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@cetc.com.cn

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

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

Pengyuan Zhou*
USTC
Heifei, 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

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583500

ACM Reference Format:
Yingguang Yang, Renyu Yang, Hao Peng, Yangyang Li, Tong Li, Yong Liao, and Pengyuan Zhou. 2023. FedACK: Federated Adversarial Contrastive Knowledge Distillation for Cross-Lingual and Cross-Model Social Bot Detection. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. 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 becomes 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-model 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,

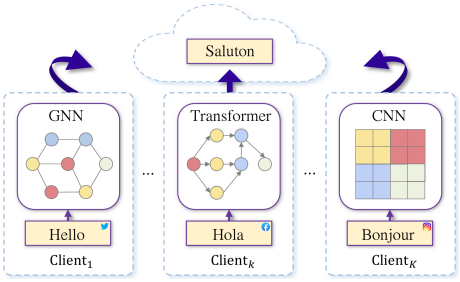


Figure 1: Incorporating multiple social platforms with heterogeneous languages, context spaces and model architectures

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 direct 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.

Federated 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 siloed way [8]. FedACK can fill this gap through enhanced adversarial learning with a shared discriminator and an exclusive discriminator to support designated cross-model bot detection.

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 $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$. 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:

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

where $\mathcal{L}$ is the loss function that evaluates the prediction model w on the data sample (x_i^k, y_i^k) of $\mathcal{D}_k = \{(x_i^k, y_i^k)\}|_{i=1}^{N_k}$ 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 $x = \{x_1, ..., x_m\}$ in one language and the corresponding n-word text $y = \{y_1, ..., y_n\}$ in another language, we use an **Encoder** ϕ_E to transform the source text x and the target text y into context representations $z_x = \{z_{x_1}, ..., z_{x_m}\}$ and $z_y = \{z_{y_1}, ..., z_{y_n}\}$. We devise a **Mapper** $\mathcal{M}$ for converting between two context representation spaces, i.e., $z_x{}' = \mathcal{M}(z_y)$ and $z_y{}' = \mathcal{M}(z_x)$.

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

$$\bar{z}_x = \frac{1}{m}\sum_{k=1}^{m} z_{x_k}, \bar{z}'_x = \frac{1}{n}\sum_{k=1}^{n} z'_{x_k}. \tag{2}$$

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

$$\begin{aligned} \mathcal{L}_{dis} &= \mathcal{L}_{dis_x} + \mathcal{L}_{dis_y}, \\ \mathcal{L}_{dis_x} &= (D(\bar{z}_x) - y_{\bar{z}_x})^2 + (D(\bar{z}'_x) - y_{\bar{z}'_x})^2, \end{aligned} \tag{3}$$

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

$$\mathcal{L}_{gen} = (D(\bar{z}'_x) - y_{\bar{z}'_x})^2 + (D(\bar{z}'_y) - y_{\bar{z}'_y})^2, \tag{4}$$

where the label $y_{\bar{z}}$ 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 ϕ_D and generate the corresponding translation $(\tilde{z}_y, \tilde{z}_x, \tilde{z}'_y, \tilde{z}'_x)$, 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:

$$\begin{aligned} \mathcal{L}_{trans} &= \mathcal{L}_{z_x} + \mathcal{L}_{z_y} + \mathcal{L}_{z_x{}'} + \mathcal{L}_{z_y{}'}, \\ \mathcal{L}_{z_x} &= -\sum_{t=1}^{n}\log P(y_t|\tilde{z}_y < t, z_x), \\ \mathcal{L}_{z_x{}'} &= -\sum_{t=1}^{n}\log P(y_t|\tilde{z}'_y < t, z_x{}'). \end{aligned} \tag{5}$$

3.2 Backbone Model for Feature Extraction

The main task is to let user-level backbone feature extractor model ε 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 $u_t = \{t_1^1, \ldots, t_{Q_1}^1, t_1^2, \ldots, t_{Q_M}^M\}$, possibly in the form of different languages; t_i^j 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., $z_{u_t} = \{z_1^1, \ldots, z_{Q_1}^1, z_1^{2\prime}, \ldots, z_{Q_M}^M\}$. A Convolutional Neural Networks (CNN) layer, i.e., TextCNN[21], is then used to obtain the tweet-level representation h_{u_t}:

$$h_{u_t} = \{h_1^t, \ldots, h_M^t\}, h_j^t = CNN(\{z_1^j, \ldots, z_{Q_1}^j\}). \tag{6}$$

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:

$$r_u = MLP(concat(r_p, r_t)). \tag{7}$$

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 ε to get the user's representation $r_u = \varepsilon(x_u)$ from local data $\mathcal{D}_k$. 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

