FEDACK: Federated Adversarial Contrastive Knowledge Distillation for Cross-Lingual and Cross-Model Social Bot Detection

Yingguang Yang
USTC
Heifei, China
dao@mail.ustc.edu.cn

Renyu Yang*
University of Leeds
Yorkshire, United Kingdom
r.yang1@leeds.ac.uk

Hao Peng*
Beihang University
Beijing, China
penghao@buaa.edu.cn

Yangyang Li*
NERC-RPP, CAEIT
Beijing, China
liyangyang@ctec.com.cn

Tong Li
Tsinghua University
Beijing, China
tongli@mail.tsinghua.edu.cn

Yong Liao
USTC
Beifei, China
yliao@ustc.edu.cn

Pengyuan Zhou*
USTC
Beifei, China
pyzhou@ustc.edu.cn

ABSTRACT
Social bot detection is of paramount importance to the resilience and security of online social platforms. The state-of-the-art detection models are siloed and have largely overlooked a variety of data characteristics from multiple cross-lingual platforms. Meanwhile, the heterogeneity of data distribution and model architecture make it intricate to devise an efficient cross-platform and cross-model detection framework. In this paper, we propose FEDACK, a new federated adversarial contrastive knowledge distillation framework for social bot detection. We devise a GAN-based federated knowledge distillation mechanism for efficiently transferring knowledge of data distribution among clients. In particular, a global generator is used to extract the knowledge of global data distribution and distill it into each client’s local model. We leverage local discriminator to enable customized model design and use local generator for data enhancement with hard-to-decide samples. Local training is conducted as multi-stage adversarial and contrastive learning to enable consistent feature spaces among clients and to constrain the optimization direction of local models, reducing the divergences between local and global models. Experiments demonstrate that FEDACK outperforms the state-of-the-art approaches in terms of accuracy, communication efficiency, and feature space consistency.

CCS CONCEPTS
• Computing methodologies → Adversarial learning; Machine learning.

KEYWORDS
social bot detection, contrastive federated learning, knowledge distillation

*Corresponding authors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '23, May 1–5, 2023, Austin, TX, USA
© 2023 Copyright held by the owner/authors(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9416-1/23/04... $15.00
https://doi.org/10.1145/3543507.3583500

1 INTRODUCTION
Social bots imitate human behaviors on social networks such as Twitter, Facebook, Instagram, etc. [43]. Millions of bots, typically controlled by automated programs or platform APIs [1], attempt to sneak into genuine users as a disguise to pursue malicious goals such as actively engaging in election interference [11, 17], misinformation dissemination [8], and privacy attacks [37]. Bots are also involved in spreading extreme ideologies [3, 18], posing threats to online communities. Effective bot detection is necessitated by the jeopardized user experience on social media platforms and the induction of unfavorable social effects.

There is a new yet understudied problem in bot detection – a society of bots tend to be exposed to multiple social platforms and behave as collaborative cohorts. Existing bot detection solutions largely rely on user property features extracted from metadata [9, 41], or features derived from textual data such as a tweet post [15, 39], before adopting graph-based techniques to explore neighborhood information [14, 42, 46]. While such models can uncover camouflage behaviors, they are siloed and subject to the amount, shape, and quality of platform-specific data. To this end, Federated Learning (FL) has become the main driving force of model training across heterogeneous platforms without disclosing local private datasets. Some studies [32, 44, 45, 49] augmented FL by Generative Adversarial Networks (GANs) and Knowledge Distillation (KD) in a data-free manner to safeguard privacy against intrusions. However, they have the following limitations:

i) Restriction to homogeneous model architecture. As FL models assume homogeneous model architecture on a per client basis – which however no longer holds – participants are stringently required to conform to the same model architecture managed by a central server. It is therefore imperative to enable each individual platform to customize heterogeneous models as per unique data characteristics. ii) Inconsistent feature learning spaces. The state-of-the-art...
Federated KD approaches are largely based on image samples and assume consistent feature space. However, the distinction between global and local data distribution tends to result in non-negligible model drift and inconsistent feature learning spaces, which will in turn cause performance loss. It is highly desirable to align feature spaces among different clients to improve the global model performance. iii) Sensitivity to content language. Textual data based anomaly detection approaches to date are sensitive to the languages that the model is built upon. Existing solutions for cross-lingual content detection in online social networks either substantially raise computational costs [10, 13, 50] or require labor-intensive feature engineering to identify cross-lingual invariant features [7, 12, 36]. Arguably, how to incorporate into a synergetic model a variety of customized models with heterogeneous data in different languages to enable consistent feature learning space is still under-explored.

This paper proposes FedACK, a novel bot detection framework through Federated Adversarial learning Contrastive learning and Knowledge distillation. FedACK envisions to enable personalization of local models in a consistent feature space across different languages (see Fig. 1). We present a new federated GAN-based knowledge distillation architecture – a global generator is used to extract the knowledge of global data distribution and to distill the knowledge into each client’s local model. We elaborate two discriminators – both globally shared and local – to enable customized model design and use a local generator for data enhancement with hard-to-decide samples. Specifically, the local training on each client side is regarded as a multi-stage adversarial learning procedure to efficiently transfer data distribution knowledge to each client and to learn consistent feature spaces and decision boundaries. We further exploit contrastive learning to constrain the optimization direction of local models and reduce the divergences between local and global models. To replicate non-IID data distribution across multi- platforms, we employ two real-world Twitter datasets, partitioned by the Dirichlet distribution. Experiment shows that FedACK outperforms the state-of-the-art approaches on accuracy and achieves competitive communication efficiency and consistent feature space. This work makes the following contributions.

