TRUST: Towards Robust Social Bot Detection via Uncertainty-Guided Pseudo-Labeling and Graph Structure Purification

Ruixuan Xu[1], **Mengting Hu**[2*], **Zhunheng Wang**[1], **Ming Jiang**[1], **Rui Ying**[1], **Zhen Zhang**[2], **Hang Gao**[3], **Shuaipeng Liu**[4], **Renhong Cheng**[1]
[1]College of Computer Science, Nankai University, [2]College of Software, Nankai University
[3]College of Artificial Intelligence, Tianjin University of Science and Technology
[4]XiaoAI Team, Xiaomi
xuruixuan@mail.nankai.edu.cn, mthu@nankai.edu.cn

Abstract

Social bots threaten online platforms by mimicking human behavior and forming deceptive connections, enabling the dissemination of misinformation while evading detection. Existing graph-based detection models leverage graph neural networks (GNNs) to capture relational structures and multimodal user features. However, such models are vulnerable to deceptive message propagation, where bots deliberately interact with legitimate users. These interactions create heterophilous edges—connections between nodes with different labels (i.e. human and bot)—which undermine the homophily assumption that connected users typically share similar characteristics. In this work, we propose a novel framework to mitigate deceptive message propagation through node-level uncertainty estimation and graph structure purification. The framework comprises three key components: (1) *Node uncertainty estimation* employs evidential deep learning with an error-sensitive uncertainty loss to obtain calibrated node-wise uncertainty; (2) *Uncertainty-guided pseudo-label generation* assigns pseudo-labels to low-uncertainty nodes using a dynamic threshold; (3) *Graph structure purification* selectively disconnects heterophilous edges identified between differently labeled nodes. Extensive experiments on three benchmark datasets and six GNN backbones demonstrate that our framework consistently enhances detection performance and serves as an effective general-purpose enhancement module for social bot detection.

1 Introduction

Social bots pose a significant threat to the credibility of online platforms by mimicking human behavior and forming deceptive connections to influence public discourse (Ferrara, 2023). Their ability to infiltrate real user communities through realistic profiles, interactions, and network connections allows them to amplify misinformation and erode public trust by manipulating public opinion (Ferrara et al., 2020), spreading low-credibility content (Shao et al., 2018), eroding institutional trust (Denniss and Lindberg, 2025), and polarizing online communities (Bail et al., 2018). As social bots become increasingly scalable and sophisticated, the development of robust and generalizable detection methods has emerged as a critical goal in both academic research and real-world applications.

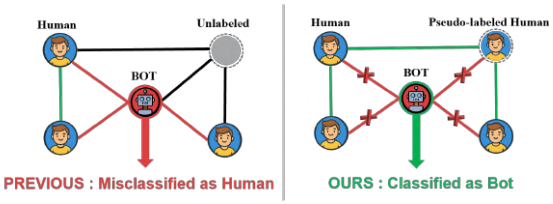


Figure 1: Illustration of deceptive message propagation (Left) and our framework TRUST (Right).

Graph neural networks (GNNs) have become the dominant paradigm for social bot detection due to their capacity to capture structural dependencies and fuse multimodal user information such as textual content, profile metadata, and behavioral patterns. Prior methods, including BotRGCN (Feng et al., 2021c), RGT (Feng et al., 2022a), HGT (Hu et al., 2020), and BotDGT (He et al., 2024), model heterogeneous user graphs with relational or attention-based message passing. Others like BotMoE (Liu et al., 2023), CACL (Chen et al., 2024), and ESA-BotRGCN (Zeng et al., 2025) incorporate multimodal signals such as text semantics, sentiment, and community structure. These approaches achieve strong performance by leveraging rich relational and semantic cues. However, they fundamentally rely on the homophily assumption that connected nodes are likely to share the same label, which facilitates effective representa-

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 31477–31491

tion learning (Kipf and Welling, 2017). In real-world social networks, this assumption often breaks down (Zhu et al., 2020). Social bots deliberately connect to legitimate users to camouflage themselves, forming heterophilous edges—connections between nodes with different labels such as human and bot—that enable the spread of incorrect label information. As illustrated in Figure 1 (left), a central bot node actively connects to nearby human nodes by initiating follow or interaction links. These heterophilous connections allow the bot to blend into human-dominated neighborhoods and mislead the aggregation process during message passing, ultimately causing it to be misclassified as human.

In this paper, we propose a novel framework named TRUST (Towards Robust Social Bot Detection via Uncertainty-Guided Pseudo-Labeling and Structure Purification), which aims to address the problem of deceptive message propagation caused by bot-human interactions. As shown in Figure 1 (right), our key idea is to estimate node-level uncertainty, assign pseudo-labels to low-uncertainty unlabeled nodes, remove heterophilous edges between nodes with inconsistent labels, and retain homophilous edges between nodes with consistent labels. Once pseudo-labels are generated, edges originally connecting labeled and unlabeled nodes can be reinterpreted as either homophilous or heterophilous based on label agreement. This process transforms previously ambiguous edges into confidently classified ones, enabling targeted structure purification and more reliable message passing. Together, these steps enhance structural consistency and improve classification robustness across diverse GNN architectures.

Specifically, our approach consists of the following three components. First, we perform *node uncertainty estimation* by adopting evidential deep learning (EDL) (Sensoy et al., 2018) to model prediction evidence with a Dirichlet distribution and derive node-wise uncertainty. To improve calibration, we combine an EDL loss with our proposed error-sensitive uncertainty loss that explicitly encourages low uncertainty for correctly classified nodes and high uncertainty for misclassified ones. Second, we implement *uncertainty-guided pseudo-label generation*. The uncertainty threshold is dynamically determined on the validation set by searching for the value that maximizes agreement between pseudo-labels and ground-truth labels. Nodes with uncertainty below this threshold are assigned pseudo-labels, which are used exclusively for structural refinement. Third, we conduct *graph structure purification*. We identify and remove heterophilous edges between pseudo-labeled nodes when their labels conflict. This targeted disconnection performs graph structure purification, mitigating deceptive message propagation and improving representation quality.

