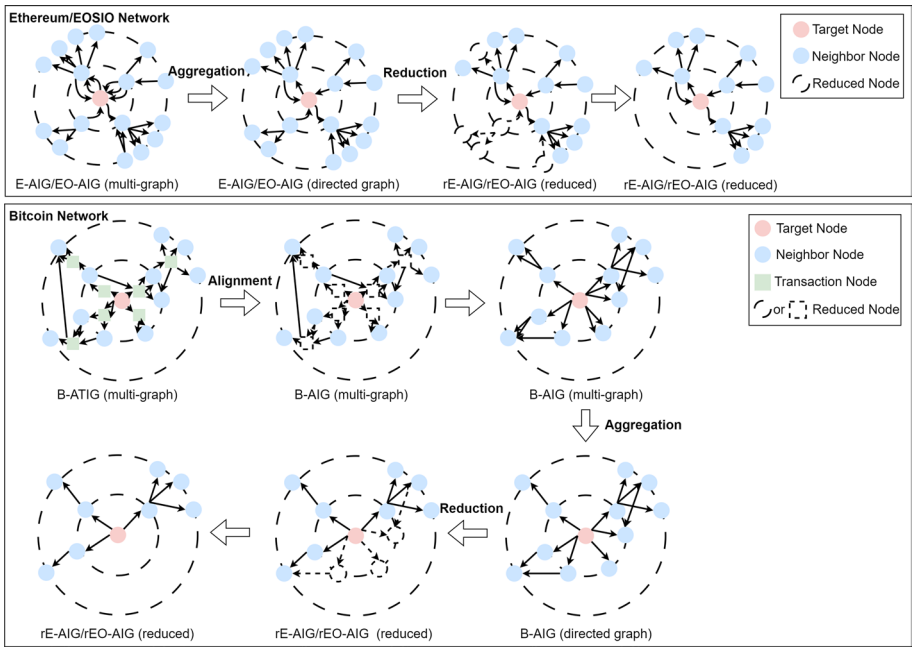

Fig. 3 ADA-Spear subgraph construction flowchart

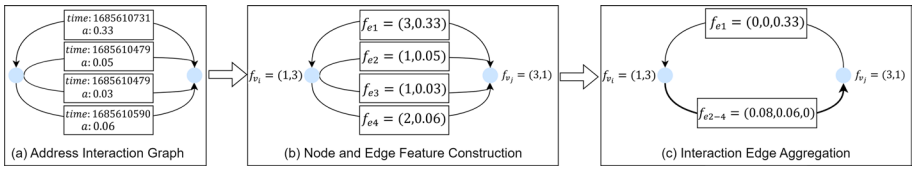

Fig. 4 ADA-Spear directed graph construction flow chart

schematic is shown in Fig. 4a, b. In the AIGs formed by the three chains, node attributes are denoted $f_v \in F_v$. Specifically, $f_v = (\text{in}, \text{out})$, $f_e = (t, a)$ where in and out represent the in-degree and out-degree of the node, respectively. Edge attributes are represented as $f_e = (t, a), f_e \in F_e$, where t indicates the order of the transaction occurrence in the subgraph. Specifically, $t = \text{rank}(\text{time}, g_i)$, $i \in \{1, 2, \ldots, \text{count}\}$, with time being the transaction timestamp, g_i being the subgraph containing the target node, and count being the total number of nodes. Additionally, a represents the transaction amount type attribute.

Merging interactions into a directed graph

To transform multigraphs into directed graphs, edge aggregation is employed, as shown in Fig. 4b, c. For clarity, the figure only displays the aggregation method for the sum of amounts. Given the temporal characteristics of phishing behaviors, the graph preserves crucial temporal information of transactions to the fullest extent. Multiple parallel edges between two nodes are aggregated into a single edge, and the attributes on the edge (v_i, v_j) are $((\Sigma a_{t1}, \text{Max}(a_{t1}), \text{Min}(a_{t1}), \text{Mean}(a_{t1})), \ldots, (\Sigma a_{t_M}, \text{Max}(a_{t_M}), \text{Min}(a_{t_M}), \text{Mean}(a_{t_M})))$ where Σa_{tM}, $\text{Max}(a_{tM})$, $\text{Min}(a_{tM})$, $\text{Mean}(a_{tM})$, $m \in M$ respectively represent the sum, maximum, minimum, and mean of all transaction amounts at the mth moment, and M indicates the maximum occurrence sequence of transactions in the subgraph.

It should be noted that different target nodes will form different subgraphs, and the number of transactions within different subgraphs may vary. Therefore, the dimensions of the edges in different subgraphs may differ after aggregation. Subsequently, the issue of dimension alignment will be addressed by using a variable-length long short-term memory (LSTM).

Graph reduction with directed edges

Although sampling second-order nodes reduces the data volume compared to the full blockchain interaction graph, the data volume remains significant. Therefore, further reduction of the directed AIG, as shown in Fig. 5, occurs while retaining information relevant to phishing detection. In this subsection, the TopK technique is utilized to sample nodes at each order. The

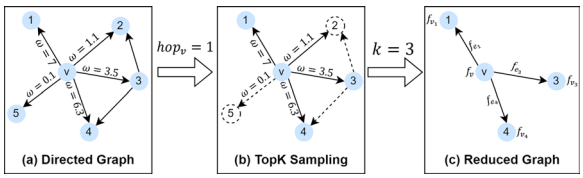

Fig. 5 ADA-Spear directed graph reduction flow chart

strategy is based on the following considerations: When comparing nodes of the same order, nodes with fewer zeros in the feature vector of the edges they belong to (i.e., fewer zeros), a larger sum of all amounts in the vector (i.e., larger sum), and higher degrees (i.e., higher degree) are more important and have a higher probability of being retained. In each order, the weight at which each node is retained, denoted as w, can be expressed as:

$$w(v_i) = rank((c_{zero} + a_{all} + d), hop_{v_i}, g_{v_i}) \tag{1}$$

where each node utilizes the sum of all incoming edges and outgoing edges $c_{zero} + a_{all} + d$, hop_{v_i} denotes the current order of node v_i, g_{v_i} represents the subgraph where v_i belongs. After the final iteration and reduction, the resulting node set can be represented as:

$$\begin{aligned} V^r = &\cup_{v_i \in V^r} TopK(G = g_{v_i}, K, w(v_i), hop), \\ &hop \in \{0, 1, 2\} \end{aligned} \tag{2}$$

In conclusion, a reduced sub-directed graph is obtained. For the reduction process in this paper, we choose $K = 25$.

Encoder network

This section presents a comprehensive overview of the ADA-Spear model's feature extractor $f_{encod}(\boldsymbol{A}, \boldsymbol{F}_v, \boldsymbol{F}_e; \boldsymbol{\theta}_{encod})$. It employs a hierarchical attention mechanism based on LSTM to map the behavioral features of target nodes into representation vectors for subsequent transfer learning. The schematic diagram is illustrated in Fig. 6. Additionally, the encoder, in conjunction with the prediction head described in the next section, can also be applied for phishing detection within the same blockchain domain. In the following, the specific details of the encoder will be presented.

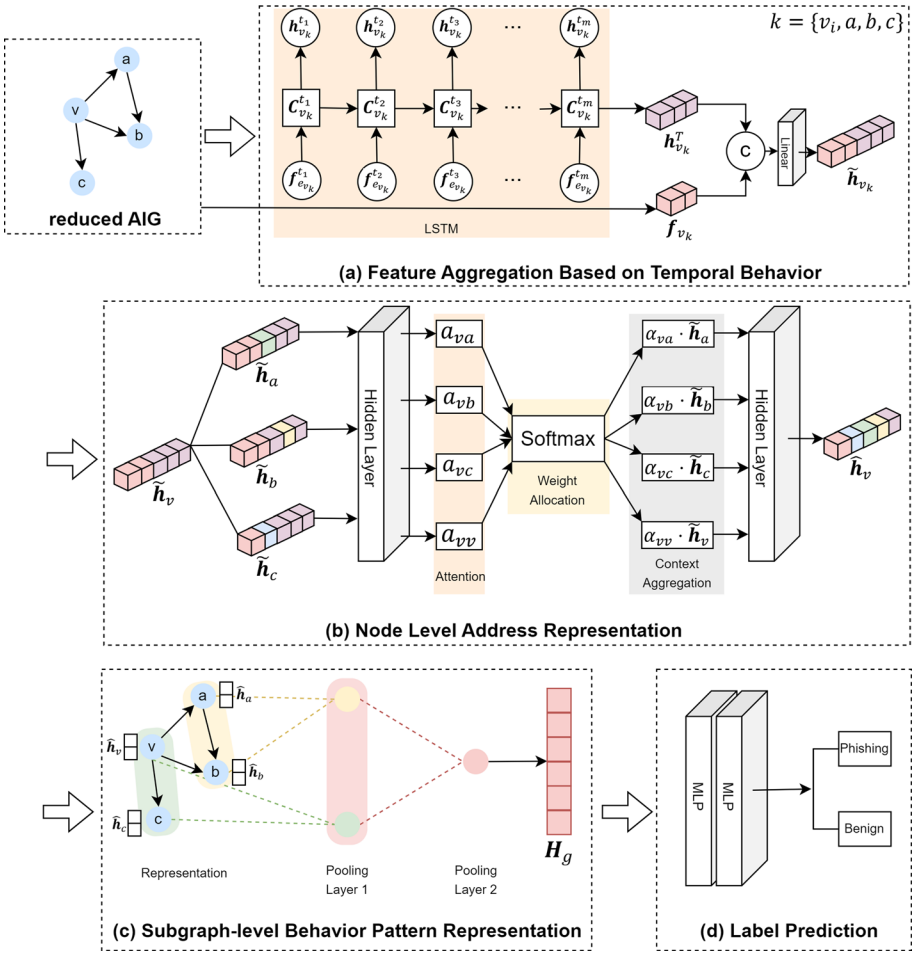

Fig. 6 The architecture of ADA-Spear encoder network

Node level addreses representation based on temporal and spatial behavior

Node Level Address Representation Based on Temporal Behavior In the previously discussed r-AIG, nodes are encoded considering transaction flow and frequency, while edges are encoded based on transaction occurrence order. Since AIG is a graph centered around addresses, and each node has both incoming and outgoing transactions with its neighboring nodes, it is essential to preserve the temporal characteristics of transaction information. Therefore, in this subsection, the edge features between the central node v_i and its neighboring nodes are aggregated into node v_i. The input vector of the central node v_i is represented as $\boldsymbol{f}'_{v_i} = [(f^{t_1}_{e\_in}, f^{t_1}_{e\_out}), (f^{t_2}_{e\_in}, f^{t_2}_{e\_out}), \ldots (f^{t_m}_{e\_in}, f^{t_m}_{e\_out})]$. $\boldsymbol{f}'_{v_i}$ serves as the input for a variable-length LSTM, resulting in embedded vectors $\boldsymbol{h}^t_{v_i}$. This operation is performed for all nodes on the graph, resulting in $\boldsymbol{h}^t_{v_i}$, where v_j represents the neighbors of v_i. Since the total number of time steps m may vary between different subgraphs, a variable-length LSTM is employed to adapt to m. Figure 6a presents a visualization of this process.