- To the best of our knowledge, FedACK is the first social bot detection solution based on federated knowledge distillation that envisions cross-lingual and cross-modal bot detection.
- contrast and adversarial learning mechanisms for enabling consistent feature space for better knowledge transfer and representation when tackling non-IID data and data scarcity among clients.
- FedACK outperforms other FL-based approaches by up to 15.19% accuracy improvement in high heterogeneity scenarios, and achieves up to 4.5x convergence acceleration against the 2nd fastest method.

To enable replication and foster research, FedACK is publicly available at: https://github.com/846468230/FedACK.

2 PRELIMINARIES
2.1 Background

Federated learning (FL). FL is a distributed learning paradigm that allows clients to perform local training before aggregation without sharing clients’ private data [4, 22, 27, 28, 30]. While promising, FL can have inferior performance particularly when training data is not independent and identically distributed (Non-IID) on local devices [25, 47], which could make the model deflected to a local optimum [20]. Most of the existing works mainly fall into two categories. The first is introducing additional data or using data enhancement to address model drift issue caused by the non-IID data. FedGAN [32] trains a GAN to tackle the non-IID data challenge in a communication efficient way but inevitably produces bias. FedGen [49] and FedDTG [45] utilize generator to simulate the global data distribution to improve performance. The second category mainly focuses on local regularization. FedProx [25] adds an optimization item to local training, and SCAFFOLD [20] uses control variants to correct the client-drift in local updates while guaranteeing a faster convergence rate. FedDyn [2] and MOON [24] constrain the direction of local model updates by comparing the similarity between model representations to align the local and global optimization objectives. However, these approaches either directly model aggregation to get the global model that causes non-negligible performance deterioration [35] or neglect the impact of data heterogeneity, which may lead to knowledge loss of local data distribution during the model aggregation.

Federaled knowledge distillation (KD). KD is first introduced to use compact models to approximate the function learned by larger models [5]. Knowledge is formally referred to as the softened logits, and in a typical KD, a student model absorbs and mimics the knowledge from teacher models [19]. KD is inherently beneficial for FL since it requires less or no data to enable the model to understand the data distribution. FedDistill [33] jointly refines logits of user-data obtained through model forward propagation and performs global knowledge distillation to reduce the global model drift problem. FedDF [26] proposes an ensemble distillation for model fusion and trains the global model through averaged logits from local models. FedGen [49] combines each local model’s average logits as the teacher in KD to train a global generator. FedFTG [44] uses each local model’s logit as the teacher to train a global generator and distill knowledge by using the pseudo-data generated by the global generator to fine tune the global model. However, none of them focuses on enabling consistent feature space, which will lead to ineffective knowledge dissemination. FL and KD have been largely overlooked to date in social bot detection, which is investigated in a siolated way [8]. FedACK can fill this gap through enhanced adversarial learning with a shared discriminator and an exclusive discriminator to support designated cross-modal bot detection.

Figure 1: Incorporating multiple social platforms with heterogeneous languages, context spaces and model architectures
Cross-lingual content detection in social networks. Publishing fake or misleading contents through social bots on social networks in different languages has become the norm rather than the exception. [7, 12] have explored the possibilities of cross-lingual content detection by seeking cross-lingual invariant features. There is also a huge body of research on cross-lingual text embedding and model representation [10, 13, 29, 31, 50] for detecting hate speeches, fake news or abnormal events. These works usually require huge efforts in finding cross-lingual invariants in the data, and thus computational inefficiency. While InfoXLM [6] could be applied in FedACK as a substitute for our cross-lingual module, it may involve additional overhead given only a few mainstream languages in social platforms. FedACK implemented text embedding by mapping the cross-lingual texts into the same context space.

2.2 Problem Scope
We consider federated social bot detection setting that includes a central server and $K$ clients holding private datasets $\{D_1, \ldots, D_K\}$. These private datasets contain benign accounts and different generations of bots. Presumably, different model architectures or parameters exist on different clients. FedACK focus on meta and text data, rather than multimodal data. Instead of collecting raw client data, the server tackles heterogeneous data distribution across clients and aggregates model parameters for the shared networks. The objective is to minimize the overall error among all clients:

$$\arg\min_w \mathcal{L}(w) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k} \mathcal{L}(x_i^k, y_i^k; w),$$

where $\mathcal{L}$ is the loss function that evaluates the prediction model $w$ on the data sample $(x_i^k, y_i^k)$ of $D_k = \{(x_i^k, y_i^k)\}_{i=1}^{N_k}$ in $k$-th client.

3 METHODOLOGY

As shown in Fig. 2, FedACK consists of cross-lingual mapping, backbone model, and federated adversarial contrastive KD.