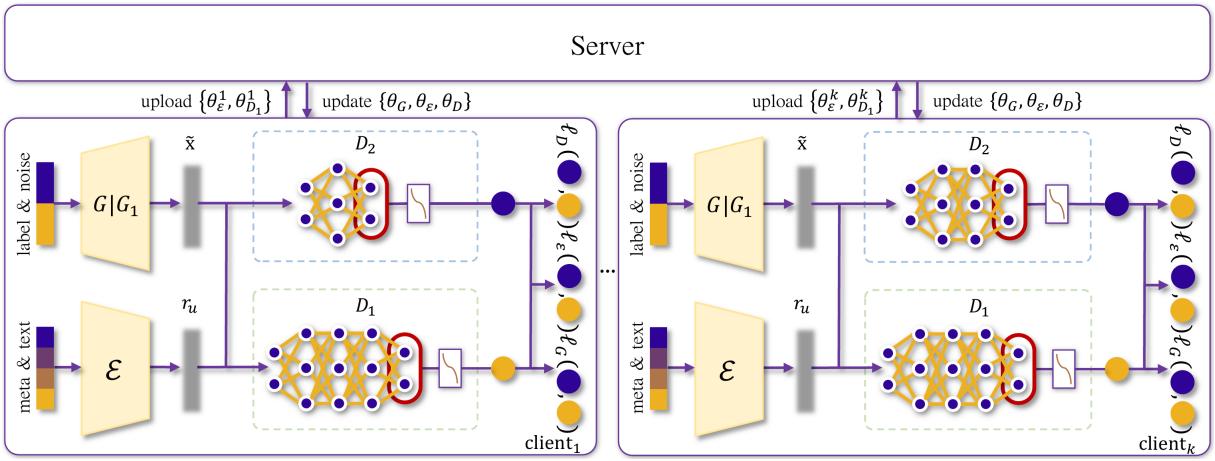


Figure 2: The proposed FedACK framework.

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 teacher network and distill the knowledge of global data distribution from it. For each sample (x_i^k, y_i^k), G uses a standard Gaussian noise $z \sim \mathcal{N}(0, 1)$ and label y_i^k to generate pseudo-data $\tilde{x} = G(z, y_i^k; \theta_G)$. $(x_i^k, \tilde{x})$ is fed into D_1 to obtain the probability distributions $(p, \tilde{p})$. We make D_1 fit probability p of real data x_i^k close to $\tilde{p}$ to distill knowledge by minimizing Eq. (8):

$$\mathcal{L}_{dis}^k = \frac{1}{N_k} \sum_{i=1}^{N_k} D_{KL}(\sigma(D_1(r_i^k)) || \sigma(D_1(\tilde{x})), \tag{8}$$

where σ is the softmax function, D_{KL} is the Kullback–Leibler divergence and $r_i^k = \varepsilon(x_i^k)$. Similar $\mathcal{L}_{dis}^{k\prime}$ is performed for D_2.

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

$$\mathcal{L}_{adv}^k = \frac{1}{N_k} \sum_{i=1}^{N_k} D_{KL}(\sigma(D_1(r_i^k)) || \sigma(D_2(r_i^k))). \tag{9}$$

In essence, maximizing Eq. (9) can produce distinct decision boundaries between two discriminators to compress overlapping decision space. Intuitively, this is because ε would learn more precise feature space if the representations could be correctly classified by both discriminators. Similarly to Eq. (9), we calculate adversarial loss $\mathcal{L}_{advg}^k$ 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:

$$\mathcal{L}_D^k = \mathcal{L}_{cls}^k + \alpha(\mathcal{L}_{dis}^k + \mathcal{L}_{dis}^{k\prime}) + \gamma(\mathcal{L}_{advg}^k - \mathcal{L}_{adv}^k). \tag{10}$$

Stage-2: Training ε. 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 ε_t^k to produce a representation $r = \varepsilon_t^k(x_i^k)$ as close as possible to the $r_{glo} = \varepsilon_t(x_i^k)$ generated by the global extractor ε while as far away as possible from the last round result of the feature extractor $r_{pre} = \varepsilon_{t-1}(x_i^k)$. We define the following contrastive loss function:

$$\mathcal{L}_{con}^{k,i} = -\log \frac{\exp(\text{sim}(r, r_{glo})/\tau)}{\exp(\text{sim}(r, r_{glo})/\tau) + \exp(\text{sim}(r, r_{pre})/\tau)}, \tag{11}$$

where *sim* is the similarity measure function and τ 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:

$$\mathcal{L}_\varepsilon^k = \mathcal{L}_{cls}^k + \gamma \mathcal{L}_{adv}^k + \mu \frac{1}{N_k} \sum_{i=1}^{N_k} \mathcal{L}_{con}^{k,i}. \tag{12}$$