Specifically, for any node $v_k, k \in \{i, j\}$, $\boldsymbol{f}'_{v_k}$ is used as the LSTM input, and the forgetting gate and input gate are calculated as follows:

$$\boldsymbol{fg}^t_{v_k} = \sigma(\boldsymbol{W}^{fg}_{v_k}[\boldsymbol{h}^{t_{h-1}}_{v_k}, (f^{t_h}_{e_{v_k}\_in}, f^{t_h}_{e_{v_k}\_out})] + \boldsymbol{b}^{fg}_{v_k}) \tag{3}$$

$$\boldsymbol{i}^{t_h}_{v_k} = \sigma(\boldsymbol{W}^{i}_{v_k}[\boldsymbol{h}^{t_{h-1}}_{v_k}, (f^{t_h}_{e_{v_k}\_in}, f^{t_h}_{e_{v_k}\_out})] + \boldsymbol{b}^{i}_{v_k}) \tag{4}$$

Then, we calculate the current candidate cell state:

$$\widetilde{\boldsymbol{C}}^{t_h}_{v_k} = tanh(\boldsymbol{W}^{C}_{v_k}[\boldsymbol{h}^{t_{h-1}}_{v_k}, (f^{t_h}_{e_{v_k}\_in}, f^{t_h}_{e_{v_k}\_out})] + \boldsymbol{b}^{C}_{v_k}) \tag{5}$$

Combine the forget gate and input gate to update the current cell state as follows:

$$\boldsymbol{C}^{t_h}_{v_k} = \boldsymbol{fg}^{t_h}_{v_k} \boldsymbol{C}^{t_{h-1}}_{v_k} + \boldsymbol{i}^{t_h}_{v_k} \widetilde{\boldsymbol{C}}^{t_h}_{v_k} \tag{6}$$

At time step t, the current cell's output hidden layer is expressed as:

$$\boldsymbol{o}_{v_k}^{t_h} = \sigma(\boldsymbol{W}_{v_k}^{o}[\boldsymbol{h}_{v_k}^{t_{h-1}}, (f_{e_{v_k}\_in}^{t_h}, f_{e_{v_k}\_out}^{t_h})] + \boldsymbol{b}_{v_k}^{o}) \tag{7}$$

$$\boldsymbol{h}_{v_k}^{t_h} = \boldsymbol{o}_{v_k}^{t_h} tanh(\boldsymbol{C}_{v_k}^{t_h}) \tag{8}$$

Here, σ represents the sigmoid activation function, and t takes values from the set $\{t_1, t_2, \ldots, t_m\}$.

Node Level Address Representation Based on Address Source Data on Blocks Subsequently, the node features $\boldsymbol{f}_{v_k}$ are concatenated with $\boldsymbol{h}_{v_k}^{T} = (\boldsymbol{h}_{v_k}^{t_1}, \boldsymbol{h}_{v_k}^{t_2}, \boldsymbol{h}_{v_k}^{t_3}, \ldots, \boldsymbol{h}_{v_k}^{t_m})$, resulting in $\widetilde{\boldsymbol{h}}_{v_k} = [\boldsymbol{f}_{v_k} \| \boldsymbol{h}_{v_k}^{T}]$. The same process is applied to other neighboring nodes. At this point, each node has aggregated edge features into the node and has fused them with the original node features, preserving transaction information based on time sequences.

Node Level Address Representation Based on Spatial Behavior After obtaining the features of each node, this step aims to retain the most relevant neighbor interaction information between nodes. In phishing behavior, each neighboring node contributes differently to phishing detection. For example, a phishing address may engage in small transactions with benign addresses to obfuscate its malicious behavior, which can interfere with the detection process and should be excluded. Leveraging this insight, the paper utilizes the graph attention mechanism (GAT) mechanism to understand the distinct contributions of each neighbor node to phishing behavior detection and learn the hidden layer representation between each pair of nodes. The process is shown in Fig. 6a.

Specifically, for any node v_i in the subgraph, the attention score between it and its neighbor node v_j is calculated as follows:

$$a_{i,j} = \text{LeakyRelu}(\boldsymbol{W}_a \widetilde{\boldsymbol{h}}_{v_i}, \boldsymbol{W}_a \widetilde{\boldsymbol{h}}_{v_j}) \tag{9}$$

where $\boldsymbol{W}_a$ represents the weight to be learned in a single-layer feedforward network, and LeakyRelu is used as a non-linear activation layer for subsequent normalization operations.

To ensure that the attention scores between nodes are comparable, a normalization operation is performed:

$$\alpha_{i,j} = \text{softmax}(a_{i,j}) = \frac{e^{a_{i,j}}}{\sum_{k \in N(i)} e^{a_{i,k}}} \tag{10}$$

where $N(i)$ represents all neighboring nodes of node v_i.

Neighboring nodes are aggregated based on their attention scores to obtain the representation of node v_i:

$$\widehat{\boldsymbol{h}}_{v_i} = \sigma\left(\alpha_{i,i} \boldsymbol{W}_\alpha \widetilde{\boldsymbol{h}}_{v_i} + \sum \alpha_{i,j} \boldsymbol{W}_\alpha \widetilde{\boldsymbol{h}}_{v_j}\right) \tag{11}$$

where $\boldsymbol{W}_\alpha$ represents the weight of the linear layer to be learned, and σ is a non-linear activation function, with ReLU chosen in this case. After iteration, the node-level embedding $\widehat{\boldsymbol{h}}_{v_i}$ is obtained for each node.

The specific iteration process is shown in Fig. 7. Iteration is performed using the graph attention layer at the *hop* level, which involves propagating, transforming, and aggregating the representations between nodes in each subgraph. This process allows the interaction behavior of nodes at each level to be fully embedded into vectors, where *hop* represents the order of each subgraph. The initial input to the iteration layer is the node-level representations obtained from the temporal feature extraction, denoted as $\widehat{h}_{v_i}^{0} = \widehat{h}_{v_i}$, where the neighboring nodes include all first-order nodes in the subgraph. After *hop* iterations, the final node-level representations for all nodes in a subgraph, denoted as $\widehat{H}_g^{hop} = \{\widehat{h}_{v_i}\}_{v_i \in V_g}$, contain all the information about second-order nodes in that subgraph.

Subgraph-level behavior pattern representation

This module aims to characterize the behavior patterns of each target subgraph. The identity and methods of the phisher can lead to varied subgraph differences in phishing behavior on Ethereum. Therefore, it is necessary to design subgraph-level feature characterization. To overcome the limitations of flat GNNs), this subsection combines DiffPool technology (Ying et al. 2018) hierarchically to aggregate subgraph information, as shown in Fig. 6c.

Specifically, the representation $\boldsymbol{H}_g^{(0)} = \widehat{\boldsymbol{H}}_g^{hop}$ and the subgraph adjacency matrix $\boldsymbol{A}_g^{(l)}$ are used as inputs to the DiffPool layer, which calculates the representation matrix $\boldsymbol{Z}_g^{(l)}$ and the assignment matrix $\boldsymbol{S}_g^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$, where n_l is the number of nodes in the lth DiffPool layer:

$$\begin{aligned} \boldsymbol{Z}_g^{(l)} &= GNN_{represent}(\boldsymbol{A}_g^{(l)}, \boldsymbol{H}_g^{(l)}) \\ &= ReLU(\widetilde{\boldsymbol{D}}_g^{(l)^{-\frac{1}{2}}} \widetilde{\boldsymbol{A}}_g^{(l)} \widetilde{\boldsymbol{D}}_g^{(l)^{-\frac{1}{2}}} \boldsymbol{H}_g^{(l-1)} \boldsymbol{W}_{GNN}^{(l-1)}) \end{aligned} \tag{12}$$

$$\boldsymbol{S}_g^{(l)} = softmax(GNN_{pool}(\boldsymbol{A}_g^{(l)}, \boldsymbol{H}_g^{(l)})) \tag{13}$$

where $\widetilde{\boldsymbol{A}}_g^{(l)} = \boldsymbol{A}_g^{(l)} + I$, $\widetilde{\boldsymbol{D}}_g^{(l)} = \sum_j \widetilde{\boldsymbol{A}}_{ij}$, and $\boldsymbol{W}_{GNN}^{(l-1)}$ are learnable parameters.