Our contributions are summarized as follows:

- To the best of our knowledge, this is the first work that leverages node-level uncertainty estimation to guide edge disconnection for mitigating deceptive message propagation in social bot detection.
- We propose a simple but effective framework TRUST for social bot detection that can be flexibly combined with various GNN architectures.
- Extensive experiments on six widely used GNN architectures demonstrate that our edge filtering module serves as a general-purpose plug-in, consistently improving detection performance across diverse backbones.

2 Related Work

2.1 Graph-based Social Bot Detection

Early feature-based methods (Varol et al., 2017) rely on hand-crafted features and ensemble classifiers, while recent approaches adopt graph neural networks (GNNs) to model social structures. Homogeneous GNNs (Kipf and Welling, 2017; Veličković et al., 2018) and heterogeneous variants (Hu et al., 2020; Feng et al., 2022a; Lv et al., 2021; Feng et al., 2021c) capture topological and semantic relationships in user graphs. To enhance detection, multimodal models (Lei et al., 2023; Lyu et al., 2023; Liu et al., 2023; Zeng et al., 2025) integrate textual, structural, and behavioral cues, while contrastive (Zhou et al., 2023; Chen et al., 2024) methods improve adaptability to dynamic or shifted distributions. Lightweight architectures (Yang et al., 2020; Hayawi et al., 2022; Ng and Carley, 2023; Yang et al., 2013) offer strong generalization with minimal input features. Despite strong performance, existing models often rely on deterministic softmax outputs, leading to error propagation under uncertainty (Guo et al., 2017). Our framework addresses both challenges by coupling evidential uncertainty estimation with

homophily-guided edge pruning, improving robustness in weakly supervised environments.

2.2 Uncertainty Estimation

Most existing methods rely on softmax entropy as a proxy for uncertainty, which is often poorly calibrated and tends to produce overconfident predictions (Guo et al., 2017). SEBot (Yang et al., 2024) estimates structural entropy at the graph level, but lacks fine-grained, node-level uncertainty modeling. To address this, our framework adopts evidential deep learning (EDL) (Sensoy et al., 2018) to quantify predictive uncertainty at the node level by modeling class probabilities with a Dirichlet distribution. We employ an EDL loss consisting of a mean squared error term and a KL divergence regularizer, and we propose an error-sensitive uncertainty loss that explicitly encourages low uncertainty for correctly classified nodes and high uncertainty for misclassified ones. Based on the resulting calibrated uncertainty, we design a dynamic thresholding strategy to assign pseudo-labels only to low-uncertainty nodes, which in turn guides edge pruning and structure refinement.

2.3 Graph Structure Optimization

Recent studies improve graph homophily by pruning or refining edges based on structural or semantic signals (Ye et al., 2023; Li et al., 2023; Ashmore and Chen, 2023; Qiao et al., 2025). However, they typically apply static pruning strategies, rely on fixed edge classifiers, or overlook uncertainty in pseudo-labels. In contrast, our framework combines EDL-based uncertainty estimation with Bayesian thresholding to select reliable pseudo-labels, followed by iterative edge filtering and GNN retraining. This process enhances structural consistency and improves model robustness under both weakly and fully supervised settings.

3 Methodology

3.1 Formulation and Overview

We formulate social bot detection as a node classification task on a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of user nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of social connections, and $\mathcal{R}$ is the set of edge relation types (e.g., *follower*, *friend*). Each node $i \in \mathcal{V}$ is associated with a binary label $y_i \in \{0, 1\}$, where $y_i = 0$ denotes a human and $y_i = 1$ denotes a bot. We denote the set of labeled nodes as $\mathcal{V}_L \subset \mathcal{V}$, which is further split into three disjoint subsets for training, validation, and testing: $\mathcal{V}_L = \mathcal{V}_{\text{train}} \cup \mathcal{V}_{\text{val}} \cup \mathcal{V}_{\text{test}}$. The remaining nodes are denoted as unlabeled nodes $\mathcal{V}_U$, and the complete node set can be expressed as $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_U$.

As shown in Figure 2, TRUST follows the workflow indicated by the arrows. Starting from the input graph where a camouflaged bot is linked to many human nodes, the node uncertainty estimation module encodes the graph with a GNN and an evidential head to obtain node representations and uncertainty scores u_i. The uncertainty-guided pseudo-label generation module then selects a validation-based threshold $\tau^\star$ and assigns pseudo-labels to nodes with low uncertainty. Finally, the graph purification module removes heterophilous edges and retrains the GNN on the purified graph, so that the camouflaged bot in the example is corrected from a human prediction to a bot prediction.

3.2 GNN Representation

To comprehensively characterize each user node $i \in \mathcal{V}$, we construct four types of features: (i) a description embedding $\boldsymbol{x}_i^{\text{des}} \in \mathbb{R}^{d_{\text{des}}}$ obtained from a RoBERTa-base encoder (Liu et al., 2019) over the profile description; (ii) a tweet embedding $\boldsymbol{x}_i^{\text{tweet}} \in \mathbb{R}^{d_{\text{tweet}}}$ obtained by encoding the user's tweets with RoBERTa and aggregating their sentence-level representations; (iii) a numerical feature vector $\boldsymbol{x}_i^{\text{num}} \in \mathbb{R}^{d_{\text{num}}}$ formed by normalized statistics such as followers count, following count, tweet count, account age, and screen-name length; and (iv) a categorical feature vector $\boldsymbol{x}_i^{\text{cat}} \in \mathbb{R}^{d_{\text{cat}}}$ with three binary indicators for protected, verified, and default-profile-image status. Each modality-specific feature vector is projected into a shared hidden space via a linear transformation followed by ReLU activation:

$$f_{\text{des}} : \mathbb{R}^{d_{\text{des}}} \to \mathbb{R}^{d_h}, \quad f_{\text{tweet}} : \mathbb{R}^{d_{\text{tweet}}} \to \mathbb{R}^{d_h},$$
$$f_{\text{num}} : \mathbb{R}^{d_{\text{num}}} \to \mathbb{R}^{d_h}, \quad f_{\text{cat}} : \mathbb{R}^{d_{\text{cat}}} \to \mathbb{R}^{d_h}.$$