Stage-3: Training G_k. We maximize $\mathcal{L}_{advg}^k$ 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 ε. As G_k only generates the same data for each class, we add the diversity loss $\mathcal{L}_{var}$ to improve the diversity of the generated data and prevent model collapse:

$$\mathcal{L}_{var} = e^{\frac{1}{N*N} \sum\limits_{i,j \in \{1,\cdots,N\}} (-\|\tilde{x}_i - \tilde{x}_j\|_2 * \|z_i - z_j\|_2)}, \tag{13}$$

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

$$\mathcal{L}_g^k = \mathcal{L}_{cls}^k - \mathcal{L}_{advg}^k + \mathcal{L}_{var}. \tag{14}$$

3.3.2 Server Aggregation Knowledge Extraction. In each communication round, the client k uploads the local parameters of $\{\theta_{\varepsilon}^{k}, \theta_{D_1}^{k}\}$ to the server once the local training is finished, then waits for the updated global G, D and ε 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 ε and D:

$$\theta_{\varepsilon} = \sum_{k=1}^{M} \frac{\| \mathcal{D}_k \|}{\| \mathcal{D} \|} \theta_{\varepsilon}^{k}, \theta_D = \sum_{k=1}^{M} \frac{\| \mathcal{D}_k \|}{\| \mathcal{D} \|} \theta_{D_1}^{k}, \tag{15}$$

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. client's D_1 is used as the teacher network and define the loss of G as:

$$\mathcal{L}_G = \frac{1}{K} \sum_{k=1}^{K} \sum_{\tilde{x}} \alpha_t^{k,y} [D_{KL} \left(\sigma(D_1^k(\tilde{x})) || \sigma(D(\tilde{x})) \right) + \mathcal{L}_{cls}^{\tilde{x}}], \tag{16}$$

where $\tilde{x}$ is from empirical samples $\mathcal{D}_G$ generated by G using noise $z \sim \mathcal{N}(0,1)$ and label $y \sim p(y)$. $\alpha_t^{k,y}$ is the ratio of samples with label y stored in client k against the same label samples in $\mathcal{D}$. $p(y)$ is obtained by label counts from clients through communication.

3.4 Pipeline of FedACK

Alg. 1 summarizes the overall pipeline of FedACK. The cross-lingual model ϕ_E and $\mathcal{M}$ for the backbone model ε 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, ε and D to a selected subset of clients S_t (Line 5). Each client optimizes all the required models D_1^k, D_2^k, ε 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 $\theta_{\varepsilon}, \theta_D$ and to optimize the global generator G (Lines 20-21).

Algorithm 1: FedACK

Input: Local data $\mathcal{D}_k$, $k = 1, \cdots, K$, corpus data $\mathcal{D}_c$, models G_k, ϕ_E, $\mathcal{M}$, ε, D_1^k, D_2^k, G, D
Output: Local model D_2, global models ϕ_E, $\mathcal{M}$, ε, G

1 initialization;
2 Server trains ϕ_E, $\mathcal{M}$ via Eq.(5) based on $\mathcal{D}_c$, and transfers θ_{ϕ_E}, $\theta_{\mathcal{M}}$ to clients;
3 **for** each communication round $t = 1, \ldots, T$ **do**
4 $S_t \leftarrow$ random subset (C fraction) of the K clients;
5 Server broadcasts $\{\theta_G, \theta_{\varepsilon}, \theta_D\}$ to S_t ;
6 **for** each client $k \in S_t$ **in parallel do**
7 Client k updates $\{\theta_G, \theta_{\varepsilon}^k, \theta_{D_1}^k\}$;
8 **for** each local epoch $e = 1, \cdots, E$ **do**
9 calculate $\mathcal{L}_D^k$ via Eq.(10) ;
10 $\{\theta_{D_1}^k, \theta_{D_2}^k\} \leftarrow \{\theta_{D_1}^k, \theta_{D_2}^k\} - \nabla \mathcal{L}_D^k$;
11 **end**
12 **for** each local epoch $e = 1, \cdots, E$ **do**
13 calculate $\mathcal{L}_{\varepsilon}^k$ via Eq.(12), $\theta_{\varepsilon}^k \leftarrow \theta_{\varepsilon}^k - \nabla \mathcal{L}_{\varepsilon}^k$;
14 **end**
15 **for** each local epoch $e = 1, \cdots, E$ **do**
16 calculate $\mathcal{L}_g^k$ via Eq.(14), $\theta_{G_k}^k \leftarrow \theta_{G_k}^k - \nabla \mathcal{L}_g^k$;
17 **end**
18 Client k sends $\{\theta_{\varepsilon}^k, \theta_{D_1}^k\}$ back to server;
19 **end**
20 Server update $\{\theta_{\varepsilon}, \theta_D\} \leftarrow$ Eq.(15) ;
21 Server calculate $\mathcal{L}_G$ for G via Eq.(16) ;
22 $\theta_G \leftarrow \theta_G - \nabla \mathcal{L}_G$
23 **end**

