Adapting Meta Knowledge with Heterogeneous Information Network for COVID-19 Themed Malicious Repository Detection

Yiyue Qian[1] , **Yiming Zhang**[1] , **Yanfang Ye**[1*] , **Chuxu Zhang**[2*]

[1]Department of Computer and Data Sciences, Case Western Reserve University, USA
[2]Department of Computer Science, Brandeis University, USA

[1]{yiyue.qian, yiming.zhang10, yanfang.ye}@case.edu, [2]chuxuzhang@brandeis.edu

Abstract

As cyberattacks caused by malware have proliferated during the pandemic, building an automatic system to detect COVID-19 themed malware in social coding platforms is in urgent need. The existing methods mainly rely on file content analysis while ignoring structured information among entities in social coding platforms. Additionally, they usually require sufficient data for model training, impairing their performances over cases with limited data which is common in reality. To address these challenges, we develop Meta-AHIN, a novel model for COVID-19 themed malicious repository detection in GitHub. In Meta-AHIN, we first construct an attributed heterogeneous information network (AHIN) to model the code content and social coding properties in GitHub; and then we exploit attention-based graph convolutional neural network (AGCN) to learn repository embeddings and present a meta-learning framework for model optimization. To utilize unlabeled information in AHIN and to consider task influence of different types of repositories, we further incorporate node attribute-based self-supervised module and task-aware attention weight into AGCN and meta-learning respectively. Extensive experiments on the collected data from GitHub demonstrate that Meta-AHIN outperforms state-of-the-art methods.

1 Introduction

Along with the global health crisis, the cyberattacks have increased by more than 250% and caused over $300 million loss in 2020 [Waggoner and Markowitz, 2021]. During the pandemic, using malware as a major weapon, cybercriminals utilized coronavirus disease (COVID-19) related information as a lure to infiltrate victims' systems to perform attacks such as stealing users' virtual assets and compromising network, which have caused significant loss of individuals, corporations, and governments. To perform the attacks, cybercriminals may post malicious code repositories with a COVID-19 theme in social coding platforms to disseminate malware. As GitHub has become the largest source code repository platform with 40 million users and over 280 million public repositories, it has been utilized by cybercriminals to disseminate malware to compromise security of the cyberspace [Fan *et al.*, 2019]. For example, a COVID-19 themed repository with embedded malicious code hosted in GitHub could be directly forked by other developers and easily disseminated through other social platforms like Reddit. Therefore, preventing the propagation of COVID-19 themed malware is in urgent need. In this paper, we solve this problem by developing an intelligent framework to detect COVID-19 themed malicious repositories in GitHub.

To detect malicious code repositories, existing methods [Semmle, 2019a; Semmle, 2019b; Rokon *et al.*, 2020] mainly rely on code content while ignoring the rich structured relations among entities in social coding platforms, impairing the effectiveness for judging their legitimacy. More specifically, GitHub brought in two vulnerability detection products CodeQL [Semmle, 2019a] and LGTM [Semmle, 2019b] in 2019, which can automatically analyze the variants of critical vulnerabilities based on code content of the repository. Additionally, although some effective models [Zhang *et al.*, 2020a; Fan *et al.*, 2019] for malicious code repository detection have been proposed, they require sufficient samples for model training. For example, Rokon *et al.* [Rokon *et al.*, 2020] utilized 97K labeled repositories in GitHub to train a supervised-learning model for malicious repository detection. Thus, existing models are not suitable for detecting novel types of malicious repositories with few data, such as malware with a COVID-19 theme and their variants.

In this paper, we aim to address the above challenges by developing a novel model called **Meta-AHIN** (Figure 1(a)), which integrates graph convolutional neural network and meta-learning for COVID-19 themed malicious repository detection with few data constraint in GitHub. First, we construct an Attribute Heterogeneous Information Network (AHIN) to model the relationships among four types of entities (i.e., repository, keyword, file, and user) in GitHub. Next, we build an attention-based graph convolution network (AGCN) to fuse both structural relations and semantic content information for learning node representations in AHIN. To exploit and capture unlabeled information in AHIN, we further design a self-supervised module and refine the node embeddings which are fed to downstream classifier for ma-

*Corresponding authors.

licious repository detection. Since novel types of malicious repositories with a COVID-19 theme may only have few data samples, we build a meta-learning framework to transfer knowledge from detection tasks of regular repositories and adapt them quickly for new detection tasks (e.g., COVID-19 themed malicious repository detection). In addition, considering different malicious repository detection tasks contribute differently to the meta-learner, we introduce task-aware attention weights to measure their importances. Self-supervised module and task weights are incorporated into AGCN and meta-learning respectively for better detection performance. To summarize, the major contributions of this work include:

- We solve the problem of COVID-19 themed malicious repository detection with few data constraint in GitHub, which is novel and important.
- To handle structural relations and unstructured content information in GitHub, and to account for the constraint of few labeled samples of COVID-19 themed malicious repository, we develop a novel model called Meta-AHIN by integrating graph convolutional network and meta-learning.
- We collect COVID-19 themed malicious repository data from GitHub and conduct extensive experiments using this dataset. Promising results demonstrate the effectiveness of our model by comparison with state-of-the-art methods.