The final node representation is obtained by concatenating the transformed modality embeddings:

$$\boldsymbol{h}_i^{(0)} = \text{Concat}\big(f_{\text{des}}(\boldsymbol{x}_i^{\text{des}}), f_{\text{tweet}}(\boldsymbol{x}_i^{\text{tweet}}), f_{\text{num}}(\boldsymbol{x}_i^{\text{num}}), f_{\text{cat}}(\boldsymbol{x}_i^{\text{cat}})\big) \in \mathbb{R}^{4d_h}. \quad (1)$$

Given the initial node embeddings $\boldsymbol{h}_i^{(0)} \in \mathbb{R}^{4d_h}$, we adopt a relation-aware GNN to encode the topological structure of the graph $\mathcal{G}$. The message passing process at layer $l+1$ follows the general GNN

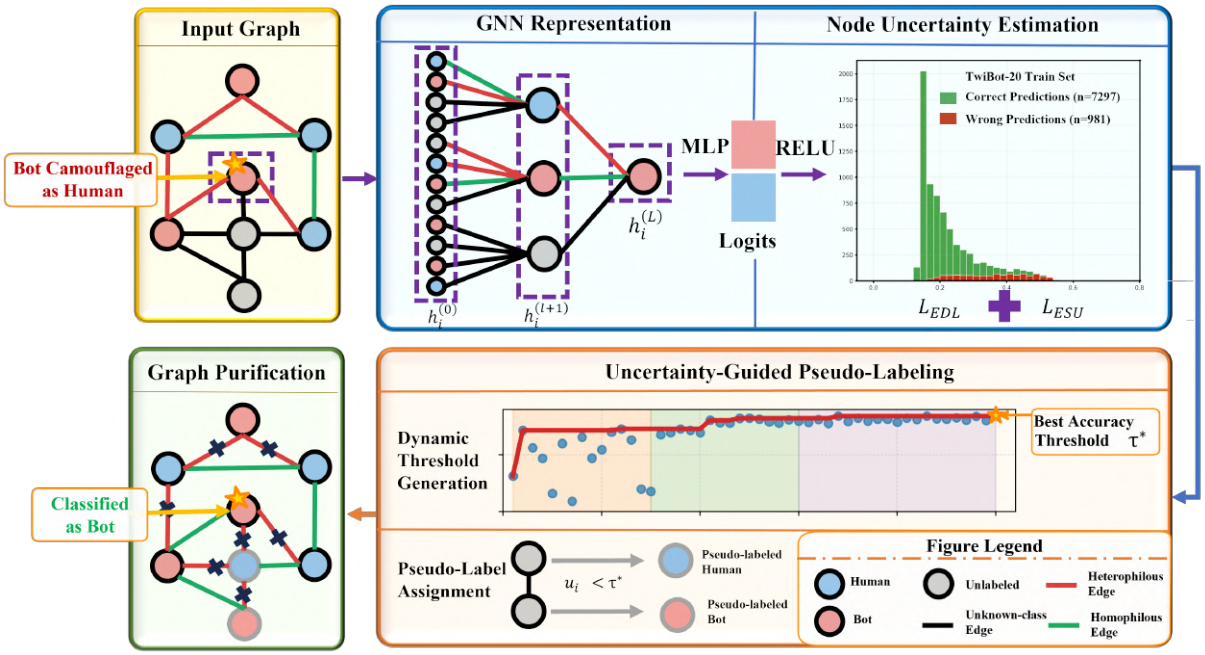


Figure 2: Overall architecture of TRUST. As illustrated on the left, a bot node can camouflage itself in the input graph; through node uncertainty estimation, uncertainty-guided pseudo-labeling, and graph structure purification, TRUST purifies the graph and ultimately identifies it as a bot.

paradigm:

$$\begin{aligned} \boldsymbol{a}_{i,r}^{(l+1)} &= \phi_r^{(l+1)}\left(\{\boldsymbol{h}_j^{(l)} : j \in \mathcal{N}_r(i)\}, \boldsymbol{h}_i^{(l)}\right), \\ \boldsymbol{h}_i^{(l+1)} &= \psi^{(l+1)}\left(\boldsymbol{h}_i^{(l)}, \{\boldsymbol{a}_{i,r}^{(l+1)}\}_{r\in\mathcal{R}}\right), \end{aligned} \tag{2}$$

where $\boldsymbol{a}_{i,r}^{(l+1)}$ denotes the relation-specific message aggregated for node i from its neighbors connected via relation $r \in \mathcal{R}$. $\phi_r^{(l+1)}$ denotes the relation-specific aggregation function at layer l+1. $\psi^{(l+1)}$ denotes the node update function at layer l+1. Each hidden representation $\boldsymbol{h}_i^{(l+1)}$ is recursively updated based on $\boldsymbol{h}_i^{(l)}$ and the aggregated messages from all relation types. After L layers of propagation, we obtain the final node representation $\boldsymbol{h}_i^{(L)}$.

3.3 Node Uncertainty Estimation

Conventional GNNs typically employ a Softmax function to output deterministic probabilities, which lacks the capability to capture predictive uncertainty (Sensoy et al., 2018). To address this, we adopt Evidential Deep Learning (EDL) to model the classification distribution as a Dirichlet distribution. For a node i, the model first projects its final embedding $\boldsymbol{h}_i^{(L)} \in \mathbb{R}^d$ into non-negative evidence $\mathbf{e}_i \in \mathbb{R}^K$ for K classes ($K = 2$ for bot detection). Conceptually, each $e_{i,k}$ quantifies the class-specific support accumulated from the node's attributes and graph topology, with larger values indicating stronger evidence for class k. This process is formulated as:

$$\mathbf{e}_i = \mathrm{ReLU}(\mathbf{W}\boldsymbol{h}_i^{(L)} + \mathbf{b}), \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{K\times d}$ is a learnable weight matrix, and $\mathbf{b} \in \mathbb{R}^K$ denotes the bias vector.