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

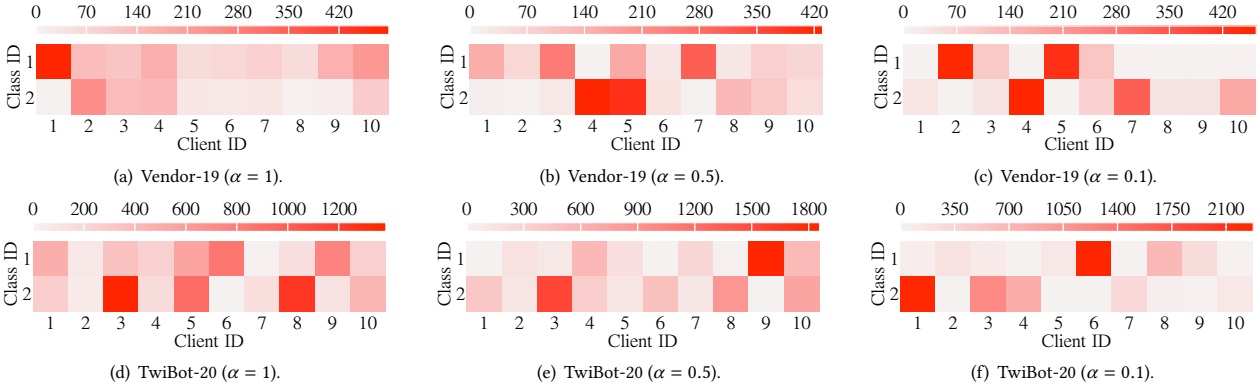
4.1.1 Software and Hardware. FedACK is implemented with Python 3.8.10, Pytorch 1.7.1 and runs on two servers, one is equipped with NVIDIA Tesla V100 GPU, 2.20GHz Intel Xeon Gold 5220 CPU and 512GB RAM, and the other is equipped with NVIDIA GeForce RTX 3090 GPU, 3.40GHz Intel Xeon Gold 6246 CPU and 256GB 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 (α) to mock the non-IID given in [23] and split the bot dataset with heterogeneity. α is an indicator of Dirichlet distribution – the smaller α 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.

4.1.5 Model Parameters. The cross-lingual module is trained upon the corpus released in [48]. To be fair, the same cross-lingual module and backbone model ε (same architecture and initial parameters)



(a) Vendor-19 ($\alpha = 1$). (b) Vendor-19 ($\alpha = 0.5$). (c) Vendor-19 ($\alpha = 0.1$).

(d) TwiBot-20 ($\alpha = 1$). (e) TwiBot-20 ($\alpha = 0.5$). (f) TwiBot-20 ($\alpha = 0.1$).

Figure 3: Visualization of data heterogeneity. The darker color means more training samples with a label available to the client.

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

Dataset	Vendor-19				TwiBot-20			
Setting	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.05$
FedAvg	71.30±0.60	61.06±1.52	60.88±2.85	59.81±2.48	54.04±0.50	55.41±1.35	51.37±0.77	52.46±0.02
FedProx	84.37±0.43	78.25±1.02	51.86±0.04	63.27±2.32	74.34±0.06	73.32±0.25	51.86±0.04	52.30±0.63
FedDF	86.37±1.23	80.17±2.21	63.16±1.37	67.01±1.78	72.12±1.96	71.25±1.03	55.23±1.32	53.35±1.41
FedEnsemble	81.12±2.22	76.70±1.21	64.51±2.56	68.05±1.15	55.98±2.55	54.15±0.04	54.21±0.04	54.15±0.04
FedDistill	79.68±0.58	68.77±1.13	52.88±0.06	70.25±0.39	64.11±0.29	63.34±0.56	50.00±0.00	54.30±0.05
FedGen	90.05±0.33	84.83±0.96	65.12±0.60	70.79±2.39	74.14±0.47	73.12±2.09	59.19±2.70	55.78±1.79
FedFTG	88.31±1.41	82.17±1.52	66.01±1.25	68.39±1.94	74.27±1.21	74.13±0.53	60.14 ±1.74	56.17±1.27
FedACK-A	**91.31±0.52**	84.79±1.05	66.10±2.90	68.21±1.95	**77.16±1.09**	74.70±1.64	63.52±1.09	55.39±1.24
FedACK	88.58±1.91	**87.05±2.03**	**76.04±3.40**	**75.27±2.50**	77.08±1.83	**78.26±2.60**	**67.81±2.20**	**60.14±1.32**
Gain	↑ **1.26~20.01**	↑ **2.22~25.99**	↑ **10.03~24.18**	↑ **4.48~15.46**	↑ **2.82~23.12**	↑ **4.13~24.11**	↑ **7.67~16.44**	↑ **3.97~7.84**