2 Related Work

Malware Repository Detection. The recent studies for malicious repository detection usually focus on the analysis of code (or app) content [Ye *et al.*, 2017; Ragkhitwetsagul *et al.*, 2019; Meli *et al.*, 2019; Rokon *et al.*, 2020]. For example, Rokon *et al.* proposed a content-based supervised learning method to identify malicious repositories in GitHub [Rokon *et al.*, 2020]. These methods, however, ignore the rich structural information among different types of entities (e.g., repository, user) in social coding platforms. Different from existing works, we study the characteristics of code repository by incorporating both structural relation and unstructured content information in GitHub.

Graph Convolutional Network (GCN). Our graph representation learning framework is conceptually inspired by GCN [Wu *et al.*, 2020]. There are two main types of GCN: spectral methods and spatial methods. For spectral methods, they aim to represent nodes in graph and perform convolution in the spectral space [Bruna *et al.*, 2013; Henaff *et al.*, 2015; Defferrard *et al.*, 2016]. For spatial methods, they operate the original graph directly and define convolution layers over nodes to aggregate information of local neighbors [Hamilton *et al.*, 2017; Veličković *et al.*, 2018]. Besides homogeneous GCN, some models [Zhang *et al.*, 2019; Zhang *et al.*, 2020b; Ye *et al.*, 2020b; Ye *et al.*, 2020a] handle heterogeneous graphs. Inspired by these studies, we build an attention-based GCN to learn node embeddings in AHIN.

Meta-Learning. Recent meta-learning models generally fall into two groups: (1) metric-based meta-learning [Vinyals *et al.*, 2016; Snell *et al.*, 2017], and (2) gradient-based meta-learning [Finn *et al.*, 2017; Lee and Choi, 2018; Finn *et al.*, 2018]. For the former ones, they implement a generalized metric and matching functions from training tasks to train the model; for the latter ones, they propose to employ data of existing tasks to learn well initialized model parameters that can be updated to new tasks in a fast manner with few data. In this paper, we are motivated to use gradient based meta-learning to address the challenge of few labeled data for COVID-19 themed malicious repository detection.

3 Preliminaries

3.1 Problem Definition

Let $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ denote an attributed heterogeneous information network (AHIN), where $\mathcal{V}$ is the set of different types of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of different edges, and $\mathcal{X}$ is the attributed feature set. A node can be regarded as any type of entities and an edge can be regarded as any type of relations between two nodes. There are four types of nodes (repository, keyword, file, and user) and seven types of relations (e.g., file-contain-keyword) in our built AHIN (see Figure 1(b)). The goal is to learn repository embeddings $\mathbf{f}$ which can be fed to a classifier for malicious repository detection. Given the features of all repository nodes $X_r = (x_1, \ldots, x_N)$ (N: number of repositories) and their labels $Y = (y_1, \ldots, y_N)$ ($y_i = 1$ denotes malicious and $y_i = 0$ represents benign), we aim to learn a mapping function $U_\phi : X_r \rightarrow Y$ (with parameter ϕ). Unlike existing works that use sufficient samples for model training, we consider a more practical scenario that only few labeled data are available since novel COVID-19 themed repositories only have limited malicious samples. Specifically, given different types of regular repositories and their labels, we aim to build a classifier which can be quickly adapted to predict labels of different types of COVID-19 themed repositories that are unseen during the model training, given few malicious samples of these repositories. Formally, the problem is defined as follows.