3.1 Cross-Lingual Mapping
We adopt a Transformer encoder-decoder-based method to achieve the alignment of different language contents. In essence, given a $m$-word text $x = \{x_1, \ldots, x_m\}$ in one language and the corresponding $n$-word text $y = \{y_1, \ldots, y_n\}$ in another language, we use an Encoder $\phi_E$ to transform the source text $x$ and the target text $y$ into context representations $z_k = \{z_{x1}, \ldots, z_{xm}\}$ and $z_y = \{z_{y1}, \ldots, z_{yn}\}$. We devise a Mapper $M$ for converting between two context representation spaces, i.e., $z_k' = M(z_k)$ and $z_y' = M(z_y)$.

We introduce an adversarial mechanism for optimizing $M$ so that the original $z_k$ and the mapped $z_k'$ can be sufficiently similar. We first obtain the embedding of the context representations:

$$\bar{z}_x = \frac{1}{m} \sum_{k=1}^{m} z_{x_k}, \quad \bar{z}_y' = \frac{1}{n} \sum_{k=1}^{n} z_{y_k}.'$$

Then we use the discriminator $D$ to distinguish whether an embedding $\bar{z}_x$ or $\bar{z}_y'$ is forward propagated from $M$ ($\bar{z}_y$ is equal to $\bar{z}_y'$). Accordingly, the loss of the discriminator is defined as:

$$\mathcal{L}_{dis} = \mathcal{L}_{dis_x} + \mathcal{L}_{dis_y},$$

$$\mathcal{L}_{dis_x} = (D(\bar{z}_x) - y_{z_x})^2 + (D(\bar{z}_x') - y_{z_x'})^2,$$

where the label $y_z$ is set as 0 if an embedding $\bar{z}$ is mapped from $M$. We combine the encoder $\phi_E$ and the mapper $M$ into the generator. Similarly, the loss of the generator is:

$$\mathcal{L}_{gen} = (D(\bar{z}_x') - y_{z_y})^2 + (D(\bar{z}_y') - y_{z_y})^2,$$

where the label $y_z$ is always set as 1 to produce sufficiently similar representations to confuse the discriminator. We collect the context representations $(z_x, z_y, z_x', z_y')$ and decode them with a Decoder $\phi_D$ and generate the corresponding translation $(\tilde{z}_y, \tilde{z}_x, \tilde{z}_x', \tilde{z}_y')$, where $\tilde{z}_y = \phi_D(z_x)$ and other terms are calculated in a similar way.

The loss of the Transformer is therefore defined as the cross-entropy loss between $x$ and $y$:

$$\mathcal{L}_{trans} = \mathcal{L}_{z_x} + \mathcal{L}_{z_y} + \mathcal{L}_{z_x'} + \mathcal{L}_{z_y'},$$

$$\mathcal{L}_{z_x} = -\sum_{i=1}^{n} \log P(y_i | \bar{z}_y < t, z_x),$$

$$\mathcal{L}_{z_x'} = -\sum_{i=1}^{n} \log P(y_i | \bar{z}_y' < t, z_x').$$

3.2 Backbone Model for Feature Extraction
The main task is to let user-level backbone feature extractor model $c$ to extract features per user metadata (e.g., account properties) and textual data (e.g., tweets). We concatenate the key items extracted from the metadata into the property vector $u_p$, following the similar way as [41, 42], which is converted into user’s property representation $r_p$ by a Multi-layer Perceptron $r_p = MLP(u_p)$. For textual data, assume a user has posted $M$ Tweets $u_t = \{t_1, \ldots, t_i, \ldots, t_M\}$, possibly in the form of different languages; $t_i$ represents the $i$-th word in the $j$-th tweet. We leverage the Encoder and Mapper delivered by the aforementioned cross-lingual module to convert each $u_t$ into a uniformed contextual space, i.e., $z_{u_t} = \{z_{t1}, \ldots, z_{t1'}, \ldots, z_{tM}, \ldots, z_{tM}'\}$. A Convolutional Neural Networks (CNN) layer, i.e., TextCNN[21], is then used to obtain the tweet-level representation $h_{u_t}$:

$$h_{u_t} = \{h_{t1}, \ldots, h_{t1}', \ldots, h_{tM}, \ldots, h_{tM}'\}.$$

An attention layer is used to quantify the influence of each tweet on the overall semantics of the user and to calculate the user-level tweet representation $r_t$ through weighted aggregation of all tweets. The complete user-level representation $r_u$ is:

$$r_u = MLP(\text{concat}(r_p, r_t)).$$

3.3 Federated Adversarial Contrastive KD
Architecturally, we follow the conventional server-client design for the Federated GAN-based KD. A client $k$ contains a global generator $G$ for KD, and a local generator $G_k$ for data enhancement. We use two discriminators $D_1$ shared across all clients with the same architecture and initial parameters but trained on local data, and $D_2$ exclusively designated for each client to satisfy its individual demand. Each client uses a backbone model $e$ to get the user’s representation $r_u = e(x_u)$ from local data $D_k$. As a significant departure from the state-of-the-arts, a new multi-stage adversarial mechanism is proposed for jointly optimizing classification in both discriminators at intra-client level and a contrastive mechanism for aligning different feature spaces across clients. The notations are detailed in Appendix A.1.
3.3.1 Local Adversarial Contrastive KD. The adversarial learning on a per client basis include the following multi-phases.