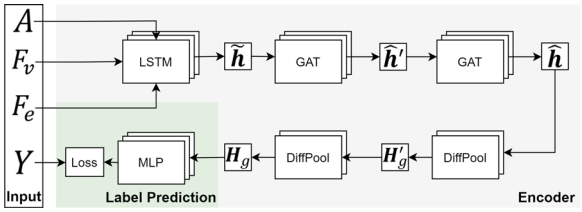

Fig. 7 The process of encoder iteration

After obtaining $\boldsymbol{Z}_g^{(l)}$ and $\boldsymbol{S}_g^{(l)}$, we can calculate the representation $\boldsymbol{H}_g^{(l+1)}$ and the adjacency matrix $\boldsymbol{A}_g^{(l+1)}$ for the $(l+1)$th layer as follows:

$$\boldsymbol{H}_g^{(l+1)} = \boldsymbol{S}_g^{(l)\,T} \boldsymbol{Z}_{\boldsymbol{g}}^{(l)}, \boldsymbol{H}_g^{(l+1)} \in \mathbb{R}^{n_{l+1} \times d} \tag{14}$$

where d is the number of columns in $\boldsymbol{Z}_g^{(l)}$, which is the feature dimension of nodes in the lth layer.

$$\boldsymbol{A}_g^{(l+1)} = \boldsymbol{S}_g^{(l)\,T} \boldsymbol{A}_g^{(l)} \boldsymbol{S}_g^{(l)}, \boldsymbol{A}_g^{(l+1)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}} \tag{15}$$

We use two layers of DiffPool to characterize subgraph behavior and finally obtain the subgraph-level behavior pattern representation $\boldsymbol{H}_g$, which serves as the output of the encoder.

Label prediction

Both the source domain and target domain use the methods mentioned in Sect. 4.2 to represent subgraphs as $\boldsymbol{H}_g^r, r \in \{s, t\}$. The semi-supervised learning classification module is designed to perform binary classification separately on the source and target domains, preparing for the subsequent domain adaptation module.

Specifically, we employ a multi-layer perceptron $f_{pred}(\boldsymbol{H}; \boldsymbol{\theta}_{pred})$, where $\boldsymbol{H}$ is the representations of all subgraphs, and $\boldsymbol{\theta}_{pred}$ represents the trainable parameters of the classifier. The predicted labels can be expressed as:

$$\widehat{\boldsymbol{y}}^r = f_{pred}(\boldsymbol{H}_g^r; \boldsymbol{\theta}_{pred}), r \in \{s, t\} \tag{16}$$

Subsequently, we use the cross-entropy loss function for the training of the classifier f_{pred}:

$$\mathcal{L}_{pred} = -\frac{1}{n^s} \sum_{i=1}^{n^s} y_i^s log \widehat{y}_i^s \tag{17}$$

Similarly, this loss function can also be computed on the target domain.

The specific iterative process of the address encoding encoder is as Fig. 7. Subgraphs' $\boldsymbol{A}$, $\boldsymbol{F}_v$ and $\boldsymbol{F}_e$ serve as inputs to the LSTM, resulting in the representation $\widetilde{\boldsymbol{h}}$. After passing through two layers of GAT and two layers of DiffPool, the address representation $\boldsymbol{H}_g$ is obtained. Multi-layer perceptron (MLP) is employed for address representation classification. The original label set $\boldsymbol{Y}$ is used as the input to the loss function, which is then compared with the labels predicted by MLP to update the encoder's parameters.

Adversarial domain adaptation

We use a adversarial domain adaptation module to eliminate the divergence between the Ethereum, Bitcoin, and EOSIO networks, facilitating knowledge transfer between these networks. For each subgraph in each network, we obtain subgraph representations $\boldsymbol{H}_g^r, r \in \{s, t\}$ through the encoder. To generate similar representations in the source and target domains, we use a fully connected layer as the domain discriminator, denoted as $f_{disc}(\boldsymbol{H}_g^r; \boldsymbol{\theta}_{disc})$, with input $\boldsymbol{H}_g^r = f_{encod}[(\boldsymbol{A}^r, \boldsymbol{F}_v^r, \boldsymbol{F}_e^r; \boldsymbol{\theta}_{encod})]_g$, representing the representation of subgraph g, and $\boldsymbol{\theta}_{encod}$ as trainable parameters. The output is a real number indicating the similarity between the source and target domains.

Next, we let the generator f_{encod} and domain discriminator f_{disc} play against each other, making it impossible for the domain discriminator to distinguish the domain of the samples. Specifically, we first compute the optimal transportation distance also named Wasserstein distance (Arjovsky et al. 2017; Gulrajani et al. 2017) between the source and target domain distributions:

$$\begin{aligned} W_1(\mathbb{P}_{\boldsymbol{H}^s}, \mathbb{P}_{\boldsymbol{H}^t}) = & \sup_{\|f_{disc}\|_{L_c} \leq 1} \mathbb{E}_{\mathbb{P}_{\boldsymbol{H}^s}}[f_{disc}(\boldsymbol{H}; \boldsymbol{\theta}_{disc})] \\ & - \mathbb{E}_{\mathbb{P}_{\boldsymbol{H}^t}}[f_{disc}(\boldsymbol{H}; \boldsymbol{\theta}_{disc})] \end{aligned} \tag{18}$$

where $\|f_{disc}\|_{L_c} \leq 1$ enforces the Lipschitz continuity condition on the domain discriminator to prevent gradient explosions or vanishing between the generator and domain discriminator, and *sup* represents the supremum. Furthermore, the optimal transportation distance maximizes the domain discriminator loss function under this condition, where the domain discriminator loss function is given by:

$$\mathcal{L}_{disc} = -\mathbb{E}_{\mathbb{P}_{\boldsymbol{H}^s}}[f_{disc}(\boldsymbol{H}; \boldsymbol{\theta}_{disc})] + \mathbb{E}_{\mathbb{P}_{\boldsymbol{H}^t}}[f_{disc}(\boldsymbol{H}; \boldsymbol{\theta}_{disc})] \tag{19}$$

This loss encourages the domain discriminator to correctly classify the source and target domain representations while the generator aims to generate domain-invariant representations. The overall objective is to minimize $\mathcal{L}_{disc}$ while maximizing $W_1(\mathbb{P}_{\boldsymbol{H}^s}, \mathbb{P}_{\boldsymbol{H}^t})$, which promotes domain adaptation and similarity between the source and target domain representations.

$$\begin{aligned} \mathcal{L}_{disc} = & \frac{1}{n^s} \sum_{i=1}^{n^s} f_{disc}([f_{encod}(\boldsymbol{A}^s, \boldsymbol{F}_v^s, \boldsymbol{F}_e^s; \boldsymbol{\theta}_{encod})]_i; \boldsymbol{\theta}_{disc}) \\ & - \frac{1}{n^t} \sum_{i=1}^{n^t} f_{disc}([f_{encod}(\boldsymbol{A}^t, \boldsymbol{F}_v^t, \boldsymbol{F}_e^t; \boldsymbol{\theta}_{encod})]_i; \boldsymbol{\theta}_{disc}) \end{aligned} \tag{20}$$

To ensure the Lipschitz continuity condition, a gradient penalty factor $\mathcal{L}_{penal}$ is introduced to θ_{disc}:

$$\mathcal{L}_{penal}(\widehat{\boldsymbol{H}}) = (\|\nabla_{\widehat{\boldsymbol{H}}} f_{disc}(\widehat{\boldsymbol{H}}; \boldsymbol{\theta}_{disc})\|_2 - 1)^2 \tag{21}$$

where the representation $\widehat{\boldsymbol{H}}$ refers to a random point along the line between the representations of the source and target domains, or the source or target domain itself.

Therefore, the subgraph representation is maintained constant by minimizing and maximizing the following:

$$\min_{\boldsymbol{\theta}_{encod}} \max_{\boldsymbol{\theta}_{disc}} \{\mathcal{L}_{disc} - \gamma \mathcal{L}_{penal}\} \tag{22}$$

where γ is the gradient penalty coefficient, set to zero during the training of the generator. Generally, the parameters in the domain discriminator $f_{disc}(\cdot)$ are trained to optimality before updating the generator $f_{encod}(\cdot)$ parameters to minimize the optimal transport distance.

Model training

Expanding formula (22) and integrating it into the semi-supervised learning prediction loss function yields the final loss function:

$$\min_{\boldsymbol{\theta}_{encod},\boldsymbol{\theta}_{pred}} \{\mathcal{L}_{pred} + \lambda \max_{\boldsymbol{\theta}_{disc}} [\mathcal{L}_{disc} - \gamma \mathcal{L}_{penal}]\} \tag{23}$$

where λ is the balancing coefficient between semi-supervised learning and domain adaptation.

The training process is depicted in Algorithm 1.

Algorithm 1 Training algorithm for ADA-Spear

Input: $G^s = (V^s, E^s, A^s, F_v^s, F_e^s), Y_s), Y^s$: source domain data; $G^t = (V^t, E^t, A^t, F_v^t, F_e^t))$: target domain data; s_{disc}: domain discriminator training step; γ, λ: coefficients in the loss function; $\alpha_{disc}, \alpha_{encod}$: learning rates of discriminator and encoder; N^s, N^t: number of subgraphs in the source and target domain; ϵ: a real number randomly drawing from the interval [0,1].

Output: The optimal encoder f_{encod}, semi-supervised learning classifier f_{pred} and domain discriminator f_{disc}.

Initialize parameters θ_{encod} for encoder f_{encod};
Initialize parameters θ_{pred} for classifier f_{pred};
Initialize parameters θ_{disc} for discriminator f_{disc};
while *not converge* **do**
  // training discriminator;
  for $t = 1, \ldots, s_{disc}$ **do**
    $H^s \leftarrow f_{encod}(A^s, F_v^s, F_e^s; \theta_{encod})$;
    $H^t \leftarrow f_{encod}(A^t, F_v^t, F_e^t, \theta_{encod})$;
    $N \leftarrow min\{N^s, N^t\}$;
    $h^s \leftarrow H^s$,$h^t \leftarrow H^t$, $\epsilon \leftarrow [0, 1]$;
    $h_i \leftarrow \epsilon h^s + (1 - \epsilon)h^t$;
    $H = \{h_i\}_{i=1}^N$;
    $\hat{H} \leftarrow \{H^s, H^t, H\}$;
    $\theta_{disc} \leftarrow \theta_{disc} + \alpha_{disc} \cdot \nabla_{\theta_{disc}} \{\mathcal{L}_{disc} - \lambda \mathcal{L}_{penal}(\hat{H})\}$;
  end for
  // training encoder and classifier;
  $\theta \leftarrow \{\theta_{encod}, \theta_{pred}\}$;
  $\theta \leftarrow \theta - \alpha_{encod} \cdot \nabla_{\theta} \{\mathcal{L}_{pred} + \lambda \mathcal{L}_{disc}\}$;
end while
return f_{encod}, f_{pred}, f_{disc};

Experiments

In this section, the performance of the proposed method is assessed in comparison to current advanced methods through experiments. The analysis includes evaluating the efficiency and scalability of the method. Following that, it investigates diverse cross-blockchain-network phishing behavior patterns. Additionally, it assesses the effectiveness of different modules in the adversarial domain adaptation architecture and explores the impact of inter-chain distribution differences on the model.