Problem 1. ***COVID-19 themed malicious repository detection.*** *Given a set of different types of code repositories (w/o a COVID-19 theme) along with their attributed features $X_r = (x_1, \ldots, x_N)$ and corresponding labels $Y = (y_1, \ldots, y_N)$ as well as the built AHIN G (training data), the problem is to build a machine learning model to detect malicious code repositories with a COVID-19 theme that have few malicious samples (testing data).*

3.2 Graph Convolutional Network

Since GCN has been proved to be powerful for learning the node representations [Kipf and Welling, 2017] by exploring both structure of a graph G and node features X, we employ GCN as the base model to learn the repository representations in AHIN. Our goal is to learn the embedding $\mathbf{f}_i \in \mathbb{R}^d$ (d: embedding dimension), which corresponds to the representation of repository node i. The layer-wise propagation rule of GCN is defined as follows:

$$H^{l+1} = \sigma(\widetilde{A}\, H^l\, W^l), \tag{1}$$

where H^{l+1} denotes the node representations at $l+1$ layer and $H^0 = X$, $\widetilde{A}$ is a symmetric normalization of A with self-loop, i.e., $\widetilde{A} = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$ with $\hat{A} = A + I_N$. A, I_N, $\hat{D}$ are the adjacency matrix, the identity matrix, and the diagonal

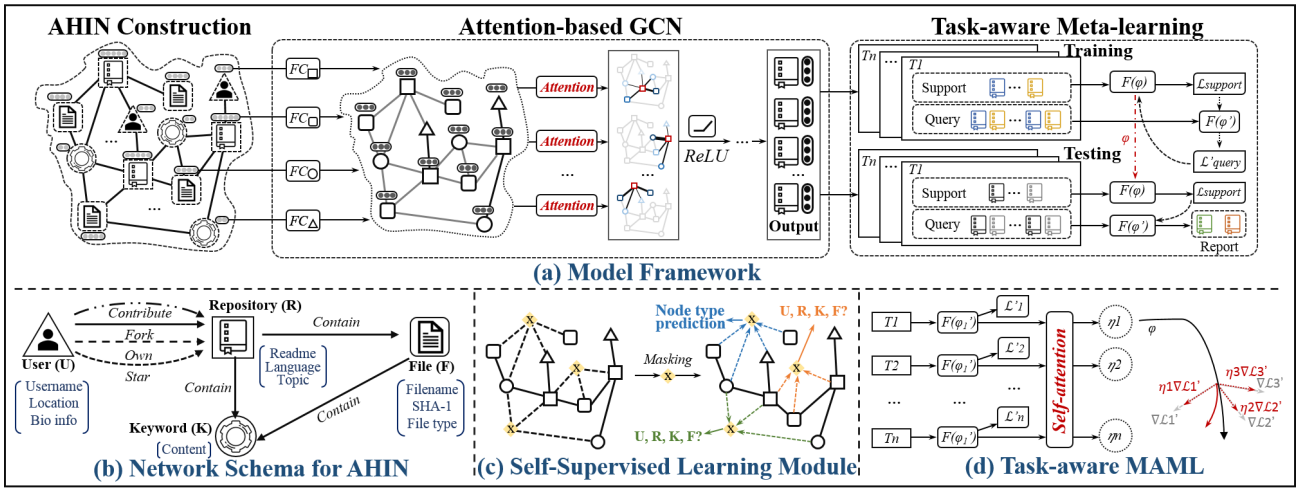

Figure 1: The overall framework of Meta-AHIN: (a) It first constructs an AHIN, and then develops an attention-based GCN to obtain repository embedding. Task attention-based meta-learning is designed to optimize model parameters. (b) AHIN contains four types of entities and seven types of relations as well as content features of all nodes. (c) Self-supervised module by masking node types. (d) Task-aware MAML by incorporating task attention weight into MAML.

node degree matrix of $\hat{A}$ respectively. W^l denotes the weight matrix at l-th layer, and σ is the activation function. For simplicity, we use $\mathbf{f} = \text{GCN}(X, A)$ to denote a GCN model.

4 Proposed Model

In this section, we present the details of Meta-AHIN. At first, we construct an AHIN to depict both structural relations and unstructured content information in GitHub. Then we propose an attention-based GCN which is further augmented by a self-supervised module, to learn the node embeddings for malicious repository detection. Lastly, we introduce a task attentive meta-learning procedure for model optimization.

4.1 AHIN Construction

To comprehensively describe repositories in GitHub, besides content-based features, we also take semantic context and structural relation information within GitHub into consideration. As shown in Figure 1(b), we build an AHIN with four types of entities and seven types of relations as well as features of each entity, such that both content and relation information can be exploited simultaneously. The content features and relation information are introduced as follows.

Content Feature. For each type of node in AHIN, we extract different content features (as listed in Table **??**) and concatenate them to denote feature vector of the node. Note that for text content (e.g., readme), we merge all text information for each entity and feed them into the pre-trained BERT language model [Devlin *et al.*, 2018] to acquire the corresponding feature vector. For instance, for keyword feature, we extract all of the keywords from every source code file to represent the repository. Then, we select a set of malicious-oriented keywords based on word frequency, and feed the keyword set of each repository to BERT to obtain a keyword-based feature embedding. For enumerated content (e.g., file type such as executable file), we apply one-hot encoding to convert it to a binary feature vector.