**Stage-1: Training D1 and D2 as classifiers.** We aim to facilitate the two models to learn different decision boundaries for the same class of samples and compress the feature space of the feature extractor. To tackle non-IID data distribution and data scarcity among clients, we treat the shared global generator $G$ as a teacher network and distill the knowledge of global data distribution from it. For each sample $(x_i, y_i)$, $G$ uses a standard Gaussian noise $z \sim N(0, 1)$ and label $y_i$ to generate pseudo-data $\hat{x} = G(z, y_i; \theta_G)$. $(x_i, \hat{x})$ is fed into $D_1$ to obtain the probability distributions $(p, \hat{p})$. We make $D_1$ fit probability $p$ of real data $x_i$ close to $\hat{p}$ to distill knowledge by minimizing Eq. (8):

$$L_{\text{dis}}^k = \frac{1}{N_k} \sum_{i=1}^{N_k} D_{KL}(\sigma(D_1(x_i)))||\sigma(D_1(\hat{x}))).$$

(8)

where $\sigma$ is the softmax function, $D_{KL}$ is the Kullback–Leibler divergence and $x_i^k = \epsilon(x_i^k)$). Similar $L_{\text{dis}}^k$ is performed for $D_2$.

We induce an adversarial loss to measure the probability difference between $D_1$ and $D_2$:

$$L_{\text{adv}}^k = \frac{1}{N_k} \sum_{i=1}^{N_k} D_{KL}(\sigma(D_1(r_i^k)))||\sigma(D_2(r_i^k))).$$

(9)

In essence, maximizing Eq. (9) can produce distinct decision boundaries between two discriminators to compress overlapping decision space. Intuitively, this is because $\epsilon$ would learn more precise feature space if the representations could be correctly classified by both discriminators. Similarity to Eq. (9), we calculate adversarial loss $L_{\text{adv}}^k$ on the pseudo-data generated by the local generator $G_k$ and minimize it to intensify the above process. $D_1$ and $D_2$ also need to correctly classify the pseudo-data randomly generated by $G$ to deal with the issue of unbalanced and scarce data. To sum up, the overall loss function of the discriminators is:

$$L_D^k = L_{\text{cls}}^k + \alpha(L_{\text{dis}}^k + L_{\text{dis}}^k) + \gamma(L_{\text{adv}}^k - L_{\text{adv}}^k).$$

(10)

**Stage-2: Training $\epsilon$.** We minimize the adversarial loss Eq. (9) to reduce the difference between the probability outputs of two discriminators for the same feature. This means that the same data point can fall on the same side of the decision boundaries of the two discriminators. The feature space of the feature extractor is compressed and enforced to generate more precise features.

We use contrastive learning to navigate the optimization direction of feature extractor, thereby overcoming the model drift between local extractors and the global extractor. We expect the newly optimized $r_i^k$ to produce a representation $r = r_i^k(x_i^k)$ as close as possible to the $r_{\text{glo}} = r_i^k(x_i^k)'$ generated by the global extractor $\epsilon$ while as far away as possible from the last round result of the feature extractor $r_{\text{pre}} = r_i^k(x_i^k)$. We define the following contrastive loss function:

$$L_{\text{con}}^k = -\log \frac{\exp(\text{sim}(r, r_{\text{glo}})/\tau)}{\exp(\text{sim}(r, r_{\text{glo}})/\tau) + \exp(\text{sim}(r, r_{\text{pre}})/\tau)}.$$  

(11)

where $\text{sim}$ is the similarity measure function and $\tau$ denotes a temperature parameter. This can not only reduce the local models’ drift but also serve as a bridge with adversarial learning to make models of different clients have a consistent feature space. The loss function extractor is defined as follows:

$$L_{e}^k = L_{\text{cls}}^k + \gamma L_{\text{adv}}^k + \mu \frac{1}{N_k} \sum_{i=1}^{N_k} L_{\text{con}}^k.$$  

(12)

**Stage-3: Training $G_k$.** We maximize $L_{\text{adv}}^k$ to ensure $G_k$ can generate pseudo-data that falls near the decision boundaries of two discriminators. This enforces such boundaries closer to the coincidence region, which further compresses the consistent feature space of the $\epsilon$. As $G_k$ only generates the same data for each class, we add the diversity loss $L_{\text{var}}$ to improve the diversity of the generated data and prevent model collapse:

$$L_{\text{var}} = \frac{1}{N_k} \sum_{i=1}^{N_k} (-||\hat{x}_i - x_i||_2 ||y_i - z_i||_2).$$  

(13)

where $\hat{x}_i = G_k(\bar{z}_i, \bar{y}_i)$. The overall loss for generator $G_k$ can be calculated through:

$$L_g^k = L_{\text{cls}}^k - L_{\text{adv}}^k + L_{\text{var}}.$$  

(14)

3.3.2 Server Aggregation Knowledge Extraction. In each communication round, the client $k$ uploads the local parameters of $(\theta_{G}^k, \theta_{D}^k)$ to the server once the local training is finished, then waits for the
updated global $G$, $D$ and $\epsilon$ from the server to start a new round of local training. Once the server receives the latest parameters of the participating clients, it performs the model aggregation by weighted average and gets the updated global $\epsilon$ and $D$:

$$\theta_s = \frac{M}{\sum_{k=1}^{M} \|D_k\|} \theta_D = \frac{\sum_{k=1}^{M} \|D_k\| \|\theta_D^k\|}{\sum_{k=1}^{M} \|D_k\|},$$

where $M$ is the number of participating clients.