Data preparation

In this subsection, the most popular two blockchain platforms and the largest Initial Coin Offering (ICO) platform are selected as the subjects of the experiment: Ethereum (Etherscan)(ETH), Bitcoin (Nakamoto 2008) (BTC), and EOSIO (EOSIO)(EOS). Transaction data from the Ethereum blockchain for the years 2019–2021, Bitcoin for 2019–2021, and EOSIO for 2018–2019 are extracted. From each chain, 1000 phishing nodes are randomly sampled, and to balance the training samples, the same number of benign nodes are also obtained. Subgraphs are constructed for each using the method described in Sect. 4.1. Detailed statistics of the dataset are shown in Table 2.

For the three datasets, the reduced second-order subgraphs of phishing and benign nodes are used as the experimental input. The Ethereum dataset contains a total of 1,765,559 nodes and 4,319,271 edges, with the phishing nodes accounting for 0.06% of the total; the Bitcoin dataset includes 983,176 nodes and 4,859,188 edges, with phishing nodes making up 0.10%; the EOSIO dataset consists of 746,688 nodes and 9,727,013 edges, with phishing nodes comprising 0.13%. Due to EOSIO's characteristic of rapidly recording interaction information, it presents the highest total number of edges and average degree despite having the fewest total nodes. However, the proportion of phishing nodes in EOSIO is comparable to the other datasets, which does not affect the subsequent experimental analysis.

To further assess the performance of ADA-Spear across these three domains, it is divided into six transfer learning tasks, which include: ETH → BTC, EOS → BTC, ETH → EOS, BTC → EOS, BTC → ETH, and EOS → ETH, where ETH, BTC, and EOS respectively represent the Ethereum dataset, the Bitcoin dataset, and the EOSIO dataset.

Experiment setup

This subsection primarily describes the baseline methods and implementation details.

Table 2 Dataset statistics

Dataset	Total nodes	Phishing nodes	Proportion (%)	Total edges	Average degree
ETH	1,765,559	1000	0.06	4,319,271	4.87
BTC	983,176	1000	0.10	4,859,188	9.88
EOS	746,688	1000	0.13	9,727,013	26.05

Baselines

We select methods from different research approaches for comparative experiments, mainly divided into feature engineering, single-network learning, graph-based semi-supervised learning, and cross-network learning. The specifics are as follows:

Feature Engineering Inspired by Chen et al. (2020a), we select 219-dimensional features to characterize subgraph behavior for phishing behavior detection.

Single Network Learning We use DeepWalk (Perozzi et al. 2014), Graph2vec (Narayanan et al. 2017), Trans-2vec (Wu et al. 2020), and T-Edge (Lin et al. 2020) as comparative methods for extracting subgraph representations. DeepWalk and Graph2vec can extract the structural information of subgraphs, while Trans2vec and T-Edge incorporate additional information such as transaction amounts and timing on top of structural information.

After obtaining subgraph (feature) representations using feature engineering and single network learning methods, logistic regression, random forests, and support vector machines are used for the subgraph classification task.

Graph-based Semi-supervised Learning We employ GCN (Kipf and Welling 2016), GraphSage (Hamilton et al. 2017), and MCGC (Zhang et al. 2021) as three deep learning methods, detecting phishing behavior in an end-to-end learning manner. GCN can integrate network structure and attribute information. GraphSAGE is derived from a variant of the GCN aggregation function. MCGC is a deep learning method that extracts subgraph information hierarchically. The attributes required for this class of methods are consistent with those proposed in this subsection.

Cross-Network Learning We use NetTr (Fang et al. 2013) and CDNE (Shen et al. 2020) as two transfer learning methods. NetTr transfers only network structural information. CDNE introduces a MMD loss function to perform domain adaptation learning in an autoencoder fashion. The attributes required for this class of methods are consistent with those proposed in this subsection.

Implementation details

This subsection conducts experiments on a Linux operating system with 128 GB of memory. NetworkX (2020) is used for graph data processing, PyTorch (Paszke et al. 2019) for model construction, and Scikit-learn (Pedregosa et al. 2011) for handling evaluation metrics.

To better compare the effectiveness of the methods, the dimensions of the subgraph representations used in this subsection are all the same, each with 128 dimensions. The encoder of the method proposed in the paper consists of 1 layer of LSTM, 2 layers of GAT, 2 layers of DiffPool, and 2 layers of MLP, with each layer using a 128-dimensional output for hidden layers. The dropout rate for hidden neurons is set to 0.4. The function $f_{pred}(\cdot)$ is a multilayer perceptron, using a 128-dimensional output layer for label prediction. The domain discriminator $f_{disc}(\cdot)$ has 2 layers with 128 neurons each, and the balance coefficient λ, the gradient penalty coefficient γ, and the number of training steps for the domain discriminator s_{disc} are set to 0.8, 10, and 15, respectively. The learning rates for the encoder and domain discriminator, α_{encod} and α_{disc}, are both set to 0.001. The model is trained for 30 epochs.

For all non-cross-network learning methods, we merge the source and target networks into one network for experimentation. In the merged network, there are no edges between the source and target networks. Therefore, an 8:2 split is used as the training and test sets within each merged network, with fivefold cross validation used for experimentation. For single network learning methods, the walking length for DeepWalk, Trans2vec, and T-Edge is set to 10, and the context size is set to 4. The shift parameter α in Trans2vec and T-Edge is set to 0.5. In Graph2vec, the number of training epochs is set to 30, with a learning rate of 0.001. For subsequent classifiers, the maximum number of iterations is set to 100 in logistic regression. The support vector machine uses an RBF kernel with hyperparameter γ set to 1000. In the random forest, the maximum depth is set to 7, with the number of base decision trees set to 100.

For graph-based semi-supervised learning methods, the GCN layers in GCN, GraphSage, and MCGC are set to 128 dimensions. The number of training epochs is set to 30, with a learning rate of 0.001. For MCGC, the number of aggregation layers is set to 3 layers.

For cross-network learning methods, NetTr and CDNE adopt the hyperparameters recommended in the literature.

Experimental results

This section primarily investigates the comparative effectiveness of the proposed method against current advanced methods. To ensure that the behavior patterns of the three blockchains are sufficiently similar to warrant transfer learning, the target domains in the experiments are divided into fully unlabeled and partially labeled (with a label rate of 5%). In this section, ADA-Spear's detection effectiveness is verified to surpass current advanced methods across these three datasets and six transfer learning tasks. The main results are presented in terms of F1 scores, as shown in Table 3.

Firstly, from the table for the target domain with no labels, it can be seen that the proposed method ADA-Spear outperforms all comparison methods, demonstrating its effectiveness in cross-blockchain-network phishing behavior detection. Under the six cross-blockchain-network detection tasks, ADA-Spear's F1 score is, on average, 7.1% higher than the best comparison method.

The F1 scores of feature engineering and single network learning methods are comparatively low, which aligns with expectations. Feature engineering methods can only depict certain types of phishing behaviors on a specific chain and struggle to comprehensively characterize phishing behaviors across different chains.

Within single network learning methods, it's evident that DeepWalk and Graph2vec, which only consider network structure, perform very poorly, even comparable to feature engineering methods. This can be attributed to the lack of interconnectivity between ETH, BTC, and EOS networks, resulting in incomparable representation vectors trained from the source and target domains. This is the reason for the subpar detection effectiveness. Moreover, because they do not incorporate any semantic characterization of on-chain phishing behavior, they cannot compare with Trans2vec and T-Edge, which introduce semantics. Semantic-aware detection methods outperform those considering only network structure by an average of 7.25% in F1 score, suggesting that phishing behaviors across heterogeneous chains share certain semantic similarities.

Graph-based semi-supervised learning methods show significant improvement over the previous two categories, with the MCGC method achieving the best results among them. These methods are, on average, 10.04% higher in F1 score compared to single network learning methods. This improvement can be attributed to the end-to-end learning mode of semi-supervised methods, which allows for the adjustment of subgraph behavior representation while classifying, rather than post-representation classification as with the previous two unsupervised learning approaches. MCGC achieves the best results because it overcomes the flattening issue of GCN-based methods, considering subgraph representation from a hierarchical perspective, and portraying behaviors more completely and accurately at the subgraph level.

The cross-network learning methods NetTr and CDNE are the most effective among all comparison methods but still have a significant gap compared to the detection effectiveness of the proposed method in this paper. ADA-Spear's F1 score exceeds the average of cross-network learning methods by 7.1%. Especially NetTr did not achieve the expected detection performance. This may be due to NetTr only transferring the topological structure between the source and target domains, not considering the semantic information of behaviors, which also indirectly proves that phishing behaviors on the three different chains have significant topological drift. The substantial improvement in the detection effectiveness of ADA-Spear is mainly due to the integration of more semantic information on phishing behavior, hierarchical stereoscopic behavior characterization, and the loss function in adversarial domain adaptation being more effective than the MMD loss function.

From these six transfer learning tasks, it is observed that transfers to EOS are more challenging, which indirectly suggests that the behavior patterns of nodes on EOS differ significantly from those on other chains. However, the detection results are still at a high level, which indicates that although there are distribution differences in behavior, the method proposed in this paper can still mitigate distribution drift and achieve good detection results.