Content Feature	
Repository	readme, program language, topic keywords
Keyword	keywords from every source code file
File	filename, SHA-1, and type of file
User	username, location, bio information
Relation Information	
R1	repository-contributed-by-user
R2	repository-forked-by-user
R3	repository-owned-by-user
R4	repository-stared-by-user
R5	repository-contain-file
R6	file-contain-keyword
R7	repository-contain-keyword

Table 1: Content feature and relation information in AHIN.

Relation. Besides content features, relations among different entities are also indispensable to judge the legitimacy of repository. For instance, a repository is more likely to be judged as malicious if it is owned by a cyber attacker (e.g., R3: repository-owned-by-user). Hence, we leverage seven types of edges (R1-R7 in Table 1) to describe the informative and complex relationships among four types of entities.

4.2 AHIN Representation Learning

Attention-based GCN. After constructing the AHIN, we design an attention-based GCN to learn the repository embedding. Specifically, we first transform the input attribute features of all entities to a common space. For each node $v_i \in V$ with type T_i and its attributes X_i, the transform function $g(X)$ is defined as follows:

$$g(X_i) = X_i W_{T_i}, \tag{2}$$

where W_{T_i} is the transform weight matrix for node type T_i ($|T| = 4$). Then, considering the converge of node features with Laplacian smoothing [Kipf and Welling, 2017], we use a two-layer GCN, where the node embedding is formulated:

$$\mathbf{f} = \text{GCN}(g(X), A) = \widetilde{A}\,\text{ReLU}\left(\widetilde{A}X\,W_T\,W^0\right)W^1. \quad (3)$$

To handle the heterogeneous graph with different types of nodes, we then leverage attention mechanism to capture different influence of neighbors for generating node embeddings. Given two neighboring nodes $v_i, v_j \in V$, we define the effect coefficient between them via the Softmax function:

$$e_{i,j} = \frac{\exp\left(\mathbf{f}_i\,\mathbf{f}_j^T\right)}{\sum_{v_j \in \mathcal{N}_{v_i}} \exp\left(\mathbf{f}_i\,\mathbf{f}_j^T\right)}, \quad (4)$$

where $\mathcal{N}_{v_i}$ denotes the neighbor set of v_i. Then, the attention-based embedding for v_i is given as follows:

$$\mathbf{f}_i^{att} = \sum_{j \in \mathcal{N}_{v_i}} e_{i,j} \cdot \mathbf{f}_j. \quad (5)$$

Further, for each repository node $v_r \in \mathcal{V}$, we feed its embedding $\mathbf{f}_r^{att}$ into a fully-connected layer to obtain its malicious score, i.e., $\hat{Y}_r^{att} = \mathbf{f}_r^{att} W^{att}$. The objective is to minimize the cross-entropy loss between the given labels Y and the prediction score $\hat{Y}^{att}$:

$$\mathcal{L}_{att} = -\sum_{r \in V_r} Y_r\,\log(\hat{Y}_r^{att}), \quad (6)$$

where V_r is the set of labeled repository nodes in AHIN and Y is the label of repository (malicious or benign).

Self-Supervised Augmentation. In order to exploit the unlabeled information in AHIN, we further introduce a self-supervised module to enhance the representation learning ability (see Figure 1(c)). Specifically, we randomly mask type attributes of some nodes in AHIN by using special masked indicators. Then, we employ a mean pooling layer AGG_{MEAN} to aggregate neighboring nodes information of the masked nodes and obtain their embeddings:

$$\mathbf{f}_k^m = \text{AGG}_{\text{MEAN}}\left(\mathbf{f}_1^{att}, ..., \mathbf{f}_{|\mathcal{N}_{v_k}|}^{att}\right) = \frac{1}{|\mathcal{N}_{v_k}|}\sum_{p \in \mathcal{N}_{v_k}} \mathbf{f}_p^{att}, \quad (7)$$

where $\mathcal{N}_{v_k}$ is the neighbor set of v_k. Moreover, for each masked node v_k, we feed $\mathbf{f}_k^m$ into a fully-connected layer, i.e., $\hat{Z}_k^m = \mathbf{f}_k^m W^m$, to obtain the predicted value of node type. The objective of this module is to minimize the cross-entropy between the given node types Z and the predictions:

$$\mathcal{L}_m = -\sum_{k \in V_m} Z_k\,\log(\hat{Z}_k^m), \quad (8)$$

where V_m is the set of masked nodes. Finally, the joint objective is defined as the combination of repository label classification and self-supervised masked node type prediction:

$$\mathcal{L} = \mathcal{L}_{att} + \mathcal{L}_m. \quad (9)$$

4.3 Task-aware Meta-Learning

MAML. We design the optimization strategy based on Model Agnostic Meta-Learning (MAML) [Finn *et al.*, 2017]. Specifically, it applies gradient-based algorithm that learns well initialized model parameters which can be quickly adapted to unknown new tasks. In this paper, we are given a set of tasks $\mathcal{T}$ defined as classification tasks on different types of repositories. We use the classification tasks of code repositories (w/o a COVID-19 theme) as training tasks while those of COVID-19 themed repositories as testing tasks. Data sampled from each task $\tau \in \mathcal{T}$ is divided into support set $\mathcal{S}_\tau$ and query set $\mathcal{Q}_\tau$. Then, the classifier is first updated to task-specific model (w/o a COVID-19 theme) using support set $\mathcal{S}_\tau$, and further optimized to task-agnostic model using $\mathcal{Q}_\tau$ of all tasks in training data. After sufficient training, the learnt model further adapt to new tasks with few data samples in support set, which is called meta-testing. Let ϕ be the set of model parameters. For a certai task τ, α is the step size, we firstly feed $\mathcal{S}_\tau$ to the model and calculate the loss $\mathcal{L}_\tau$ on $\mathcal{Q}_\tau$ to update parameters ϕ to $\phi_\tau^{'}$ through gradient descent:

$$\phi_\tau' = \phi - \alpha \bigtriangledown_\phi \mathcal{L}_\tau(\phi). \quad (10)$$

Task-aware Attention. Unlike traditional meta-learning methods [Finn *et al.*, 2017] treating each training task with equal weight for optimizing the meta-learner, we introduce the self-attention [Lee *et al.*, 2019] to reflect the importance of different tasks (see Figure 1(d)). Particularly, the task attention weight is computed as follows:

$$O_\tau = \frac{\exp\left(\text{MLP}(\mathbf{F}_\tau)\right)}{\sum_{\tau' \in \mathcal{T}} \exp(\text{MLP}(\mathbf{F}_{\tau'}))}, \quad (11)$$

where $\mathbf{F}_\tau = \text{MEAN}\left(\mathbf{f}_\tau\right)$ represents the task embedding which is averaged by all of repository embeddings of τ. Then, with the task-specific model parameter ϕ_τ' obtained from Eq. 10, the model parameters are updated as follows:

$$\phi \leftarrow \phi - \beta \bigtriangledown_\phi \sum_{\tau \in \mathcal{T}} O_\tau \mathcal{L}_\tau(\phi_\tau'), \quad (12)$$

where β is the learning rate, and $\mathcal{L}_\tau$ is the loss on $\mathcal{Q}_\tau$. The Meta-AHIN procedure is illustrated in Algorithm 1.

Algorithm 1 Training Procedure of Meta-AHIN

Require X, A: nodes features and adjacent matrix of AHIN
1: Obtain repository embeddings $\mathbf{f}^{att}$ and masked node embeddings $\mathbf{f}^m$ via Eq. 5 and Eq. 7 respectively
2: Compute the joint objective $\mathcal{L}$ via Eq. 9
3: **while** not convergae **do**
4: Sample a batch of training tasks τ from $\mathcal{T}$
5: **for** each τ **do**
6: Sample a support set $\mathcal{S}_\tau$ and a query set $\mathcal{Q}_\tau$
7: Update parameters via Eq. 10
8: **end for**
9: Calculate the importance of each task via Eq. 11
10: Update model parameters via Eq. 12
11: **end while**