To ensure global data distribution can adapt to local distributions that may largely drift from each other, we exploit the global discriminator $D$ and global generator $G$ for global knowledge extraction, without a need for server-side KD using proxy data. The experiments aim to answer the following questions:

### 3.4 Pipeline of FedACK

Alg. 1 summarizes the overall pipeline of FedACK. The cross-lingual model $\phi_E$ and $M$ for the backbone model $\epsilon$ are first trained on the server (Line 2) and distributed to all clients. In each communication round, FedACK first broadcasts the up-to-date $G$, $\epsilon$ and $D$ to a selected subset of clients $S_t$ (Line 5). Each client optimizes all the required models $D_K^k, D_E^k, \epsilon$ and $G_k$ using local data (Lines 8-17). When the parallel optimization completes, the server aggregates the clients’ parameters in this round to update the global parameters $\theta_s, \theta_D$ and to optimize the global generator $G$ (Lines 20-21).

## 4 EVALUATION

The experiments aim to answer the following questions:

- **Q1.** How does FedACK perform in classification under different data distribution scenarios?
- **Q2.** How does FedACK perform in learning efficiency?
- **Q3.** Can FedACK learn consistent feature space across clients?
- **Q4.** What is the effect of the different parameter values in different stages of FedACK?
- **Q5.** How does the cross-lingual module perform when combined FedACK and other baselines?

### 4.1 Experimental Setup

#### 4.1.1 Software and Hardware

FedACK is implemented with Python 3.8.10, PyTorch 1.7.1 and runs on two servers, one is equipped with NVIDIA Tesla V100 GPU, 2.20GHz Intel Xeon Gold 5220 CPU and 512GB RAM, and the other is equipped with NVIDIA GeForce RTX 3090 GPU, 3.40GHz Intel Xeon Gold 6246 CPU and 256GB RAM.

#### 4.1.2 Datasets

We conduct experiments on two Twitter bot datasets **Vendor-19** [40] and **TwiBot-20** [16], the largest ones in the public domain by far. We mix the **Vendor-19** with a dataset of benign accounts **Verified** which is presented in [41]. The newly released **TwiBot-20** dataset exposes users’ social relationships and enables the use of advanced graph representation-based algorithms. More dataset statistics are outlined in Appendix A.2.

#### 4.1.3 Data heterogeneity

We use Dirichlet Distribution Dir ($\alpha$) to mock the non-IID given in [23] and split the bot dataset with heterogeneity. $\alpha$ is an indicator of Dirichlet distribution – the smaller $\alpha$ is, the more heterogeneous the data distribution is.

#### 4.1.4 Baselines

As federated KD provided a natural pathway to privacy preservation without sharing original data – the key concern in cross-platform bot detection – federated KD-based approaches baselines that can handle heterogeneity of non-IID data are the pedestal focus of the comparison. **FedAvg** and [28]. **FedProx** [25] improve the local model training and update under heterogeneity through adding an optimization item. **FedDF** [26] employs data-free knowledge distillation to improve the global model on server side. **FedEnsemble** [34] uses an ensemble mechanism for combining the output of all models to predict a specific sample. **FedDistill** [34] shares label-wise average of logit vectors among users for data-free knowledge distillation without network parameter shared. **FedGen** [49] and **FedFTG** [44] offer flexible parameter sharing and knowledge distillation.
We vary the hyperparameter dataset partition with a hidden dimension of 512. TextCNN [21] used in the discriminators in the samples of different categories in the datasets are balanced, we cross-platform scenarios where multiple languages coexist. Since the samples of different categories in the datasets are balanced, we simply use accuracy and deviation as the main metrics.

4.1.6 Methodology and Metrics. The comparison is five-fold: 1) effectiveness (model accuracy and capability of handling heterogeneity), 2) efficiency (the number of communication rounds required to achieve a target accuracy) and 3) the effect of learning consistent feature space. 4) sensitivity (variation of model accuracy under different hyperparameter settings) 5) cross-lingual validation (performance gains of our proposed cross-lingual modules in cross-platform scenarios where multiple languages coexist). Since the samples of different categories in the datasets are balanced, we simply use accuracy and deviation as the main metrics.

4.2 Effectiveness (Q1)

We vary the hyperparameter dataset partition $\alpha$ from $\{1, 0.5, 0.1, 0.05\}$ for each dataset to validate the performance of different methods with varying degrees of data distribution heterogeneity. The darkness of coloring represents the sample number of a specific class stored on a client. As shown in Fig. 3, increased data heterogeneity (e.g., $\alpha = 0.1$) leads to more clients store only one class of samples.

Accuracy Comparison. Table 1 compares the accuracy among baseline algorithms. All experiments are repeated over 3 random seeds. Overall, our method outperforms all baselines in any scenario. FedACK achieves 1.26%–10.03% accuracy improvement in absolute terms when compared with the runner-up methods (i.e., FedGen, FedFTG). While FedProx can achieve relatively competitive performance when data heterogeneity is less intense (e.g., $\alpha = 1$), due to its limited to local model updates, it cannot well handle more heterogeneous data distribution. The data-free knowledge distillation in FedGen and FedFTG can substantially improve the server’s global model; however, they are insufficient to effectively tackle feature space inconsistency and model drift. The performance gain FedACK is more significant against other baselines when data heterogeneity (e.g., $\alpha = 0.1, 0.05$), indicating its superiority from $\{0.1, 0.05\}$, indicating its superiority.