Comparing the tables for target domains with and without labels shows that introducing target domain labels improves the detection performance of all models, with many phenomena similar to those observed in the absence of labels. While detection performance improves with the introduction of labels, the increase is not substantial, which also demonstrates the robustness of the model. Furthermore, it indicates that phishing behaviors across these three chains possess certain similarities and that the distributions have consistency. This validates the meaningfulness of using domain adaptation methods.

Efficiency analysis

This section conducts an efficiency analysis. Firstly, an analysis of the relationship between time and cost across different sampling ranges was carried out, with the results shown in Fig. 8. This subsection uses the ratio of the number of nodes and edges to the total number of nodes and edges as a cost indicator for analysis. Since the task duration remains the same when the datasets used in the source and target domains of transfer learning are swapped, so nodes and edges increasing are showing

Table 3 Comparison of F1 scores of ADA-Spear and other methods' detection results, with optimal results within all methods indicated by bolded style and optimal results within each type indicated by wavy lines

	Target domain unlabeled (%)						Target domain 5% labeled (%)					
	ETH → BTC	**EOS → BTC**	**ETH → EOS**	**BTC → EOS**	**EOS → ETH**	**BTC → ETH**	**ETH → BTC**	**EOS → BTC**	**ETH → EOS**	**BTC → EOS**	**EOS → ETH**	**BTC → ETH**
Feature based												
Static Feature + LR (Chen et al. 2020a)	40.57 ± 0.029	39.73 ± 0.031	34.60 ± 0.037	32.55 ± 0.020	41.01 ± 0.034	41.86 ± 0.028	48.64 ± 0.029	48.19 ± 0.023	42.78 ± 0.023	40.74 ± 0.021	49.09 ± 0.020	50.91 ± 0.024
Static feature +SVM (Chen et al. 2020a)	41.34 ± 0.040	40.52 ± 0.024	35.23 ± 0.019	33.19 ± 0.023	42.28 ± 0.020	43.46 ± 0.020	49.39 ± 0.018	48.99 ± 0.018	43.38 ± 0.023	41.35 ± 0.026	50.30 ± 0.024	51.51 ± 0.028
Static feature + RF (Chen et al. 2020a)	*42.01 ± 0.028*	*41.20 ± 0.016*	*37.97 ± 0.030*	*35.95 ± 0.028*	*43.03 ± 0.036*	*43.84 ± 0.016*	*50.05 ± 0.014*	*49.65 ± 0.028*	*46.09 ± 0.024*	*44.06 ± 0.025*	*51.11 ± 0.020*	*51.92 ± 0.021*
Single- network based												
DeepWalk + LR (Perozzi et al. 2014)	40.65 ± 0.027	39.84 ± 0.027	37.83 ± 0.031	40.12 ± 0.024	42.04 ± 0.020	43.27 ± 0.016	48.78 ± 0.018	48.37 ± 0.021	45.88 ± 0.027	43.86 ± 0.016	50.21 ± 0.025	52.40 ± 0.021
DeepWalk + SVM (Perozzi et al. 2014)	42.20 ± 0.023	43.72 ± 0.019	39.81 ± 0.026	37.80 ± 0.030	43.20 ± 0.020	46.26 ± 0.015	50.20 ± 0.017	49.80 ± 0.022	47.80 ± 0.015	48.18 ± 0.021	51.20 ± 0.017	51.43 ± 0.028
DeepWalk + RF (Perozzi et al. 2014)	44.56 ± 0.032	47.89 ± 0.031	45.46 ± 0.024	37.80 ± 0.030	45.66 ± 0.019	50.25 ± 0.017	52.62 ± 0.021	52.22 ± 0.016	50.20 ± 0.028	45.80 ± 0.027	53.74 ± 0.025	54.95 ± 0.024
Graph2vec + LR (Narayanan et al. 2017)	46.43 ± 0.038	45.83 ± 0.024	43.99 ± 0.016	41.96 ± 0.037	48.01 ± 0.033	44.40 ± 0.027	54.58 ± 0.026	54.17 ± 0.026	53.44 ± 0.026	50.11 ± 0.033	57.78 ± 0.028	57.55 ± 0.019
Graph2vec + SVM (Narayanan et al. 2017)	48.49 ± 0.034	41.38 ± 0.023	*47.20 ± 0.021*	43.47 ± 0.025	49.70 ± 0.025	52.40 ± 0.027	56.56 ± 0.028	*58.05 ± 0.012*	52.14 ± 0.028	51.44 ± 0.027	56.33 ± 0.018	58.99 ± 0.016
Graph2vec + RF (Narayanan et al. 2017)	*50.50 ± 0.026*	*50.10 ± 0.020*	42.13 ± 0.028	*53.75 ± 0.019*	*51.40 ± 0.024*	*53.42 ± 0.022*	*58.45 ± 0.014*	56.16 ± 0.027	*55.20 ± 0.020*	*57.81 ± 0.021*	*59.36 ± 0.021*	*60.40 ± 0.019*
Trans2vec + LR (Wu et al. 2020)	55.58 ± 0.024	55.32 ± 0.018	53.54 ± 0.021	51.37 ± 0.026	55.72 ± 0.013	59.47 ± 0.018	59.78 ± 0.014	59.37 ± 0.014	57.58 ± 0.012	55.41 ± 0.024	61.12 ± 0.016	62.76 ± 0.017
Trans2vec + SVM (Wu et al. 2020)	*59.48 ± 0.012*	56.47 ± 0.020	54.86 ± 0.018	45.21 ± 0.028	56.99 ± 0.029	58.70 ± 0.020	62.22 ± 0.015	60.82 ± 0.023	*61.34 ± 0.013*	56.91 ± 0.018	62.45 ± 0.013	63.54 ± 0.015
Trans2vec + RF (Wu et al. 2020)	57.09 ± 0.020	57.78 ± 0.017	55.78 ± 0.024	52.82 ± 0.025	*58.82 ± 0.008*	60.74 ± 0.016	61.23 ± 0.018	61.82 ± 0.015	59.84 ± 0.018	53.20 ± 0.013	*65.05 ± 0.012*	64.92 ± 0.015
TEdge + LR (Lin et al. 2020)	55.06 ± 0.020	54.65 ± 0.022	52.60 ± 0.019	50.60 ± 0.017	58.04 ± 0.019	60.50 ± 0.018	59.06 ± 0.018	62.26 ± 0.013	56.60 ± 0.012	54.60 ± 0.015	60.40 ± 0.017	61.40 ± 0.015
TEdge + SVM (Lin et al. 2020)	58.66 ± 0.020	*59.07 ± 0.012*	*57.29 ± 0.017*	54.16 ± 0.016	56.19 ± 0.017	57.40 ± 0.019	62.67 ± 0.021	58.66 ± 0.014	60.18 ± 0.016	58.17 ± 0.021	63.72 ± 0.014	*65.73 ± 0.014*
TEdge + RF (Lin et al. 2020)	58.13 ± 0.022	58.26 ± 0.017	56.17 ± 0.021	*55.26 ± 0.020*	58.32 ± 0.021	*61.89 ± 0.008*	*63.51 ± 0.010*	*63.11 ± 0.013*	59.84 ± 0.018	*59.32 ± 0.020*	63.44 ± 0.016	64.53 ± 0.017
Semi-supervised based												
GCN (Kipf and Welling 2016)	64.70 ± 0.012	63.75 ± 0.019	64.74 ± 0.017	60.16 ± 0.014	66.07 ± 0.016	66.93 ± 0.016	67.21 ± 0.015	66.81 ± 0.016	67.73 ± 0.011	63.21 ± 0.017	68.77 ± 0.014	70.46 ± 0.010
GraphSAGE (Hamilton et al. 2017)	66.53 ± 0.016	66.20 ± 0.014	62.20 ± 0.017	62.41 ± 0.011	63.34 ± 0.015	66.40 ± 0.015	69.60 ± 0.009	*72.97 ± 0.012*	65.24 ± 0.017	*67.54 ± 0.011*	70.40 ± 0.015	*73.77 ± 0.012*

Table 3 (continued)

	Target domain unlabeled (%)						Target domain 5% labeled (%)					
	ETH → BTC	**EOS → BTC**	**ETH → EOS**	**BTC → EOS**	**EOS → ETH**	**BTC → ETH**	**ETH → BTC**	**EOS → BTC**	**ETH → EOS**	**BTC → EOS**	**EOS → ETH**	**BTC → ETH**
MCGC (Zhang et al. 2021)	*68.87 ± 0.018*	*69.71 ± 0.014*	*66.20 ± 0.013*	*64.52 ± 0.017*	*69.28 ± 0.006*	*71.07 ± 0.011*	*71.74 ± 0.013*	71.34 ± 0.011	*69.21 ± 0.011*	65.40 ± 0.015	*72.93 ± 0.013*	71.04 ± 0.015
Cross-network based												
NetTr (Fang et al. 2013)	57.45 ± 0.020	57.26 ± 0.014	56.87 ± 0.020	53.69 ± 0.013	59.51 ± 0.019	61.31 ± 0.023	60.78 ± 0.013	60.39 ± 0.008	58.62 ± 0.014	55.65 ± 0.017	62.83 ± 0.012	64.43 ± 0.007
CDNE (Shen et al. 2020)	70.79 ± 0.013	68.34 ± 0.018	65.99 ± 0.018	65.25 ± 0.017	72.21 ± 0.006	73.74 ± 0.013	73.38 ± 0.011	69.20 ± 0.012	67.82 ± 0.016	67.07 ± 0.013	75.53 ± 0.013	76.55 ± 0.008
ADA-Spear	**77.53 ± 0.011**	**75.76 ± 0.012**	**74.16 ± 0.012**	**73.23 ± 0.013**	**78.98 ± 0.010**	**80.86 ± 0.010**	**78.58 ± 0.008**	**77.73 ± 0.010**	**75.10 ± 0.011**	**74.04 ± 0.010**	**80.61 ± 0.009**	**82.45 ± 0.005**

the same trend in the tasks such as ETH-BTC: ETH → BTC and BTC → ETH but time cost between the tasks have slight fluctuation. It can be observed from the figure that the time increases with the number of sampled nodes, and the increase tends to be linear. Since graph data needs to be read, the ratio of nodes and edges represents the memory usage rate, which, as shown in the figure, increases linearly with the number of sampled nodes. Therefore, the method proposed in this paper tends towards linearity in both time and cost aspects, indicating that ADA-Spear is suitable for large-scale networks and has good scalability.