Given the evidence $\mathbf{e}_i$, we define the Dirichlet parameters $\boldsymbol{\alpha}_i$ and the total evidence strength S_i:

$$\boldsymbol{\alpha}_i = \mathbf{e}_i + \mathbf{1}, \quad S_i = \sum_{k=1}^{K} \alpha_{i,k} \tag{4}$$

where $\mathbf{1}$ is a vector of ones, and $\alpha_{i,k}$ represents the k-th element of $\boldsymbol{\alpha}_i$. The predictive class probability $\hat{\mathbf{p}}_i$ and the vacuity uncertainty u_i are derived as:

$$\hat{\mathbf{p}}_i = \frac{\boldsymbol{\alpha}_i}{S_i}, \quad u_i = \frac{K}{S_i} \tag{5}$$

Intuitively, a higher S_i implies that the model has gathered more evidence from the input data, leading to a lower uncertainty u_i. This effectively quantifies the uncertainty, where u_i decreases as the model becomes more knowledgeable about the node, shifting the Dirichlet distribution from a state of ignorance towards a specific class assignment.

Evidential Deep Learning Loss. To obtain calibrated class probabilities while discouraging overconfident predictions, we follow evidential deep

learning and learn the evidence parameters by minimizing the following loss:

$$\mathcal{L}_{\text{EDL}} = \frac{1}{|\mathcal{V}_{train}|} \sum_{i \in \mathcal{V}_{train}} \Big(\ell_{\text{MSE}}(i) + \ell_{\text{KL}}(i) \Big). \tag{6}$$

The $\ell_{\text{MSE}}(i)$ denotes the expected squared error under the induced Dirichlet distribution:

$$\ell_{\text{MSE}}(i) = \sum_{k=1}^{K} \left[(y_{i,k} - \hat{p}_{i,k})^2 + \frac{\hat{p}_{i,k}(1 - \hat{p}_{i,k})}{S_i + 1} \right], \tag{7}$$

where $y_{i,k}$ is the k-th component of the one-hot ground-truth label for node i, and $\hat{p}_{i,k}$ is the predicted probability for class k. This term aligns the predicted probabilities with the labels while explicitly penalizing the predictive variance. It ensures that high confidence predictions are supported by sufficient evidence.

The $\ell_{\text{KL}}(i)$ serves as a regularizer:

$$\ell_{\text{KL}}(i) = \lambda(t)\, \text{KL}[\text{Dir}(\boldsymbol{\alpha}_i) \parallel \text{Dir}(\mathbf{1})], \tag{8}$$

where $\text{Dir}(\mathbf{1})$ is a non-informative uniform prior. This KL regularizer suppresses unwarranted evidence by pulling $\boldsymbol{\alpha}_i$ toward $\mathbf{1}$ when support is insufficient, which mitigates overconfidence; $\lambda(t)$ is annealed from 0 to 1 over epochs.

Error-Sensitive Uncertainty Loss. To explicitly structure node-wise uncertainty, we propose an error-sensitive objective that encourages low uncertainty for correctly classified nodes and high uncertainty for misclassified ones. Specifically, we minimize the discrepancy between the estimated uncertainty u_i and a binary target $\tilde{u}_i$:

$$\mathcal{L}_{\text{ESU}} = \frac{1}{|\mathcal{V}_{train}|} \sum_{i \in \mathcal{V}_{train}} (u_i - \tilde{u}_i)^2, \tag{9}$$

$$\tilde{u}_i = \begin{cases} 0, & \text{if} \quad \hat{y}_i = y_i \\ 1, & \text{if} \quad \hat{y}_i \neq y_i \end{cases}$$

where y_i denotes the ground-truth class index of node i, $\hat{y}_i = \arg\max \hat{p}_{i,k}$. By explicitly penalizing low uncertainty on misclassified nodes and high uncertainty on correct ones, this objective ensures that the vacuity u_i reliably reflects the model's predictive confidence.

Overall Loss Function. The overall loss function combines $\mathcal{L}_{\text{EDL}}$ and $\mathcal{L}_{\text{ESU}}$:

$$\mathcal{L} = \lambda_{\text{edl}} \mathcal{L}_{\text{EDL}} + \lambda_{\text{esu}} \mathcal{L}_{\text{ESU}}, \tag{10}$$

where λ_{edl} and λ_{esu} are hyperparameters.

3.4 Uncertainty-Guided Pseudo-Labeling

Dynamic Threshold Generation. To determine a reliable threshold for pseudo-labeling, we formulate a one-dimensional hyperparameter optimization problem over the validation set. For each candidate threshold $\tau \in [0, 1]$, we select low-uncertainty nodes $\mathcal{S}(\tau) = \{i \in \mathcal{V}_{\text{val}} \mid u_i < \tau\}$, and evaluate the pseudo-label accuracy on this subset as $q(\tau) = \text{Accuracy}(\{(y_i, \hat{y}_i)\}_{i \in \mathcal{S}(\tau)})$. To ensure sufficient coverage, we discard thresholds with $|\mathcal{S}(\tau)| < 0.5 \cdot |\mathcal{V}_{\text{val}}|$. We adopt the Optuna framework (Akiba et al., 2019) with the Tree-structured Parzen Estimator (TPE) sampler (Bergstra et al., 2011), which adaptively proposes new candidate thresholds based on the accuracy history. The optimization objective is to find the threshold that maximizes accuracy:

$$\tau^\star = \arg\max_{\tau \in [0,1]} q(\tau), \tag{11}$$

where $\tau^\star$ denotes the optimal uncertainty threshold.

Pseudo-Label Assignment. Given the optimal uncertainty threshold $\tau^\star$, we assign pseudo-labels to nodes based on their predicted class probabilities and uncertainty estimates. Specifically, a node $i \in \mathcal{V}$ is assigned a pseudo-label if its uncertainty u_i satisfies $u_i < \tau^\star$. We collect nodes with pseudo-labels in $\mathcal{V}_{\text{P}} = \{i \in \mathcal{V} \mid u_i < \tau^\star\}$.

3.5 Graph Structure Purification

Based on the high-confidence pseudo-labels $\hat{y}_i$ for nodes $i \in \mathcal{V}_{\text{P}}$, we perform homophily-guided graph purification followed by GNN training on the optimized graph structure.

Homophily-Aware Edge Filtering. Motivated by the homophily principle in social networks—connected users tend to share the same label—we remove heterophilous edges and obtain a purified edge set $\mathcal{E}' = \{(i, j) \in \mathcal{E} \mid i \notin \mathcal{V}_{\text{P}} \vee j \notin \mathcal{V}_{\text{P}} \vee \hat{y}_i = \hat{y}_j\}$. The resulting purified graph is denoted as $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$. This procedure removes only edges that connect two high-confidence nodes with conflicting pseudo-labels, that is, edges (i, j) with $i, j \in \mathcal{V}_{\text{P}}$ and $\hat{y}_i \neq \hat{y}_j$, which reduces the impact of deceptive links in the graph.