Setting		1-shot		3-shot		5-shot		8-shot		15-shot	
Group	Model	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC
B1	Feature+LR	0.4566	0.5169	0.4640	0.5296	0.4639	0.5432	0.4869	0.5524	0.5313	0.5597
	Feature+DNN	0.4843	0.5266	0.4918	0.5308	0.4931	0.5554	0.5016	0.5591	0.5564	0.5770
B2	Feature+Protonet	0.4853	0.5278	0.5088	0.5524	0.5259	0.5765	0.6241	0.6438	0.6517	0.6722
	Feature+Matching	0.4975	0.5489	0.5325	0.5905	0.5478	0.6253	0.6325	0.6758	0.6417	0.6959
	Feature+MAML	0.5268	0.5771	0.5646	0.6204	0.5685	0.6527	0.6653	0.7178	0.6835	0.7356
B3	Deepwalk+DNN	0.4834	0.5210	0.4932	0.5360	0.5056	0.5458	0.5184	0.5691	0.5646	0.5964
	metapath2vec+DNN	0.5074	0.5243	0.5137	0.5508	0.5327	0.5687	0.5378	0.5731	0.5731	0.6111
	GCN+DNN	0.4989	0.5178	0.5255	0.5543	0.5541	0.5935	0.5848	0.6002	0.6190	0.6304
	GAT+DNN	0.5021	0.5160	0.5231	0.5500	0.5713	0.5920	0.5918	0.6280	0.6348	0.6415
B4	Deepwalk+MAML	0.5523	0.6049	0.5940	0.6505	0.6248	0.6867	0.6940	0.7500	0.7303	0.7693
	metapath2vec+MAML	0.5892	0.6399	0.6446	0.6835	0.6771	0.7145	0.7316	0.7788	0.7729	0.7960
	GCN+MAML	0.6376	0.6836	0.7051	0.7236	0.7464	0.7737	0.7718	0.8058	0.8043	0.8248
	GAT+MAML	0.6139	0.6661	0.6903	0.7129	0.7404	0.7622	0.7846	0.8147	0.8209	0.8382
Ours	**Meta-AHIN**	**0.7006**	**0.7173**	**0.7747**	**0.7768**	**0.8149**	**0.8283**	**0.8591**	**0.8687**	**0.8640**	**0.8851**

Table 2: Performance comparisons of all methods with different support data sizes (shot numbers).

5 Experiments and Analysis

In this section, we first introduce our collected data. Then we conduct extensive experiments to evaluate the performance of our proposed model and show related analysis.

5.1 Data Collection

We evaluate our model with the dataset crawled from GitHub. The dataset is divided into two groups: code repository with and w/o a COVID-19 theme. we crawl the open source repositories in GitHub from Feb 2020 to Dec 2020. Particularly, for COVID-19 themed repository, we crawl them based on a set of COVID-19 related keywords (e.g., coronavirus) and the corresponding user profile. There are seven types of repositories based on the application scenarios (e.g., tracking application). Four of them are regular repositories viewed as training tasks and the other three are COVID-19 themed repositories viewed as testing tasks. To obtain the ground truth, we apply a two-step mechanism: (i) we implement VirusTotal [Virustotal, 2017] having over 70 anti-malware scanning tools to validate the legitimacy of repository; and then (ii) we ask anti-malware experts to confirm the legitimacy of repositories. Hence, we obtain 20,895 repositories including 6,965 malicious (835 with a COVID-19 theme), 13,930 benign repositories (1,670 with a COVID-19 theme). Specifically, we construct an AHIN with 174,661 nodes (i.e., 20,895 repository nodes, 52,530 keyword nodes, 82,451 file nodes, and 18,785 user nodes) and 696,464 edges of R1-R7.

5.2 Baseline Methods

We compare our model with following baseline methods:

- **Traditional Classification.** We take the attributes of each repository as feature vector and feed it to logistic regression (**LR**) or 5-layer deep neural network (**DNN**) (B1).
- **Meta-Learning.** For meta-learning baseline methods, besides **MAML** [Finn *et al.*, 2017], we also employ two popular few-shot learning models, Matching Network (**Matching**) [Vinyals *et al.*, 2016] and Prototypical Network (**Protonet**) [Snell *et al.*, 2017] to train a meta-learner to classify COVID-19 themed repository (B2).
- **Graph Embedding.** We also utilize four popular graph representation learning models to learn repository embeddings in AHIN: **Deepwalk** [Perozzi *et al.*, 2014], **metapath2vec** [Dong *et al.*, 2017], **GCN** [Kipf and Welling, 2017], and **GAT** [Veličković *et al.*, 2018]. Then the learned repository embeddings are fed to a generic 5-layer DNN (B3). In addition, we also implement MAML (B4) framework to train these four models.

5.3 Evaluation Metrics and Parameter Settings

To evaluate the performances of our model and baseline methods, we adopt two widely-used metrics: F1 score (**F1**) and accuracy (**ACC**). The experiments are conducted under the environment of the Ubuntu 16.04 OS, plus Intel i9-9900k CPU, GeForce GTX 2080 Ti Graphics Cards, and 64 GB of RAM. For meta-learning models, inner-level and outer-level learning rate are set as 0.003 and 0.001 respectively. The update steps for the meta-testing task is 2. For graph representation learning models: the dimension of node embedding is 256. We apply Pytorch to implement all methods and use 10-fold cross-validations.