Secondly, this section also compares the impact of the sampling range on the F1 score for ADA-Spear with two other methods, with results shown in Fig. 9. The other two methods selected, GCN and CDNE, were chosen because they achieved the optimal F1 scores among the comparison methods. As can be seen from the figure, the F1 scores for all six tasks are generally optimized when K = 25; therefore, K = 25 was chosen for experimentation. Additionally, it is evident from the figure that the F1 score for ADA-Spear consistently outperforms that of GCN and CDNE. The detection effect is not ideal before K reaches 20, but there is a significant improvement after K exceeds 20. However, there is a slight decline when K reaches 30. This indicates that the neighboring nodes have the maximum amount of information when K is between 20 and 25, and beyond 30, the redundant information becomes detrimental to the model's learning.

In summary, the method proposed in this section has the best efficiency in balancing time and detection effectiveness, surpassing the other methods.

Ablation experiment

This section investigates the impact of the adversarial domain adaptation module and the encoder module on the overall model's detection performance and conducts a visual analysis.

Impact of the adversarial domain adaptation module

To explore the effect of this module on the overall detection performance, the subsection conducts experiments without this module, and the comparative results are presented in Table 4. As can be seen from the table, the adversarial domain adaptation module mitigates domain discrepancies and enhances detection performance. The absence of this module would lead to a significant distributional shift, whereas ADA-Spear can make the subgraph representations have clearer class boundaries. This demonstrates that the adversarial domain adaptation module can effectively alleviate the distribution drift between chains, making a substantial contribution to the detection of phishing activities on new blockchains.

Impact of the encoder network

To explore whether the encoder network proposed in this paper accurately captures the distinctive characteristics of phishing and benign behaviors, this section

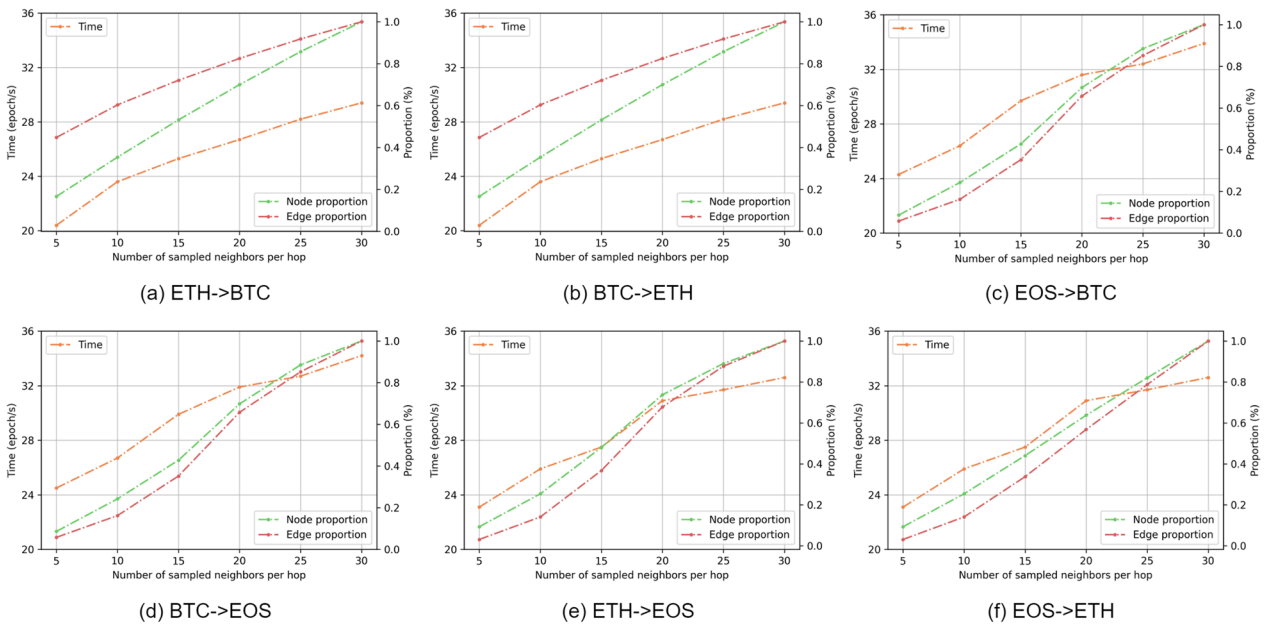

Fig. 8 Impact of sampling range on time and cost (training time, node-to-edge ratio): **a** ETH → BTC task; **b** BTC → ETH task; **c** EOS → BTC task; **d** BTC → EOS task; **e** ETH → EOS task; **f** EOS → ETH task

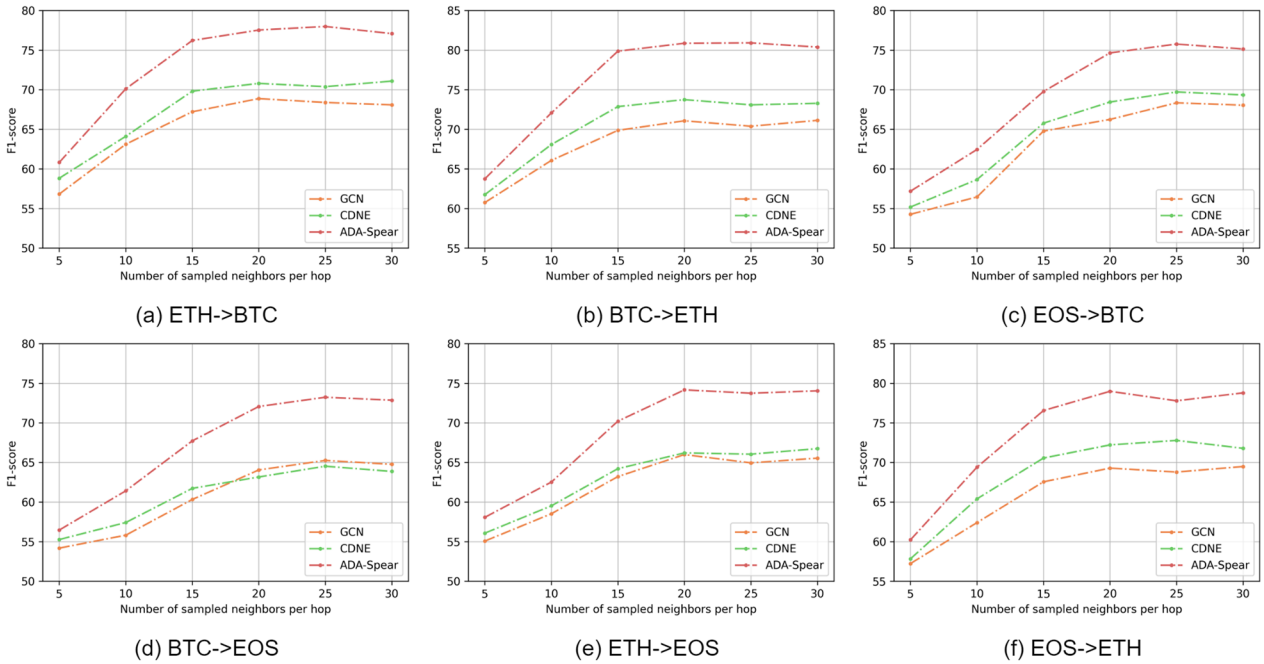

Fig. 9 Impact of sampling range on detection results: **a** ETH → BTC task; **b** BTC → ETH task; **c** EOS → BTC task; **d** BTC → EOS task; **e** ETH → EOS task; **f** EOS → ETH task

replaces the proposed encoder network with a multilayer perceptron. This perceptron utilizes a fully connected approach for feature extraction. The comparative results are shown in Table 5. As can be seen from the table, the implementation of the encoder designed in this paper leads to improvements in all six cross-blockchain-network phishing detection tasks. The averages of Rc, Pr, and F1 increased by 2.85%,

Table 4 Impact of the adversarial domain adaptation module on detection performance

Task	Method	Rc (recall) (%)	Pr (precision) (%)	F1 score (%)
ETH → BTC	ADA-Spear/ADA	73.54± 0.042	69.49± 0.015	71.46± 0.012
	ADA-Spear	79.79 ± 0.051	75.39 ± 0.024	77.53 ± 0.011
	Improvement	6.25	5.90	6.07
EOS → BTC	ADA-Spear/ADA	72.71 ± 0.029	68.56 ± 0.010	70.58 ± 0.008
	ADA-Spear	78.12 ± 0.051	73.53 ± 0.023	75.76 ± 0.012
	Improvement	5.41	4.97	5.18
ETH → EOS	ADA-Spear/ADA	71.66 ± 0.036	67.98 ± 0.012	69.78 ± 0.011
	ADA-Spear	76.25 ± 0.047	72.19 ± 0.020	74.16 ± 0.012
	Improvement	4.59	4.21	4.38
BTC → EOS	ADA-Spear/ADA	70.62 ± 0.042	67.80 ± 0.014	69.18 ± 0.013
	ADA-Spear	75.22 ± 0.050	71.34 ± 0.020	73.23 ± 0.013
	Improvement	4.60	3.54	4.05
EOS → ETH	ADA-Spear/ADA	73.75 ± 0.041	70.38 ± 0.016	72.03 ± 0.011
	ADA-Spear	81.04 ± 0.051	77.03 ± 0.026	78.98 ± 0.010
	Improvement	7.29	6.65	6.95
BTC → ETH	ADA-Spear/ADA	74.59 ± 0.058	70.89 ± 0.023	72.69 ± 0.016
	ADA-Spear	82.29 ± 0.055	79.48 ± 0.031	80.86 ± 0.010
	Improvement	7.70	8.59	8.17
Average	Improvement	5.97	5.64	5.80