GNN Training on the Purified Graph. We reinitialize all model parameters and retrain the GNN on the purified graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$. This reset-and-retrain strategy mitigates the impact of misleading edges in the original graph. The GNN architecture and message passing remain the same as described in the GNN representation in Section 3.2.

Method	Cresci-15			TwiBot-20			TwiBot-22		
	Acc.	F1	MCC	Acc.	F1	MCC	Acc.	F1	MCC
DeeProBot(2022)	72.35	81.61	44.26	77.34	80.32	54.40	74.11	12.73	14.12
EvolveBot (2013)	92.18	90.07	72.77	65.83	69.75	30.79	71.09	14.09	13.38
RoBERTa (2019)	97.01	95.86	89.55	75.55	73.09	42.15	72.07	20.53	19.35
BGSRD(2022)	87.78	90.80	74.92	66.36	70.05	32.28	71.88	21.14	20.32
SATAR(2021a)	92.71	94.55	85.61	84.02	85.74	67.83	/	/	/
GAT (2018)	93.52	94.99	86.78	82.24	84.93	64.95	74.17	43.75	28.12
SAGE (2017)	93.92	95.26	87.48	83.00	85.48	66.38	74.39	46.47	30.25
HGT (2020)	92.24	94.27	84.83	84.02	86.35	68.48	75.07	43.13	29.18
BotRGCN (2021c)	94.71	95.85	89.06	84.43	85.79	67.17	73.90	48.30	31.01
SHGN(2021)	93.13	94.68	85.91	83.93	86.33	68.40	72.53	49.59	30.74
RGT (2022a)	96.27	97.04	92.25	84.02	86.07	68.08	74.49	45.36	29.64
ESA-BotRGCN(2025)	97.25	97.83	93.26	87.46	88.83	74.68	/	/	/
GAT+CACL(2024)	94.51	95.42	88.71	83.52	86.39	68.56	74.50	44.27	28.94
SAGE+CACL(2024)	97.65	98.12	95.10	83.60	86.53	68.95	75.38	47.45	32.27
HGT+CACL(2024)	95.88	96.74	91.45	85.12	87.28	70.75	75.07	48.27	32.39
GAT+**TRUST**	**98.13**$^{\uparrow 4.61}$	**98.53**$^{\uparrow 3.54}$	**95.98**$^{\uparrow 9.20}$	85.46$^{\uparrow 3.22}$	87.28$^{\uparrow 2.35}$	70.97$^{\uparrow 6.02}$	76.99$^{\uparrow 2.82}$	45.54$^{\uparrow 1.79}$	38.39$^{\uparrow 10.27}$
SAGE+**TRUST**	97.94$^{\uparrow 4.02}$	98.38$^{\uparrow 3.12}$	95.58$^{\uparrow 8.10}$	87.66$^{\uparrow 4.66}$	88.97$^{\uparrow 3.49}$	75.21$^{\uparrow 8.83}$	79.23$^{\uparrow 4.84}$	56.59$^{\uparrow 10.12}$	45.98$^{\uparrow 15.73}$
HGT+**TRUST**	97.01$^{\uparrow 4.77}$	97.67$^{\uparrow 3.40}$	93.60$^{\uparrow 8.77}$	87.49$^{\uparrow 3.47}$	88.99$^{\uparrow 2.64}$	75.04$^{\uparrow 6.56}$	78.98$^{\uparrow 3.91}$	54.27$^{\uparrow 11.14}$	44.94$^{\uparrow 15.76}$
BotRGCN+**TRUST**	97.76$^{\uparrow 3.05}$	98.24$^{\uparrow 2.39}$	95.18$^{\uparrow 6.12}$	87.57$^{\uparrow 3.14}$	88.94$^{\uparrow 3.15}$	75.07$^{\uparrow 7.90}$	**79.37**$^{\uparrow 5.47}$	56.87$^{\uparrow 8.57}$	**46.38**$^{\uparrow 15.37}$
SHGN+**TRUST**	97.57$^{\uparrow 4.44}$	98.10$^{\uparrow 3.42}$	94.79$^{\uparrow 8.88}$	87.66$^{\uparrow 3.73}$	**89.14**$^{\uparrow 2.81}$	75.38$^{\uparrow 6.98}$	79.35$^{\uparrow 6.82}$	**57.13**$^{\uparrow 7.54}$	**46.38**$^{\uparrow 15.64}$
RGT+**TRUST**	97.94$^{\uparrow 1.67}$	98.38$^{\uparrow 1.34}$	95.58$^{\uparrow 3.33}$	**87.83**$^{\uparrow 3.81}$	89.11$^{\uparrow 3.04}$	**75.54**$^{\uparrow 7.46}$	79.35$^{\uparrow 4.86}$	54.72$^{\uparrow 9.36}$	46.01$^{\uparrow 16.37}$

Table 1: Overall performance comparison on Cresci-15, TwiBot-20 and TwiBot-22.

4 Experiments

4.1 Baselines

We compare our framework against a comprehensive set of existing methods (Hayawi et al., 2022; Yang et al., 2013; Liu et al., 2019; Guo et al., 2022; Feng et al., 2021a; Veličković et al., 2018; Hamilton et al., 2017; Hu et al., 2020; Feng et al., 2021c; Lv et al., 2021; Feng et al., 2022a; Chen et al., 2024; Zeng et al., 2025), covering a wide spectrum of social bot detection approaches. Full descriptions and details are provided in Appendix.

4.2 Datasets

We evaluate our framework on three large-scale Twitter bot detection benchmarks: Cresci-15 (Cresci et al., 2015) TwiBot-20 (Feng et al., 2021b) and TwiBot-22 (Feng et al., 2022b). We follow the same splits provided in the benchmarks. For fully annotated datasets, we further simulate a scarce-label setting within the training split by randomly selecting 50% of the training nodes as labeled and treating the remaining 50% as unlabeled.