### Table 1: Comparison of the average maximum accuracy of different methods for social bot detection (%)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vendor-19</th>
<th>TwiBot-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting</td>
<td>$\alpha = 1$</td>
<td>$\alpha = 0.5$</td>
</tr>
<tr>
<td>FedAvg</td>
<td>71.30±0.60</td>
<td>61.06±1.52</td>
</tr>
<tr>
<td>FedProx</td>
<td>84.37±0.43</td>
<td>78.25±0.02</td>
</tr>
<tr>
<td>FedDF</td>
<td>86.37±1.23</td>
<td>80.17±2.21</td>
</tr>
<tr>
<td>FedEnsemble</td>
<td>81.12±2.22</td>
<td>76.70±1.21</td>
</tr>
<tr>
<td>FedDistill</td>
<td>79.68±0.58</td>
<td>68.77±1.13</td>
</tr>
<tr>
<td>FedGen</td>
<td>90.05±0.33</td>
<td>84.83±0.96</td>
</tr>
<tr>
<td>FedFTG</td>
<td>88.31±1.41</td>
<td>82.17±1.52</td>
</tr>
<tr>
<td>FedACK-A</td>
<td>91.31±0.52</td>
<td>84.79±1.05</td>
</tr>
<tr>
<td>FedACK</td>
<td>88.58±1.91</td>
<td>87.05±2.03</td>
</tr>
</tbody>
</table>

Gain | $\uparrow$ 1.26–20.01 | $\uparrow$ 2.22–25.99 | $\uparrow$ 10.03–24.18 | $\uparrow$ 4.48–15.46 | $\uparrow$ 2.82–23.12 | $\uparrow$ 4.13–24.11 | $\uparrow$ 7.67–16.44 | $\uparrow$ 3.97–7.84 |
Table 2: The round number to reach the target accuracy on Vendor-19 (80%, 70%) and TwiBot-20 (70%, 65%).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vendor-19</th>
<th>TwiBot-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting</td>
<td>α = 1 (80)</td>
<td>α = 0.5 (70)</td>
</tr>
<tr>
<td>FedAvg</td>
<td>unreached</td>
<td>unreached</td>
</tr>
<tr>
<td>FedProx</td>
<td>25.3±3.1</td>
<td>32.6±2.3</td>
</tr>
<tr>
<td>FedDF</td>
<td>22.3±2.4</td>
<td>38.4±3.1</td>
</tr>
<tr>
<td>Ensemble</td>
<td>9.0±1.1</td>
<td>6.0±1.4</td>
</tr>
<tr>
<td>FedDistill</td>
<td>60.0±1.0</td>
<td>unreached</td>
</tr>
<tr>
<td>FedGen</td>
<td>7.3±0.4</td>
<td>5.0±0.8</td>
</tr>
<tr>
<td>FedFTG</td>
<td>43.5±37.5</td>
<td>15.6±16.5</td>
</tr>
<tr>
<td>FedACK</td>
<td>4.6±3.8</td>
<td>2.3±0.9</td>
</tr>
</tbody>
</table>

but falls off when higher heterogeneity manifests. This indicates the adversarial training and global knowledge distillation alone can function effectively in the face of low heterogeneity. The contrastive learning mechanism is of importance to constrain the model optimization direction, which demonstrate the necessity of learning consistent feature spaces when dealing with data heterogeneity.

4.3 Efficiency (Q2)

Fig. 4 shows the learning curve of different methods within 100 communication rounds and FedACK is among the top performers. FedDistill has the best stability, rapidly approaching a stable level only after a dozen rounds of communication, but the achievable accuracy is merely lower than 0.65, making it less competitive when compared with other methods. FedACK can very quickly converge to a high level of accuracy after the initial rounds and remains high in the following communication rounds.

Table 2 reports the average number of rounds required for each method to achieve the target accuracy in different settings. Unreached means the failure of achieving the target accuracy (80%, 70% for Vendor-19; 70%, 65% for TwiBot-20) in all three runs with different random seed. FedACK achieves target accuracy with a minimum number of communication rounds under any circumstance. FedGen, the second best performer, still requires 1.6–4.5 times the communication rounds of our method. This is because FedACK incorporates the proportion of the label of each pseudo sample in each client into a part of knowledge during global knowledge extraction and classification. This indicates the importance of each client to the knowledge of a specific sample. FedACK also limits the feature space and optimization direction of the models at the client side, resulting in quicker convergence to target a given accuracy.

4.4 Feature Space Consistency (Q3)

We also conduct an experiment to show how FedACK learns the feature space. Fig. 5 visualizes the learnt feature space and the decision boundaries of two classifiers in FedACK on Vendor-19 dataset. We tweak the feature extractor ε to produce 2-dimension features for each input sample. We randomly select two clients after training FedACK in 100 communication rounds and plot the features of testing data samples. It can be observed that adversarial learning makes the two classifiers in any of the two clients learn distinct decision boundaries. The different decision boundaries impose restrictions on the feature space learned by the feature extractor. To extract features from the same class of samples and locate them in overlapping areas on the same side of the decision boundary, the feature extractor compresses the generated features into a linear region for simultaneous classification. Another observation on Fig. 5(a) and Fig. 5(b) is that the feature extractors learn a consistent feature space across clients due to the contrastive learning that limits the update direction of the feature extractors. These findings show the advancements of FedACK in learning feature spaces.