5.4 Comparison with Baseline Models

The performances of all models are reported in Table 2, where the best results are highlighted in bold and the best baseline results are indicated by underline. The shot number denotes the support data size. According to this table, we can find that (i) The performance of meta-learning models (with repository feature (B2) or node embedding (B4)) are better than traditional feature based classification models (B1). Particularly, MAML outperforms Protonet and Matching. (ii) For graph representation learning models (B3 and B4), GCN has the

Setting		3-shot		8-shot	
T_{id}	Model	F1	ACC	F1	ACC
T_1	GCN+MAML	0.7113	0.7246	0.7805	0.8134
	GAT+MAML	0.725	0.7158	0.7885	0.8253
	Meta-AHIN	**0.7767**	**0.7728**	**0.8437**	**0.8658**
T_2	GCN+MAML	0.6856	0.6937	0.7532	0.7842
	GAT+MAML	0.6638	0.6854	0.7625	0.7853
	Meta-AHIN	**0.7583**	**0.7641**	**0.8335**	**0.8494**
T_3	GCN+MAML	0.6937	0.7158	0.7739	0.8046
	GAT+MAML	0.6878	0.7103	0.7870	0.8059
	Meta-AHIN	**0.7681**	**0.7613**	**0.8543**	**0.8619**

Table 3: Results of different models for each type/task (T_{id}).

best performance when the support data size is small, while GAT has the best performance when the support data size is larger. (iii) In all cases, Meta-AHIN significantly outperforms all baseline methods for COVID-19 themed malicious repository detection, demonstrating the effectiveness of our model design. (iv) To further show performance for each type of COVID-19 themed malicious repository, we report results of our model and the best baseline methods in Table 3. It is easy to find that Meta-AHIN is robust for different tasks and significantly outperforms the baseline models.

5.5 Ablation Studies

Since Meta-AHIN integrates three essential components (i.e., attention-based GCN (AGCN), self-supervised module (Ssup), and task-attention MAML (taMAML)), we conduct extensive ablation studies to analyze the contributions of different components by removing each of them independently (see Table 4). Specifically, we remove MAML (A1) and task-attention (A2) from our model respectively. We can see that both (especially A1) results drop significantly, showing that both MAML and task-attention have large contribution to Meta-AHIN. In addition, we remove self-supervised module (A3) and neighbor attention of GCN (A4) form our model. We found that Meta-AHIN is better than A3 and A4, demonstrating that both self-supervised learning and neighbor attention are effective in enhancing the model.

Model	3-shot		8-shot	
	F1	ACC	F1	ACC
Meta-AHIN	**0.7747**	**0.7768**	**0.8591**	**0.8687**
– MAML (A1)	0.5559	0.5812	0.6238	0.6570
– attention on MAML (A2)	0.7373	0.7460	0.8150	0.8225
– self-supervised (A3)	0.7489	0.7409	0.8239	0.8360
– attention on GCN (A4)	0.7625	0.7605	0.8379	0.8513

Table 4: Results of model variants.

5.6 Comparison with Industrial Methods

We further validate the performance of Meta-AHIN by comparisons with other industrial security products including LGTM (provided by GitHub) and other popular anti-malware products integrated in VirusTotal [Virustotal, 2017] (i.e., DrWeb, McAfee, and Avast). From Table 5, we observe that Meta-AHIN achieves around 50% and 20% accuracy improvement in comparison with LGTM and Avast respectively. It shows that our model can significantly improve the performance of COVID-19 themed malicious repository detection.

Method	Version	F1	ACC
LGTM	-	0.1459	0.3592
DrWeb	7.0.46.3050	0.5958	0.6174
McAfee	6.0.6.653	0.6187	0.6360
Avast	18.4.3895.0	0.6679	0.6881
Meta-AHIN	-	**0.8640**	**0.8851**

Table 5: Comparisons with other industrial security products.

5.7 Embedding Visualization

To better show the effectiveness of our model, we visualize embeddings of one type of repositories generated by Meta-AHIN and GAT with MAML in Figure 2. The green and orange points represent the embeddings for benign and malicious repositories respectively. In Figure 2, we observe that our model splits benign and malicious repositories better than GAT+MAML, which shows the superiority of our model.

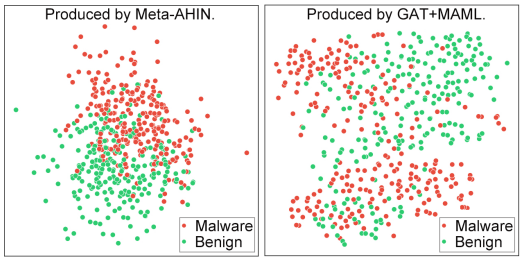

Figure 2: Embedding visualization of malicious/benign repositories.