4.3 Implementation

All experiments are conducted on two NVIDIA RTX A6000 GPUs. We use a 2-layer GNN with 2-hop neighborhoods. Full hyperparameter settings are provided in Appendix. Training on Cresci-15, TwiBot-20 and TwiBot-22 takes about 3 minutes, 30 minutes and 10 hours.

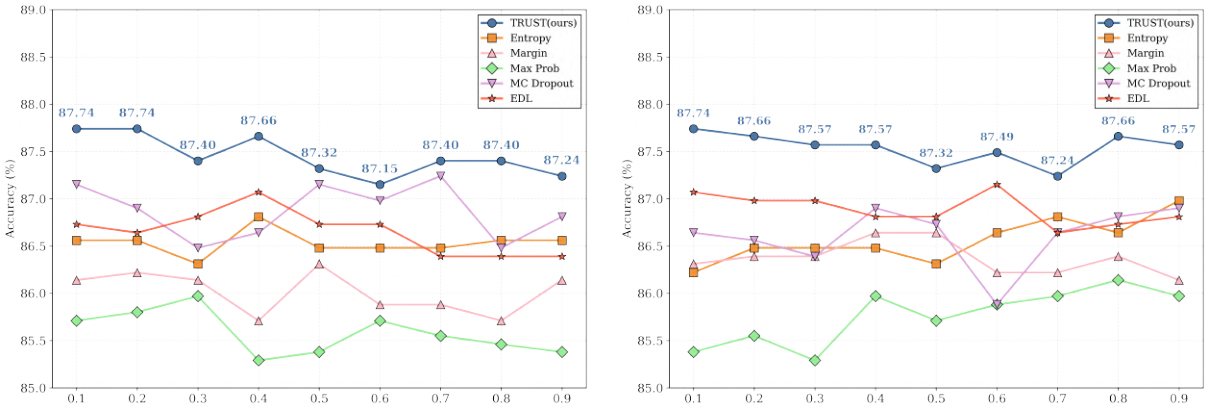
4.4 Main Results

We evaluate TRUST on Cresci-15, TwiBot-20 and TwiBot-22. Table 1 reports the overall results. Bold indicates the highest performance. "↑" denotes the improvement of TRUST over the backbone. "/" indicates not scalable to TwiBot-22. The results show that:

- TRUST can be flexibly combined with six representative GNN backbones (GAT, SAGE, HGT, BotRGCN, SHGN, and RGT), and consistently improves their performance across benchmarks, demonstrating strong generality.
- Under the same backbones, TRUST outperforms recent competitive baselines and further surpasses CACL-augmented variants, demonstrating our consistent effectiveness.
- In the scarce-label setting where only 50% of the training nodes are labeled and the rest are treated as unlabeled, TRUST still delivers strong gains and can even outperform fully supervised training of the same backbones.

4.5 Robustness Analysis

We evaluate the robustness of TRUST against structural perturbations, comparing it with five uncertainty estimation baselines: (i) Entropy estimates



(a) Robustness under edge dropout.

(b) Robustness under edge addition.

Figure 3: Robustness under structural perturbations on TwiBot-20 across different uncertainty estimation methods.

uncertainty via softmax entropy. (ii) Margin estimates uncertainty as the probability gap between the bot and human classes. (iii) Max Prob estimates uncertainty from the confidence of the most probable class prediction. (iv) MC Dropout estimates uncertainty by performing stochastic forward passes with dropout activated at inference. (v) EDL estimates uncertainty by modeling class-wise evidence through a Dirichlet distribution. All estimators produce a single uncertainty value for each node and are integrated identically into pseudo-label selection and structure purification.

Edge dropout. We randomly remove a fraction $\rho \in \{0.1, 0.2, \ldots, 0.9\}$ of edges to construct perturbed graphs for evaluation. Model weights are fixed; only the graph is perturbed. As shown in Figure 3a, TRUST (ours) consistently achieves the highest accuracy and exhibits the smallest fluctuation as ρ increases. Among the baselines, EDL is the most robust, while Entropy, Margin, Max Prob, and MC Dropout show larger accuracy drops under stronger edge removal.

Edge addition. We further inject spurious edges using the same perturbation ratios and evaluate robustness on the resulting noisy graphs. Model weights are fixed; only the graph is perturbed. As shown in Figure 3b, TRUST (ours) remains stable across perturbation levels and consistently outperforms all alternatives. In contrast, Entropy, Margin, Max Prob, MC Dropout, and EDL are more sensitive to injected noise and exhibit larger fluctuations under heavier perturbations.

Overall, Figure 3 shows that TRUST, with RGT as the backbone, is substantially more robust under edge dropout and edge addition than common uncertainty baselines.

Model +TRUST	Coverage (%) Labeled	Coverage (%) Unlabeled	Accuracy (%) Overall	Accuracy (%) Class 0	Accuracy (%) Class 1
GAT	47.06	36.57	97.56	97.23	98.01
SAGE	63.89	50.39	95.98	91.31	99.65
HGT	44.12	32.20	98.79	97.88	99.87
BotRGCN	55.77	42.12	97.51	95.48	99.41
SHGN	52.74	39.82	97.00	93.00	99.94
RGT	42.80	30.15	98.85	98.18	99.81

Table 2: Pseudo-label quality and coverage of TRUST on TwiBot-20 with different GNN backbones.

4.6 Pseudo-label Quality and Coverage

We evaluate pseudo-label quality and coverage because TRUST uses pseudo-labels both to augment supervision and to drive graph structure purification. Table 2 reports coverage on labeled and unlabeled nodes and accuracy on labeled nodes for six GNN backbones. The results show that TRUST produces pseudo-labels with consistently high accuracy, and coverage remains substantial on both labeled and unlabeled nodes.

4.7 Graph Structural Purification Analysis

We analyze graph structural purification in TRUST by measuring how many edges are removed and how the overall graph connectivity changes.

Edge retention. Table 3 summarizes the number and proportion of edges preserved or removed after applying TRUST. Across all backbones, only a small fraction of edges are disconnected and most of the original edges are retained.