Table 5 Impact of the encoder network on detection performance

Task	Method	Rc (recall) (%)	Pr (precision) (%)	F1 score (%)
ETH → BTC	ADA-Spear/encoder	76.67 ± 0.059	72.44 ± 0.025	74.49 ± 0.015
	ADA-Spear	79.79 ± 0.051	75.39 ± 0.024	77.53 ± 0.011
	Improvement	3.12	2.95	3.04
EOS → BTC	ADA-Spear/encoder	76.67 ± 0.059	72.44 ± 0.025	74.49 ± 0.015
	ADA-Spear	78.12 ± 0.051	73.53 ± 0.023	75.76 ± 0.012
	Improvement	1.45	1.09	1.27
ETH → EOS	ADA-Spear/encoder	72.71 ± 0.041	68.83 ± 0.015	70.72 ± 0.012
	ADA-Spear	76.25 ± 0.047	72.19 ± 0.020	74.16 ± 0.012
	Improvement	3.54	3.36	3.44
BTC → EOS	ADA-Spear/encoder	72.08 ± 0.051	68.38 ± 0.018	70.18 ± 0.015
	ADA-Spear	75.22 ± 0.050	71.34 ± 0.020	73.23 ± 0.013
	Improvement	3.14	2.96	3.05
EOS → ETH	ADA-Spear/encoder	78.32 ± 0.041	74.17 ± 0.019	76.19 ± 0.009
	ADA-Spear	81.04 ± 0.051	77.03 ± 0.026	78.98 ± 0.010
	Improvement	2.72	2.86	2.79
BTC → ETH	ADA-Spear/encoder	79.17 ± 0.047	75.85 ± 0.023	77.47 ± 0.010
	ADA-Spear	82.29 ± 0.055	79.48 ± 0.031	80.86 ± 0.010
	Improvement	3.12	3.63	3.39
Average	Improvement	2.85	2.81	2.83

2.81%, and 2.83%, respectively. This demonstrates that the encoder can effectively delineate the distinctive features between phishing and benign behaviors, making it an indispensable part of the model.

Distribution difference analysis

This section discusses the differences in distributions between the source chain and the target chain. Under six cross-blockchain-network phishing detection tasks, the impact of changes in the distribution differences on the detection results is explored by varying the feature overlap degree between the source and target domains. The feature overlap degree is defined as $C = \frac{|F^s \cap F^t|}{|F^s \cup F^t|}$, where F^r (with $r \in \{s, t\}$) represents the features in domain r. This is achieved by randomly removing certain attributes, causing C to vary from 10% to 50%. The final results are shown in Fig. 10. The best methods of semi-supervised learning and cross-network learning are selected for comparison with the proposed method. As seen in the figure, ADA-Spear consistently outperforms GCN and CDNE across all label rates in all tasks. This also indicates that, even with only a small portion of overlapping attributes between two domains, adversarial domain adaptation techniques still make a significant contribution to the model. This adequately demonstrates the robustness of the method proposed in this paper, making it capable of detecting phishing behaviors across an even wider range of new chains.

Sensitivity analysis

This section conducts a parameter sensitivity analysis of ADA-Spear to explore the impact of hyperparameters on the model. The sensitivity of the LSTM length m, the training steps of the domain discriminator s_{disc}, the penalty factor γ, and the balance coefficient λ are analyzed. Since the trends in hyperparameter changes are similar across various tasks, this section presents the F1 values only for the ETH → BTC task to avoid repetition. The method of controlling variables is used here; when studying a single hyperparameter, the others are kept constant as described in Sect. 5.2.

LSTM Length m The length of LSTM is used to receive subgraph feature inputs of different total time steps. As shown in Fig. 11a, subgraphs with a time span of more than 30 days have achieved good and stable detection results. The detection results show stability when LSTM encompasses information over more days, demonstrating the robustness of the model. When the length is 10, the detection performance is lower, indicating that subgraphs should ideally not be chosen with less than 10 steps. The stable performance in the graph also proves that using LSTM of variable lengths for detection is feasible.

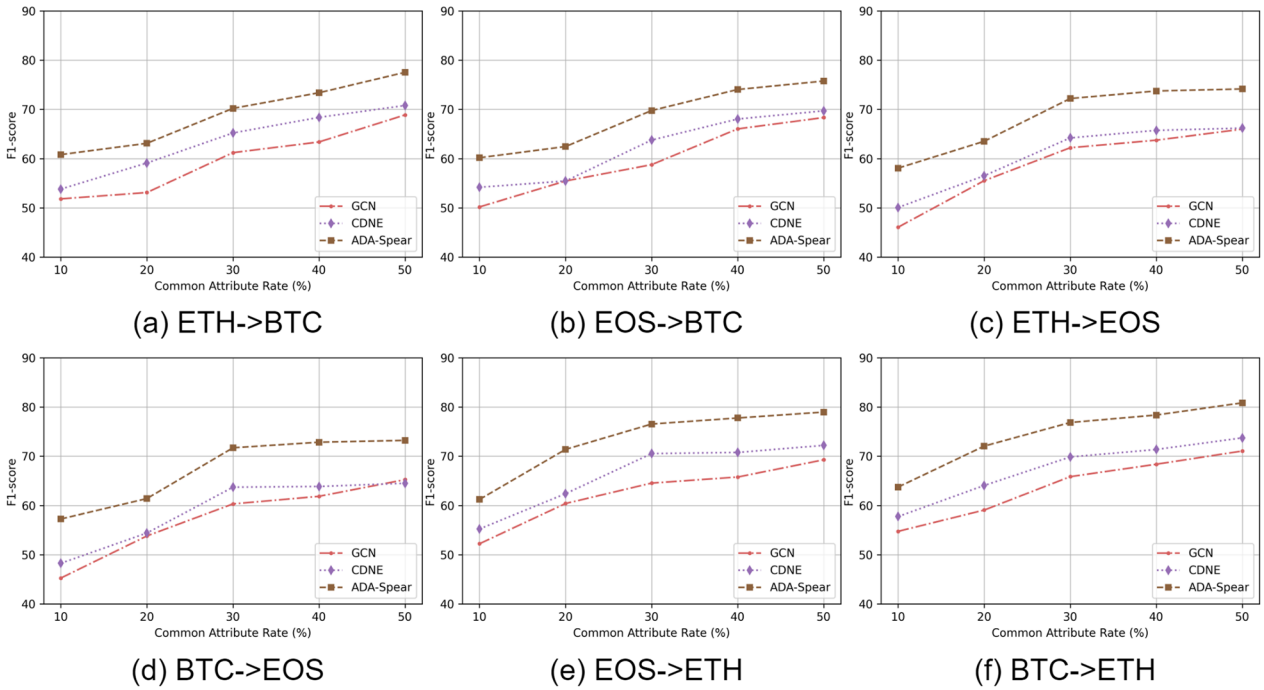

Fig. 10 Comparison chart of detection efficacy with different overlap rates of attributes between source and target domains: **a** ETH → BTC; **b** EOS → BTC; **c** ETH → EOS; **d** BTC → EOS; **e** EOS → ETH; **f** BTC → ETH

Domain Discriminator Training Steps s_{disc} As observed in Fig. 11b, there is a clear increase in the F1 value when s_{disc} changes from 5 to 10, followed by a tendency to stabilize. This confirms the optimization theory of the domain discriminator. Since the parameters of other components in the model remain unchanged during the training of the domain discriminator, as long as the number of training steps for the domain discriminator is sufficiently large, it can reach the optimal solution.

Penalty Factor γ The penalty factor is used to adjust various hyperparameters of the domain discriminator. As can be seen from Fig. 11c, the optimal detection performance is achieved when $\gamma = 10$. Both excessively high and low values lead to model degradation. Therefore, this paper adopts a penalty factor of 10 for the experiments.

Balance Coefficient λ The balance coefficient is used to adjust the balance between adversarial domain adaptation learning and semi-supervised learning. As shown in Fig. 11d, the detection results improve when λ varies between 0.4 and 1.0. There is a sharp decline in detection performance after $\lambda = 1.0$. Therefore, the chosen λ for the model should be between 0.4 and 1.0 to balance the learning of distinguishability between different categories and the similarity between domains.

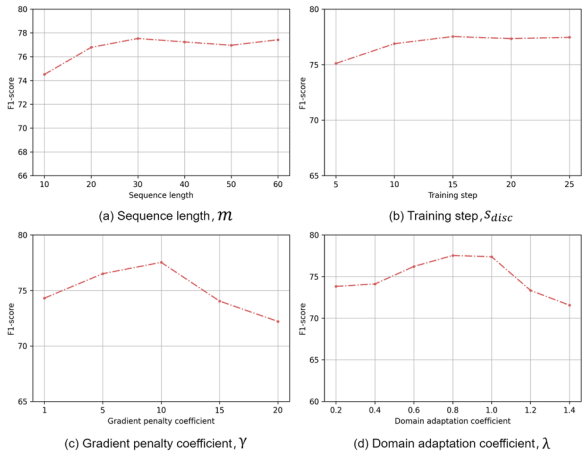

Fig. 11 ADA-Spear parameter sensitivity analysis: **a** LSTM sequence length; **b** training steps; **c** penalty factor; **d** balance coefficient

Conclusion

This paper addresses the detection of cross-blockchain-network phishing activities by combining an encoder that effectively characterizes phishing behaviors with ADA-Spear's adversarial domain adaptation module.

The encoder module combines temporal and topological information to create a hierarchical subgraph representation. Simultaneously, the adversarial domain adaptation module facilitates knowledge transfer from the source domain to the target domain. This combination enables ADA-Spear to be effectively applied to new chains for recognizing phishing behavior. Detailed experiments, using Ethereum, Bitcoin, and EOSIO as examples, further demonstrate ADA-Spear's advantages in accuracy and robustness. Meanwhile, ADA-Spear exhibits certain limitations.The precision of ADA-Spear demonstrates a considerable improvement over existing methods; however, there is still room for further enhancement until it can rival the detection performance achieved solely within a single chain. Additionally, ADA-Spear may encounter limitations in knowledge transfer in cases where there is significant disparity between the two chains. Research on generalizable phishing recognition methods across different chains is still in its early stages. This paper introduces one potential approach, offering a new research perspective to the community.