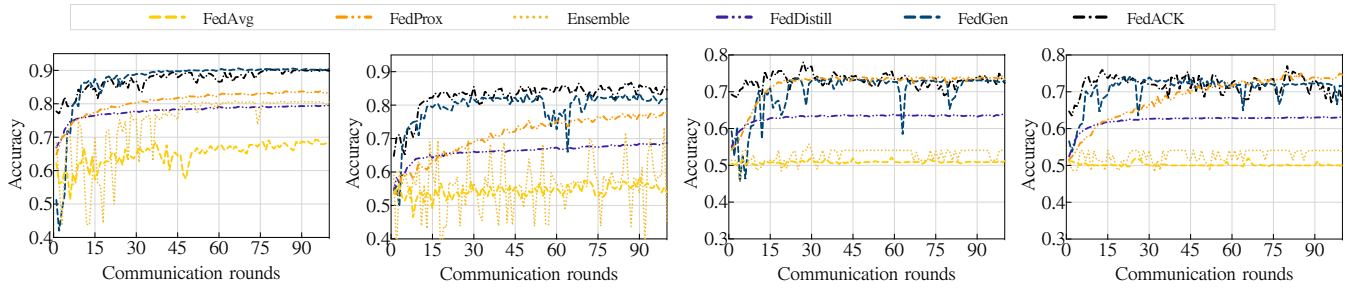
are used upon all baseline approaches. For the transformer architectures in the cross-lingual module, we use the same configuration of [38]; the number of layers, feed-forward hidden size, model hidden size and the number of heads are 6, 1024, 512, and 8, respectively. The mapper is an MLP with three linear layers with a hidden dimension of 512, and the discriminator is an MLP with four linear layers with a hidden dimension of 512. The MLP used for extracting property features in the backbone model has two linear layers with a hidden dimension of 512. TextCNN [21] used in ε has four convolution kernels of size [2, 3, 4, 5]. The generators in FedACK are MLP with two linear layers with a hidden dimension of 256. The discriminators in FedACK are MLP with 3 linear layers with a hidden dimension of 256. Common parameters for training the models include: batch size (64), learning rate (0.01), optimizer (Adam), global communication rounds (100), and local updating steps (5).

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 α 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., $\alpha = \{1, 0.5\}$) due to its limit 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 increases (e.g., $\alpha = \{0.1, 0.05\}$), indicating its superiority in handling data heterogeneity. Appendix A.3 further demonstrates the improved model generalization in our approach when compared with other baselines.



(a) Vendor-19 ($\alpha = 1$). (b) Vendor-19 ($\alpha = 0.5$). (c) TwiBot-20 ($\alpha = 1$). (d) TwiBot-20 ($\alpha = 0.5$).

Figure 4: Learning Curve of (a-b) Vendor-19 and (c-d) TwiBot-20 in 100 communication rounds in different α settings.

Table 2: The round number to reach the target accuracy on Vendor-19 (80%, 70%) and TwiBot-20 (70%, 65%).

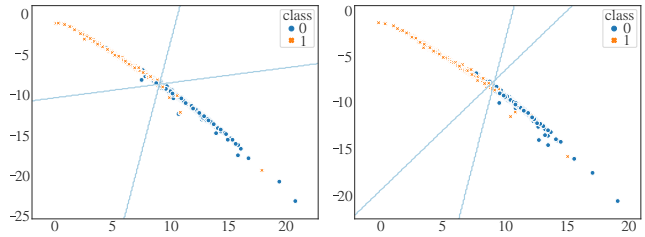
Dataset	Vendor-19		TwiBot-20	
Setting	$\alpha = 1$ (80)	$\alpha = 0.5$ (70)	$\alpha = 1$ (70)	$\alpha = 0.5$ (65)
FedAvg	unreached	unreached	unreached	unreached
FedProx	25.3±3.1	32.6±2.3	13.3±2.8	24.0±7.3
FedDF	22.3±2.4	38.4±3.1	50.3±5.2	60.2±6.4
Ensemble	9.0±1.1	6.0±1.4	unreached	unreached
FedDistill	60.0±1.0	unreached	unreached	unreached
FedGen	7.3±0.4	5.0±0.8	10.6±0.9	4.6±1.2
FedFTG	43.5±37.5	15.6±16.5	12.6±0.5	9.4±2.3
FedACK	**4.6±3.8**	**2.3±0.9**	**2.33±1.25**	**1.67±0.94**