4.5 Sensitivity (Q4)

We investigate the hyper-parameters sensitivity based on the Vendor-19 dataset. The hyperparameters include γ for adjusting the proportion of adversarial loss, and μ and τ for adjusting the proportion of contrastive loss. The number of repeated random seeds is set as 5.

As shown in Fig. 6(a), when the heterogeneity of data distribution is marginal (i.e., $\alpha = 1$), the accuracy is insensitive to γ. However, when the heterogeneity increases (i.e., $\alpha$ going down), noticeable accuracy variation manifests among different settings of γ. Similar
observations can be found in Fig. 6(c), indicating a discrepancy in the accuracy among different $r$. These findings implicate a great need for carefully examining and fine-tuning such parameters in the adversarial and contrastive loss on a case-by-case basis, considering the data characteristics (particularly the heterogeneity of data distribution), to target the optimal model performance. Fig 6(b) depicts the accuracy under different combinations of $\mu$ and $\alpha$. Observably, for a given data distribution, the model accuracy reaches its peak when $\mu$ increases to 0.5, before falling off if $\mu$ continues to increase. This is largely due to a balance between adversarial loss and contrastive loss. The dominance of either side will lead to a degradation of the model’s effectiveness.

4.6 Cross-Lingual Validation (Q5)

As there is no publicly available datasets in the field of social bot detection designated for cross-lingual performance at the time of writing, we synthesize an experimental cross-lingual dataset by randomly selecting half the social accounts from Vendor-19 dataset and translating their tweets into Chinese using Google Translate. We use the same experiment settings described in Section 4.1. We pre-train the cross-lingual mapper by using the cross-lingual summarization corpus NCLS [48] and then combine the pre-trained encoder $\phi_E$ and mapper $M$ with FedACK and other baselines. By default, each method is evaluated with the cross-lingual mapper $M$ enabled. Each method with the suffix -NC means the model only uses $\phi_E$ to extract features from text content without $M$.

Experimental results are shown in Table 3. The observations are three-fold: i) FedACK constantly outperforms others in all circumstances. The accuracy can increase 3.28~6.45 in absolute value, when data heterogeneity is low ($\alpha = 1$). ii) Compared with the model variant without $M$ (e.g. FedACK-NC), the accuracy of the model (e.g. FedACK) equipped with the cross-lingual mapper is unsurprisingly improved. This once again demonstrates the substantial capability and necessity of coping with cross-lingual issues. It is worth noting that the data heterogeneity of data distribution becomes the dominating challenge to accuracy, and the cross-lingual module itself is insufficient and thus brings limited benefit. iii) Equipped with the cross-lingual mapper, each method can well tackle the multi-lingual scenarios. There is marginal disparity between the model accuracy in the case of a singular language environment (shown in Table 1) and the accuracy when multiple languages co-exist in the tweet features (as shown in Table 3). This demonstrates the robustness and effectiveness of the proposed mapper for cross-lingual contents. Among all the comparative baselines, our method has the least degradation caused by the existence of multiple languages, owing to the unified feature space mapping for different languages.

5 CONCLUSION

Social bots have been growing in the social media platforms for years. State-of-the-art bot detection methods fail to incorporate individual platforms with different models and data characteristics. In this paper, we devise a GAN-based federated knowledge distillation mechanism for efficiently transferring knowledge of data distribution among clients. We elaborate a contrast and adversarial learning mechanism for efficiently transferring knowledge of data distribution, to target the optimal model performance. Fig 6(b) depicts the accuracy under different combinations of $\mu$ and $\alpha$. Observably, for a given data distribution, the model accuracy reaches its peak when $\mu$ increases to 0.5, before falling off if $\mu$ continues to increase. This is largely due to a balance between adversarial loss and contrastive loss. The dominance of either side will lead to a degradation of the model’s effectiveness.

5 CONCLUSION

Social bots have been growing in the social media platforms for years. State-of-the-art bot detection methods fail to incorporate individual platforms with different models and data characteristics. In this paper, we devise a GAN-based federated knowledge distillation mechanism for efficiently transferring knowledge of data distribution among clients. We elaborate a contrast and adversarial learning mechanism for efficiently transferring knowledge of data distribution among clients. Experiments show that FedACK outperforms baseline methods in terms of accuracy, learning efficiency, and feature space consistency. In the future, we will theoretically investigate the feature space consistency and extend FedACK to fit graph-based datasets with diversified entities and relations.

ACKNOWLEDGMENTS

This paper was supported by the National Key R&D Program Program of China (2021YFC3300502). Hao Peng was also supported by
A APPENDIX

A.1 Glossary of Notations
In Table 4, we summarize the main notations used in this work.

A.2 Statistics of Datasets
Table 5 summarizes the basic statistic information of datasets used in this work.