Acknowledgements
We are deeply grateful to the blind peer reviewers for their valuable suggestions and feedback on our paper. Their contributions have significantly enhanced the quality of our research. We would also like to extend our gratitude to all those who have provided assistance and support throughout the development of this paper.

Author contributions
All authors have contributed to this manuscript and approve of this submission. CY participated in all the work and drafting the article. XH participated in the drafting the article. YZ did some basic collection work. Prof. YL, ZL, and Dr. DD made a decisive contribution to the content of research and revising the article critically.

Funding
This research is supported by National Key Research and Development Program of China (Nos. 2023YFC3306305, 2021YFF0307203, 2019QY1300), Foundation Strengthening Program Technical Area Fund (No.2021-JCJQ-JJ-0908), technological project funding of the State Grid Corporation of China (Contract Number: SG270000YXJS2311060), Youth Innovation Promotion Association CAS (No. 2021156), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDC02040100) and National Natural Science Foundation of China (No. 61802404). This work is also supported by the Program of Key Laboratory of Network Assessment Technology, the Chinese Academy of Sciences, Program of Beijing Key Laboratory of Network Security and Protection Technology.

Availability of data and materials
The datasets using in this paper can be seen in https://drive.google.com/drive/folders/1xoplZg0vOBKF9Hw1aHNBD-KTpK0YW_P0?usp=sharing.

Declarations

Competing interests
The authors declare that they have no Conflict of interest.

Received: 15 January 2024 Accepted: 1 April 2024
Published online: 19 June 2024

References

Aggarwal CC et al (2015) Data mining: the textbook, vol 1
Ao X, Liu Y, Qin Z, Sun Y, He Q (2021) Temporal high-order proximity aware behavior analysis on Ethereum. World Wide Web 24:1–21
Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In: International conference on machine learning. PMLR, pp 214–223
Chainanalysis: The 2022 Crypto Crime Report. https://go.chainalysis.com/rs/503-FAP-074/images/Crypto-Crime-Report-2022.pdf
Chen L, Peng J, Liu Y, Li J, Xie F, Zheng Z (2020a) Phishing scams detection in Ethereum transaction network. ACM Trans Internet Technol (TOIT) 21(1):1–16
Chen T, Li Z, Zhu Y, Chen J, Luo X, Lui JC-S, Lin X, Zhang X (2020b) Understanding Ethereum via graph analysis. ACM Trans Internet Technol (TOIT) 20(2):1–32
CNVD-BC: blockchain security situation perception report. https://bc.cnvd.org.cn/notice_info?num=0c4088bbb6f7346000c3ac1ce13f0347
CoinMarketCap: CoinMarketCap. https://coinmarketcap.com/
Dai Q, Li Q, Tang J, Wang D (2018) Adversarial network embedding. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
Dai Q, Shen X, Zhang L, Li Q, Wang D (2019) Adversarial training methods for network embedding. In: The world wide web conference, pp 329–339
EOSIO: EOSIO. https://eos.io/
Etherscan: Etherscan. https://etherscan.io
Etherscan: explore navigate Etherescan's label world cloud. https://etherscan.io/labelcloud
Fang M, Yin J, Zhu X (2013) Transfer learning across networks for collective classification. In: 2013 IEEE 13th international conference on data mining. IEEE, pp 161–170
Ganin Y, Lempitsky V (2015) Unsupervised domain adaptation by backpropagation. In: International conference on machine learning. PMLR, pp 1180–1189
Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, March M, Lempitsky V (2016) Domain-adversarial training of neural networks. J Mach Learn Res 17(59):1–35
Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. Commun ACM 63(11):139–144
Gouk H, Frank E, Pfahringer B, Cree MJ (2021) Regularisation of neural networks by enforcing lipschitz continuity. Mach Learn 110:393–416
Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of wasserstein gans. Adv Neural Inf Process Syst 30
Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. Adv Neural Inf Process Syst 30
Heimann M, Koutra D (2017) On generalizing neural node embedding methods to multi-network problems. In: KDD MLG workshop
Jain AK, Gupta BB et al (2017) Phishing detection: analysis of visual similarity based approaches. Secur Commun Netw 2017
Ji S, Pan S, Cambria E, Marttinen P, Philip SY (2021) A survey on knowledge graphs: representation, acquisition, and applications. IEEE Trans Neural Netw Learn Syst 33(2):494–514
Jiao P, Guo X, Jing X, He D, Wu H, Pan S, Gong M, Wang W (2021) Temporal network embedding for link prediction via VAE joint attention mechanism. IEEE Trans Neural Netw Learn Syst 33(12):7400–7413
Jourdan M, Blandin S, Wynter L, Deshpande P (2018) Characterizing entities in the bitcoin blockchain. In: 2018 IEEE international conference on data mining workshops (ICDMW). IEEE, pp 55–62
Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
Li Y, Cai Y, Tian H, Xue G, Zheng Z (2020) Identifying illicit addresses in bitcoin network. In: Blockchain and trustworthy systems: second international conference, BlockSys 2020, Dali, China, August 6–7, 2020, Revised Selected Papers 2. Springer, pp 99–111
Li S, Gou G, Liu C, Hou C, Li Z, Xiong G (2022) Ttagn: Temporal transaction aggregation graph network for Ethereum phishing scams detection. In: Proceedings of the ACM web conference 2022, pp 661–669

Lin D, Wu J, Yuan Q, Zheng Z (2020) T-edge: temporal weighted multidigraph embedding for Ethereum transaction network analysis. Front Phys 8:204
Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. Decentralized Business Review
Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S (2017) graph2vec: learning distributed representations of graphs. arXiv:1707.05005
NetworkX D (2020) Networkx: network analysis in python
Ni J, Chang S, Liu X, Cheng W, Chen H, Xu D, Zhang X (2018) Co-regularized deep multi-network embedding. In: Proceedings of the 2018 world wide web conference, pp 469–478
Orunsolu AA, Sodiya AS, Akinwale A (2022) A predictive model for phishing detection. J King Saud Univ-Comput Inf Sci 34(2):232–247
Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
Pan S, Hu R, Long G, Jiang J, Yao L, Zhang C (2018) Adversarially regularized graph autoencoder for graph embedding. arXiv:1802.04407
Pan S, Hu R, Fung S-F, Long G, Jiang J, Zhang C (2019) Learning graph embedding with adversarial training methods. IEEE Trans Cybern 50(6):2475–2487
Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32:8024–8035
Patel V, Pan L, Rajasegarar S (2020) Graph deep learning based anomaly detection in Ethereum blockchain network. In: Network and system security: 14th international conference, NSS 2020, Melbourne, VIC, Australia, November 25–27, 2020, Proceedings 14. Springer, pp 132–148
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
Sayadi S, Rejeb SB, Choukair Z (2019) Anomaly detection model over blockchain electronic transactions. In: 2019 15th international wireless communications and mobile computing conference (IWCMC). IEEE, pp 895–900
Shen J, Qu Y, Zhang W, Yu Y (2018) Wasserstein distance guided representation learning for domain adaptation. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
Shen X, Dai Q, Mao S, Chung F-L, Choi K-S (2020) Network together: node classification via cross-network deep network embedding. IEEE Trans Neural Netw Learn Syst 32(5):1935–1948
Singh A (2019) Anomaly detection in the Ethereum network. A thesis for the degree of Master of Technology/Indian Institute of Technology Kanpur
SlowMist: 2022 blockchain security and AML analysis annual report. https://www.slowmist.com/report/2022-Blockchain-Security-and-AML-Analysis-Annual-Report(CN).pdf
Toyoda K, Ohtsuki T, Mathiopoulos PT (2018) Multi-class bitcoin-enabled service identification based on transaction history summarization. In: 2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData). IEEE, pp 1153–1160
Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1225–1234
Wang X, Cui P, Wang J, Pei J, Zhu W, Yang S (2017) Community preserving network embedding. In: Proceedings of the AAAI conference on artificial intelligence, vol 31
Wiles O, Gowal S, Stimberg F, Alvise-Rebuffi S, Ktena I, Dvijotham K, Cemgil T (2021) A fine-grained analysis on distribution shift. arXiv:2110.11328
Wu J, Yuan Q, Lin D, You W, Chen W, Chen C, Zheng Z (2020) Who are the phishers? Phishing scam detection on Ethereum via network embedding. IEEE Trans Syst Man Cybern Syst 52(2):1156–1166
Xu L, Wei X, Cao J, Yu PS (2017) Embedding of embedding (EOE) joint embedding for coupled heterogeneous networks. In: Proceedings of the tenth ACM international conference on web search and data mining, pp 741–749
Xu L, Wei X, Cao J, Yu PS (2018) On exploring semantic meanings of links for embedding social networks. In: Proceedings of the 2018 world wide web conference, pp 479–488
Yang Z, Cohen W, Salakhudinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: International conference on machine learning. PMLR, pp 40–48
Ying Z, You J, Morris C, Ren X, Hamilton W, Leskovec J (2018) Hierarchical graph representation learning with differentiable pooling. Adv Neural Inf Process Syst 31
Yuan Q, Huang B, Zhang J, Wu J, Zhang H, Zhang X (2020) Detecting phishing scams on Ethereum based on transaction records. In: 2020 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1–5
Zhang D, Chen J, Lu X (2021) Blockchain phishing scam detection via multichannel graph classification. In: Blockchain and trustworthy systems: third international conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3. Springer, pp 241–256
Zheng H, Ma M, Ma H, Chen J, Xiong H, Yang Z (2023) Tegdetector: a phishing detector that knows evolving transaction behaviors. IEEE Trans Comput Soc Syst
Zuraiq AA, Alkasassbeh M (2019) Phishing detection approaches. In: 2019 2nd international conference on new trends in computing sciences (ICTCS). IEEE, pp 1–6

Publisher's Note