Ablation Study. We generated a new variant model, FedACK-A, that excludes the contrastive module. As shown in Table 1, the accuracy of FedACK-A is outstanding when data heterogeneity is low 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.



(a) Visualization of client 1. (b) Visualization of client 2.

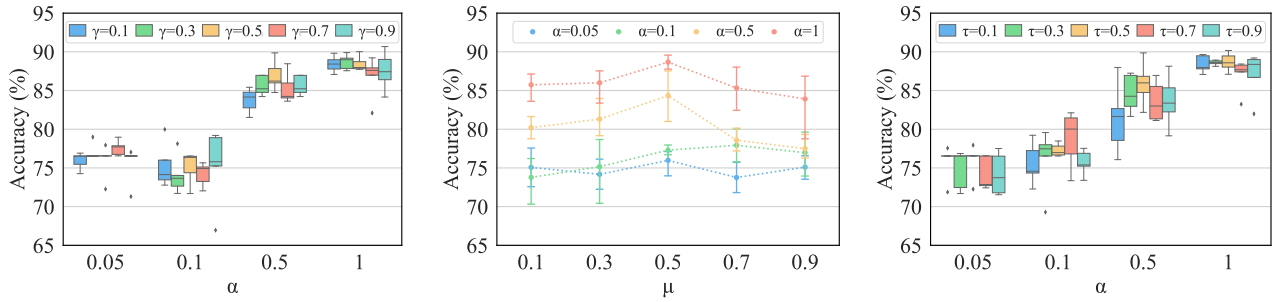
Figure 5: Decision boundaries and feature space of two randomly selected clients from FedACK trained on Vendor-19. The x-axis and y-axis represent the values of the 2-dimensional features output by ε described in Sec 4.4.

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



(a) FedACK with different γ in adversarial learning. (b) FedACK with different μ in contrastive learning. (c) FedACK with different τ in contrastive learning.

Figure 6: Hyperparameter Sensitivity (γ, μ, τ) of FedACK (a-c) on Vendor-19 under different data heterogeneity settings (α).

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., α 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 τ. 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 μ and α. Observably, for a given data distribution, the model accuracy reaches its peak when μ increases to 0.5, before falling off if μ 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 ϕ_E and mapper $\mathcal{M}$ with FedACK and other baselines. By default, each method is evaluated with the cross-lingual mapper $\mathcal{M}$ enabled. Each method with the suffix *-NC* means the model only uses ϕ_E to extract features from text content without $\mathcal{M}$.

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 $\mathcal{M}$ (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 there is also a diminishing performance gain from cross-language mapping as the α value gets smaller. This is because the 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.

Table 3: Comparison of the average maximum accuracy (%) of different methods with/without(-NC) cross-lingual mapping. Gain is the disparity between FedACK and other baselines.

Dataset	Vendor-19			
Setting	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.05$
FedDistill-NC	78.14±1.30	67.01±0.16	66.35±0.25	69.24±0.30
FedDistill	80.03±0.59	68.87±0.30	67.20±0.22	70.18±0.45
FedGen-NC	81.94±0.82	77.88±0.52	70.65±2.73	72.73±2.20
FedGen	83.20±1.07	79.63±0.57	72.42±2.18	73.26±0.53
FedFTG-NC	78.18±1.04	76.79±1.22	67.61±1.57	72.30±1.20
FedFTG	82.85±1.52	77.29±3.42	69.65±0.95	73.15±1.27
FedACK-NC	84.54±1.30	79.37±1.24	73.99±2.55	**74.48± 1.50**
FedACK	**86.48±0.99**	**81.16±1.06**	**75.05±2.78**	74.18±1.67
Gain	**↑ 3.28~6.45**	**↑ 1.53~12.29**	**↑ 2.63~7.85**	**↑ 0.92~4.0**

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 method to ensure consistent feature space for better knowledge transfer and representation when tackling non-IID data and data scarcity 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 the National Key R&D Program of China (2021YFB1714800), NSFC through grant U20B2053, and S&T Program of Hebei (21340301D).