Connectivity integrity. Figure 4 reports the largest connected component (LCC) ratio and the proportion of isolated nodes before and after purification. The LCC remains dominant and the isolated-node proportion changes only slightly, so

Model +TRUST	Edge Retention	Disconnected Edges
GAT	219,700 (96.37%)	8,279 (3.63%)
SAGE	202,742 (88.93%)	25,237 (11.07%)
HGT	216,982 (95.18%)	10,997 (4.82%)
BotRGCN	208,997 (91.67%)	18,982 (8.33%)
SHGN	213,570 (93.68%)	14,409 (6.32%)
RGT	218,677 (95.92%)	9,302 (4.08%)

Table 3: Edge retention and disconnection statistics of TRUST on TwiBot-20 with different GNN backbones.

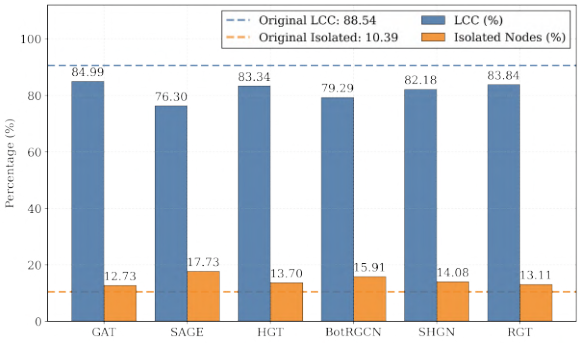


Figure 4: Largest connected component (LCC) ratio and isolated node proportion after graph purification.

the overall graph connectivity is preserved after edge removal.

Overall, these results suggest that TRUST improves performance without heavily fragmenting the graph or collapsing its global structure.

4.8 Ablation Study

We perform ablations on TwiBot-20 with RGT as the backbone to study the effect of each component in TRUST (Table 4).

Loss components. Removing the EDL loss (w/o EDL Loss) or the error-sensitive uncertainty loss (w/o ESU Loss) reduces accuracy, F1 and MCC compared with full TRUST. This indicates that both the evidential objective and the additional uncertainty shaping term are useful for training the uncertainty module.

Multimodal features. Dropping any feature type harms performance. Removing description features (w/o Des) or tweet features (w/o Tweet) degrades the results, and removing categorical profile attributes (w/o Cat) leads to the largest degradation among all feature groups. Numeric features (w/o Num) also contribute, but with a smaller impact.

Component ablation. Training without pseudo-labeling and structure purification (w/o PL & SP) gives a reasonable baseline, while adding pseudo-labels only (PL only) improves all three metrics.

Ablation	Accuracy (%)	F1-score (%)	MCC
TRUST	87.83	89.11	75.54
w/o EDL Loss	$86.90_{\downarrow 0.93}$	$88.28_{\downarrow 0.83}$	$73.66_{\downarrow 1.88}$
w/o ESU Loss	$86.98_{\downarrow 0.85}$	$88.59_{\downarrow 0.52}$	$74.07_{\downarrow 1.47}$
w/o Des	$85.71_{\downarrow 2.12}$	$87.19_{\downarrow 1.92}$	$71.24_{\downarrow 4.30}$
w/o Tweet	$83.77_{\downarrow 4.06}$	$86.31_{\downarrow 2.80}$	$68.30_{\downarrow 7.24}$
w/o Cat	$80.56_{\downarrow 7.27}$	$82.55_{\downarrow 6.56}$	$60.77_{\downarrow 14.77}$
w/o Num	$86.31_{\downarrow 1.52}$	$87.84_{\downarrow 1.27}$	$72.51_{\downarrow 3.03}$
w/o PL & SP	$86.56_{\downarrow 1.27}$	$87.95_{\downarrow 1.16}$	$72.95_{\downarrow 2.59}$
PL only	$87.24_{\downarrow 0.59}$	$88.53_{\downarrow 0.58}$	$74.31_{\downarrow 1.23}$

Table 4: Ablation study of TRUST on TwiBot-20.

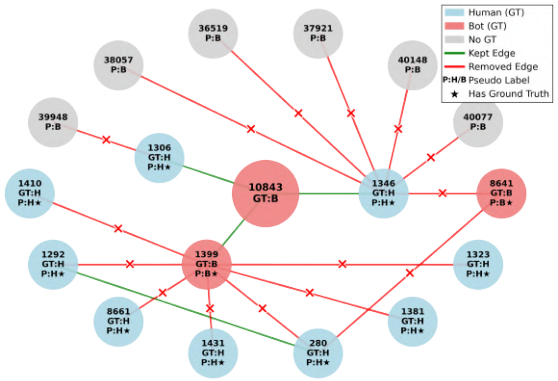


Figure 5: Case study of a corrected node. The 2-hop neighborhood subgraph centered at node 10843.

Enabling both pseudo-labeling and structure purification yields the best results.

4.9 Case Study

We present a qualitative case on TwiBot-20 with RGT+TRUST to illustrate how graph structure purification can correct predictions. The central node 10843 is a bot that is misclassified as human by the same model trained on the original graph without uncertainty-guided pseudo-label generation and structure purification, but becomes correctly classified as bot after applying TRUST. Figure 5 visualizes its 2-hop neighborhood subgraph. For clarity, we display only 2-hop neighbors that are incident to removed edges; the complete 2-hop neighborhood subgraph is provided in the appendix.

Conclusion

This paper presents TRUST, a robust framework for social bot detection that mitigates deceptive message propagation via uncertainty-guided pseudo-labeling and graph structure purification. TRUST leverages evidential deep learning to estimate node-level uncertainty, assigns pseudo-labels only to reliable nodes, and removes heterophilous edges to

obtain cleaner neighborhoods for representation learning. Experiments across six representative GNN backbones demonstrate that this design consistently improves bot detection performance.

Limitations

TRUST is built on graph neural networks and focuses on improving the quality of node representations through uncertainty-guided pseudo-labeling and structure purification. While this design effectively enhances bot detection performance, it does not exploit recent advances in large language models. Integrating TRUST with large models, or extending its uncertainty estimation and structure purification mechanisms to LLM-enhanced or multimodal graph architectures, remains an important direction for future work.