6 Conclusion

In this paper, we develop a novel model (i.e., Meta-AHIN) to solve the problem of COVID-19 themed malicious repository detection in social coding platform (i.e., GitHub) accounting the constraint of few labeled samples. In particular, our proposed Meta-AHIN employs an attention-based GCN to learn repository embeddings in AHIN constructed based on GitHub data. To exploit unlabeled information in AHIN, a self-supervised module is proposed and incorporated into the model. Moreover, a task attention-based meta-learning framework is developed to address the challenge of few labeled samples and to optimize model parameters. The experimental results based on the real-world dataset from GitHub demonstrate the effectiveness of our proposed model.

Acknowledgments. Y. Qian, Y. Zhang and Y. Ye's work is partially supported by the NSF under grants IIS-2027127, IIS-2040144, IIS-1951504, CNS-2034470, CNS-1940859, CNS-1814825, OAC-1940855 and ECCS-2026612, the DoJ/NIJ under grant NIJ 2018-75-CX-0032.

References

[Bruna *et al.*, 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dong *et al.*, 2017] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, 2017.

[Fan *et al.*, 2019] Yujie Fan, Yiming Zhang, Shifu Hou, Lingwei Chen, Yanfang Ye, Chuan Shi, Liang Zhao, and Shouhuai Xu. idev: Enhancing social coding security by cross-platform user identification between github and stack overflow. In *IJCAI*, 2019.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[Finn *et al.*, 2018] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Lee and Choi, 2018] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2018.

[Lee *et al.*, 2019] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, 2019.

[Meli *et al.*, 2019] Michael Meli, Matthew R McNiece, and Bradley Reaves. How bad can it git? characterizing secret leakage in public github repositories. In *NDSS*, 2019.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[Ragkhitwetsagul *et al.*, 2019] Chaiyong Ragkhitwetsagul, Jens Krinke, Matheus Paixao, Giuseppe Bianco, and Rocco Oliveto. Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, 2019.

[Rokon *et al.*, 2020] Md Omar Faruk Rokon, Risul Islam, Ahmad Darki, Evangelos E Papalexakis, and Michalis Faloutsos. Sourcefinder: Finding malware source-code from publicly available repositories in github. In *RAID*, 2020.

[Semmle, 2019a] Semmle. Codeql for research. https://securitylab.github.com/tools/codeql, 2019. Accessed: 2021-05-27.

[Semmle, 2019b] Semmle. Lgtm. https://github.com/marketplace/lgtm, 2019. Accessed: 2021-05-27.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

[Virustotal, 2017] Virustotal. Virustotal: R client for the virustotal api. https://cran.r-project.org/web/packages/virustotal/index.html, 2017. Accessed: 2021-05-27.

[Waggoner and Markowitz, 2021] John Waggoner and Andy Markowitz. Coronavirus scams spreading as fraudsters follow the headlines. https://www.aarp.org/money/scams-fraud/info-2020/coronavirus/, 2021. Accessed: 2021-05-27.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[Ye *et al.*, 2017] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM CSUR*, 50(3):1–40, 2017.

[Ye *et al.*, 2020a] Yanfang Ye, Yujie Fan, Shifu Hou, Yiming Zhang, Yiyue Qian, Shiyu Sun, Qian Peng, Mingxuan Ju, Wei Song, and Kenneth Loparo. Community mitigation: A data-driven system for covid-19 risk assessment in a hierarchical manner. In *CIKM*, 2020.

[Ye *et al.*, 2020b] Yanfang Ye, Shifu Hou, Yujie Fan, Yiming Zhang, Yiyue Qian, Shiyu Sun, Qian Peng, Mingxuan Ju, Wei Song, and Kenneth Loparo. $alpha$-satellite: An ai-driven system and benchmark datasets for dynamic covid-19 risk assessment in the united states. *IEEE JBHI*, 2020.

[Zhang *et al.*, 2019] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *KDD*, 2019.

[Zhang *et al.*, 2020a] Yiming Zhang, Yujie Fan, Shifu Hou, Yanfang Ye, Xusheng Xiao, Pan Li, Chuan Shi, Liang Zhao, and Shouhuai Xu. Cyber-guided deep neural network for malicious repository detection in github. In *ICKG*, 2020.

[Zhang *et al.*, 2020b] Yiming Zhang, Yiyue Qian, Yujie Fan, Yanfang Ye, Xin Li, Qi Xiong, and Fudong Shao. dstylegan: Generative adversarial network based on writing and photography styles for drug identification in darknet markets. In *ACSAC*, 2020.