REFERENCES

[1] Norah Abokhodair, Daisy Yoo, and David W McDonald. 2015. Dissecting a social botnet: Growth, content and influence in Twitter. In *CSCW*. 839–851.
[2] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. 2021. Federated Learning Based on Dynamic Regularization. In *ICLR*. OpenReview.net.
[3] Jonathon M Berger and Jonathon Morgan. 2015. The ISIS Twitter Census: Defining and describing the population of ISIS supporters on Twitter. (2015).
[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *CCS*. 1175–1191.
[5] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *SIGKDD*. 535–541.
[6] Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834* (2020).
[7] Samuel Kai Wah Chu, Runbin Xie, and Yanshu Wang. 2021. Cross-Language fake news detection. *DIM* 5, 1 (2021), 100–109.
[8] Stefano Cresci. 2020. A decade of social bot detection. *Commun. ACM* 63, 10 (2020), 72–83.
[9] Eleonora D'Andrea, Pietro Ducange, Beatrice Lazzerini, and Francesco Marcelloni. 2015. Real-time detection of traffic from twitter stream analysis. *T-ITS* 16, 4 (2015), 2269–2283.
[10] Arkadipta De, Dibyanayan Bandyopadhyay, Baban Gain, and Asif Ekbal. 2022. A Transformer-Based Approach to Multilingual Fake News Detection in Low-Resource Languages. *ACM Trans. Asian Low Resour. Lang. Inf. Process.* 21, 1 (2022), 9:1–9:20.
[11] Ashok Deb, Luca Luceri, Adam Badaway, and Emilio Ferrara. 2019. Perils and challenges of social media and election manipulation analysis: The 2018 us midterms. In *WWW*. 237–247.
[12] Daryna Dementieva and Alexander Panchenko. 2021. Cross-lingual Evidence Improves Monolingual Fake News Detection. In *ACL(Workshop)*. 310–320.
[13] Jiangshu Du, Yingtong Dou, Congying Xia, Limeng Cui, Jing Ma, and S Yu Philip. 2021. Cross-lingual covid-19 fake news detection. In *ICDMW (Workshops)*. IEEE, 859–862.
[14] Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo. 2022. Heterogeneity-aware Twitter Bot Detection with Relational Graph Transformers. In *AAAI*.
[15] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. Satar: A self-supervised approach to twitter account representation learning and its application in bot detection. In *CIKM*. 3808–3817.
[16] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. Twibot-20: A comprehensive twitter bot detection benchmark. In *CIKM*. 4485–4494.
[17] Emilio Ferrara, Herbert Chang, Emily Chen, Goran Muric, and Jaimin Patel. 2020. Characterizing social media manipulation in the 2020 US presidential election. *First Monday* (2020).
[18] Emilio Ferrara, Wen-Qiang Wang, Onur Varol, Alessandro Flammini, and Aram Galstyan. 2016. Predicting online extremism, content adopters, and interaction reciprocity. In *ICSI*. Springer, 22–39.
[19] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv* 2, 7 (2015).
[20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*. PMLR, 5132–5143.
[21] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. ACL, 1746–1751.
[22] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *CoRR* abs/1610.02527 (2016).
[23] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2021. Federated learning on non-iid data silos: An experimental study. *arXiv* (2021).
[24] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *CVPR*. 10713–10722.
[25] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *PMLS* 2 (2020), 429–450.
[26] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *NIPS* 33 (2020), 2351–2363.
[27] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S Yu. 2022. Federated social recommendation with graph neural network. *TIST* 13, 4 (2022), 1–24.
[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. PMLR, 1273–1282.
[29] Debora Nozza. 2021. Exposing the limits of Zero-shot Cross-lingual Hate Speech Detection. In *ACL/IJCNLP*. ACL, 907–914.
[30] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially private federated knowledge graphs embedding. In *CIKM*. 1416–1425.
[31] Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, and Philip Yu. 2022. Reinforced, incremental and cross-lingual event detection from social messages. *TPAMI* (2022).
[32] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. 2020. Fedgan: Federated generative adversarial networks for distributed data. *arXiv* (2020).
[33] Hyowoon Seo, Jihong Park, Seungeun Oh, Mehdi Bennis, and Seong-Lyun Kim. 2020. Federated knowledge distillation. *arXiv* (2020).
[34] Naichen Shi, Fan Lai, Raed Al Kontar, and Mosharaf Chowdhury. 2021. Fed-ensemble: Improving generalization through model ensembling in federated learning. *arXiv* (2021).
[35] Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. *NIPS* 33 (2020), 22045–22055.
[36] Kenneth Steimel, Daniel Dakota, Yue Chen, and Sandra Kübler. 2019. Investigating multilingual abusive language detection: A cautionary tale. In *RANLP*. 1151–1160.
[37] Onur Varol, Emilio Ferrara, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2017. Online human-bot interactions: Detection, estimation, and characterization. In *ICWSM*, Vol. 11. 280–289.
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS* 30 (2017).
[39] Feng Wei and Uyen Trang Nguyen. 2019. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In *TPS-ISA*. IEEE, 101–109.
[40] Kai-Cheng Yang, Onur Varol, Clayton A Davis, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2019. Arming the public with artificial intelligence to counter social bots. *Comput. Hum. Behav.* 1, 1 (2019), 48–61.
[41] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. 2020. Scalable and generalizable social bot detection through data selection. In *AAAI*, Vol. 34. 1096–1103.
[42] Yingguang Yang, Renyu Yang, Yangyang Li, Kai Cui, Zhiqin Yang, Yue Wang, Jie Xu, and Haiyong Xie. 2022. RoSGAS: Adaptive Social Bot Detection with Reinforced Self-Supervised GNN Architecture Search. *ACM Transactions on the Web* (2022).
[43] Sarita Yardi, Daniel Romero, Grant Schoenebeck, et al. 2010. Detecting spam in a twitter network. *First monday* (2010).
[44] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. 2022. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *CVPR*. 10174–10183.
[45] Zhenyuan Zhang. 2022. FedDTG: Federated Data-Free Knowledge Distillation via Three-Player Generative Adversarial Networks. *arXiv* (2022).
[46] Jun Zhao, Xudong Liu, Qiben Yan, Bo Li, Minglai Shao, and Hao Peng. 2020. Multi-attributed heterogeneous graph convolutional network for bot detection. *IS* 537 (2020), 380–393.
[47] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv* (2018).
[48] Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang, and Chengqing Zong. 2019. NCLS: Neural Cross-Lingual Summarization. In *EMNLP/IJCNLP*. ACL, 3052–3062.
[49] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *ICML*. PMLR, 12878–12889.
[50] Haris Bin Zia, Ignacio Castro, Arkaitz Zubiaga, and Gareth Tyson. 2022. Improving Zero-Shot Cross-Lingual Hate Speech Detection with Pseudo-Label Fine-Tuning of Transformer Language Models. In *ICWSM*, Vol. 16. 1435–1439.