A.3 Generalization Experiment
To validate the generalization of our method, we collected four additional public social bot detection datasets: Varol-17, Gilani-17, cresci-19, botometer-feedback-19. To simulate the scenario of bot detection in uniting multiple social platforms, we select one dataset from Vendor-19 and TwiBot-20 as the test dataset. Each of the remaining datasets is distributed to a specific client. In this circumstance, there are five clients and a server, and each client represents a social platform. We aim to demonstrate the enhanced model proposed in this work can still competitively detect new variants that have never been seen before, when all platforms share the characteristics of their own social bots with other platforms. As shown in Table 6, the difficulty detection in the two datasets is entirely different. For most of the approaches, the classification accuracy of TwiBot-20 dataset is merely 50%, which indicates a huge discrepancy of bot features among datasets. In other words, the detection models learnt from the early generations of bots can hardly detect the bots in TwiBot-20 in an accurate manner. By contrast, our approach can achieve a higher accuracy due to the ability to learn a consistent feature space; as a result, the bot account features in different clients can be effectively shared. Additionally, constraining the optimization direction of the client model can facilitate our model to obtain better representation ability, and hence a better generalization than others.

Table 4: Notations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{L}$</td>
<td>Defined loss function</td>
</tr>
<tr>
<td>$K, N_k$</td>
<td>Number of clients; Number of samples in $k$-th client</td>
</tr>
<tr>
<td>$x_m, y_n$</td>
<td>The $m$-th word and $n$-th word in two different language contents</td>
</tr>
<tr>
<td>$\phi_E$</td>
<td>The encoder for text content representation</td>
</tr>
<tr>
<td>$\phi_D$</td>
<td>The decoder for translating context representation</td>
</tr>
<tr>
<td>$M$</td>
<td>The cross-lingual mapper for converting context representation</td>
</tr>
<tr>
<td>$z_k$</td>
<td>Text context representation vector</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>The backbone model for user feature extraction</td>
</tr>
<tr>
<td>$u_t$</td>
<td>The tweets set from $t$-th user</td>
</tr>
<tr>
<td>$t^m_i$</td>
<td>The $i$-th word in user’s $m$-th tweet</td>
</tr>
<tr>
<td>$h_m^t$</td>
<td>Representation for $m$-th tweet posted by $t$-th user</td>
</tr>
<tr>
<td>$r_p, r_t$</td>
<td>User’s property representation; tweet-level representation</td>
</tr>
<tr>
<td>$r_u, r_t$</td>
<td>User-level representation</td>
</tr>
<tr>
<td>$G, D_1, D_2$</td>
<td>Generator; Discriminators</td>
</tr>
<tr>
<td>$\mathcal{D}_k$</td>
<td>The local data stored in the $k$-th client</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>The data set collected from all clients</td>
</tr>
<tr>
<td>$(x^k_i, y^k_i)$</td>
<td>The $i$-th sample pair stored in the $k$-th client</td>
</tr>
<tr>
<td>$N(0, 1)$</td>
<td>The gaussian distribution</td>
</tr>
<tr>
<td>$\tilde{z}$</td>
<td>The pseudo-data generated by generators</td>
</tr>
<tr>
<td>$p, \tilde{p}$</td>
<td>Probability of local data; Global data distribution probability</td>
</tr>
<tr>
<td>$D_{KL}$</td>
<td>The Kullback–Leibler divergence</td>
</tr>
<tr>
<td>$\tau$</td>
<td>The temperature parameter for smoothing similarity value</td>
</tr>
<tr>
<td>$\gamma, \mu$</td>
<td>The weight hyperparameter in the defined loss function</td>
</tr>
<tr>
<td>$\theta$</td>
<td>The parameters of the designed model</td>
</tr>
<tr>
<td>$M$</td>
<td>The number of clients participating in the communication</td>
</tr>
<tr>
<td>$k^y_j$</td>
<td>Ratio of samples with label $y$ stored in client $j$ against in $\mathcal{D}$</td>
</tr>
</tbody>
</table>

Table 5: Statistics of datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Humans</th>
<th>Bots</th>
<th>Total number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor-19 [40]</td>
<td>1860</td>
<td>568</td>
<td>2428</td>
</tr>
<tr>
<td>TwiBot-20 [16]</td>
<td>4175</td>
<td>5286</td>
<td>9461</td>
</tr>
</tbody>
</table>

Table 6: Comparison of the average maximum accuracy of different methods which are tested on a specific dataset (Vendor-19 or TwiBot-20) and trained from the other datasets.

<table>
<thead>
<tr>
<th>Test Dataset</th>
<th>Vendor-19</th>
<th>TwiBot-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg</td>
<td>76.53±0.11</td>
<td>49.63±1.09</td>
</tr>
<tr>
<td>FedProx</td>
<td>74.26±3.57</td>
<td>54.21±2.01</td>
</tr>
<tr>
<td>Ensemble</td>
<td>76.01±0.65</td>
<td>51.98±5.50</td>
</tr>
<tr>
<td>FedDistill</td>
<td>73.11±0.95</td>
<td>51.19±1.12</td>
</tr>
<tr>
<td>FedGen</td>
<td>77.52±0.65</td>
<td>62.30±0.93</td>
</tr>
<tr>
<td>FedFTG</td>
<td>76.51±1.15</td>
<td>60.12±1.01</td>
</tr>
<tr>
<td>FEDACK</td>
<td>78.13±1.38</td>
<td>64.79±1.61</td>
</tr>
</tbody>
</table>