Table 4: Notations.

Symbol	Definition
$\mathcal{L}$	Defined loss function
K; N_k	Number of clients; Number of samples in k-th client
x_m; y_n	The m-th word and n-th word in two different language contents
ϕ_E	The encoder for text content representation
ϕ_D	The decoder for translating context representation
$\mathcal{M}$	The cross-lingual mapper for converting context representation
z_x	Text context representation vector
ϵ	The backbone model for user feature extraction
u_t	The tweets set from t-th user
t_i^m	The i-th word in user's m-th tweet
h_m^t	Representation for m-th tweet posted by t-th user
r_p; r_t	User's property representation; tweet-level representation
r_u,r	User-level representation
G; D_1, D_2	Generator; Discriminators
$\mathcal{D}_k$	The local data stored in the k-th client
$\mathcal{D}$	The data set collected from all clients
(x_i^k, y_i^k)	The i-th sample pair stored in the k-th client
$\mathcal{N}(0,1)$	The gaussian distribution
$\tilde{x}$	The pseudo-data generated by generators
p; $\hat{p}$	Probability of local data; Global data distribution probability
D_{KL}	The Kullback–Leibler divergence
τ	The temperature parameter for smoothing similarity value
γ, μ	The weight hyperparameter in the defined loss function
θ	The parameters of the designed model
M	The number of clients participating in the communication
$\alpha_t^{k,y}$	Ratio of samples with label y stored in $client_k$ against in $\mathcal{D}$

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 5: Statistics of datasets.

Dataset	Humans	Bots	Total number
Vendor-19 [40]	1860	568	2428
TwiBot-20 [16]	4175	5286	9461

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.

Test Dataset	Vendor-19	TwiBot-20
FedAvg	76.53±0.11	49.63±1.09
FedProx	74.26±3.57	54.21±2.01
Ensemble	76.01±0.65	51.98±5.50
FedDistill	73.11±0.95	51.19±1.12
FedGen	77.52±0.65	62.30±0.93
FedFTG	76.51±1.15	60.12±1.01
FedACK	**78.13±1.38**	**64.79±1.61**