Ethical considerations

This study relies solely on publicly released social bot detection benchmarks provided for research use under their respective licenses. No additional data collection, user interaction, or analysis of real-world accounts is involved. We adhere to the usage terms of these datasets and treat all included information as potentially sensitive, refraining from any attempt to re-identify users, infer private attributes, or link identities beyond what is explicitly provided. We acknowledge that automated bot detection carries potential risks, such as misuse for surveillance or censorship and unintended harm caused by incorrect predictions, particularly false positives. Our method is therefore presented as a research tool intended to support platform integrity efforts rather than standalone enforcement, and any practical use should incorporate uncertainty-aware decision making, restricted access to model outputs, and human oversight in high-stakes scenarios.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.62406151).

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 2623–2631.

Bradley Ashmore and Lingwei Chen. 2023. Hover: Homophilic oversampling via edge removal for class-imbalanced bot detection on graphs. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 3728–3732.

Christopher A. Bail, Lisa P. Argyle, Taylor W. Brown, John P. Bumpus, Haohan Chen, M. B. Fallin Hunzaker, Jaemin Lee, Marcus Mann, Friedolin Merhout, and Alexander Volfovsky. 2018. Exposure to opposing views on social media can increase political polarization. *Proceedings of the National Academy of Sciences*, 115(37):9216–9221.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24.

Sirry Chen, Shuo Feng, Liang Songsong, Chen-Chen Zong, Jing Li, and Piji Li. 2024. CACL: Community-aware heterogeneous graph contrastive learning for social media bot detection. In *Findings of the Association for Computational Linguistics (ACL)*, pages 10349–10360.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems*, 80:56–71.

Emily Denniss and Rebecca Lindberg. 2025. Social media and the spread of misinformation: infectious and a threat to public health. *Health Promotion International*, 40(2):daaf023.

Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo. 2022a. Heterogeneity-aware twitter bot detection with relational graph transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 3977–3985.

Shangbin Feng, Zhaoxuan Tan, Herun Wan, Ningnan Wang, Zilong Chen, Binchi Zhang, Qinghua Zheng, Wenqian Zhang, Zhenyu Lei, Shujie Yang, Xinshun Feng, Qingyue Zhang, Hongrui Wang, Yuhan Liu, Yuyang Bai, Heng Wang, Zijian Cai, Yanbo Wang, Lijing Zheng, and 3 others. 2022b. Twibot-22: Towards graph-based twitter bot detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 35254–35269.

Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021a. SATAR: A self-supervised approach to twitter account representation learning and its application in bot detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 3808–3817.

Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021b. Twibot-20: A comprehensive twitter bot detection benchmark. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 4485–4494.

Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. 2021c. Botrgcn: Twitter bot detection with relational graph convolutional networks. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 236–239.

Emilio Ferrara. 2023. Social bot detection in the age of chatgpt: Challenges and opportunities. *First Monday*, 28(6).

Emilio Ferrara, Stefano Cresci, and Luca Luceri. 2020. Misinformation, manipulation, and abuse on social media in the era of COVID-19. *Journal of Computational Social Science*, 3(2):271–277.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (PMLR)*, volume 70, pages 1321–1330.

Qinglang Guo, Haiyong Xie, Yangyang Li, Wen Ma, and Chao Zhang. 2022. Social bots detection via fusing bert and graph convolutional networks. *Symmetry*, 14(1).

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1025–1035.

Kadhim Hayawi, Sujith Mathew, Neethu Venugopal, Mohammad M. Masud, and Pin-Han Ho. 2022. DeeProBot: A hybrid deep neural network model for social bot detection based on user profile data. *Social Network Analysis and Mining*, 12(1):43.

Buyun He, Yingguang Yang, Qi Wu, Hao Liu, Renyu Yang, Hao Peng, Xiang Wang, Yong Liao, and Pengyuan Zhou. 2024. Dynamicity-aware social bot detection with dynamic graph transformers. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5844–5852.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference (WWW)*, page 2704–2710.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Zhenyu Lei, Herun Wan, Wenqian Zhang, Shangbin Feng, Zilong Chen, Jundong Li, Qinghua Zheng, and Minnan Luo. 2023. BIC: Twitter bot detection with text-graph interaction and semantic consistency. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 10326–10340.

Shilong Li, Boyu Qiao, Kun Li, Qianqian Lu, Meng Lin, and Wei Zhou. 2023. Multi-modal social bot detection: Learning homophilic and heterophilic connections adaptively. In *Proceedings of the 31st ACM International Conference on Multimedia (ACM MM)*, pages 3908–3916.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yuhan Liu, Zhaoxuan Tan, Heng Wang, Shangbin Feng, Qinghua Zheng, and Minnan Luo. 2023. BotMoE: Twitter bot detection with community-aware mixtures of modal-specific experts. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 485–495.

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? revisiting, benchmarking, and refining heterogeneous graph neural networks. *Preprint*, arXiv:2112.14936.

Nuoyan Lyu, Bingbing Xu, Fangda Guo, and Huawei Shen. 2023. DCGNN: Dual-channel graph neural network for social bot detection. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 4155–4159.

Lynnette Hui Xian Ng and Kathleen M. Carley. 2023. Botbuster: Multi-platform bot detection using a mixture of experts. In *Proceedings of the International AAAI Conference on Web and Social Media (AAAI)*, pages 686–697.

Boyu Qiao, Wei Zhou, Kun Li, Shilong Li, and Songlin Hu. 2025. Dispelling the fake: Social bot detection based on edge confidence evaluation. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):7302–7315.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. 2018. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1183–1193.

Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. 2018. The spread of low-credibility content by social bots. *Nature communications*, 9(1):4787.

Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2017. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the International AAAI Conference on Web and Social Media (AAAI)*, pages 280–289.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chao Yang, Robert Harkreader, and Guofei Gu. 2013. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293.

Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. 2020. Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 1096–1103.

Yingguang Yang, Qi Wu, Buyun He, Hao Peng, Renyu Yang, Zhifeng Hao, and Yong Liao. 2024. Sebot: Structural entropy guided multi-view contrastive learning for social bot detection. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, page 3841–3852.

Sen Ye, Zhaoxuan Tan, Zhenyu Lei, Ruijie He, Hongrui Wang, Qinghua Zheng, and Minnan Luo. 2023. HOFA: Twitter bot detection with homophily-oriented augmentation and frequency adaptive attention. *Preprint*, arXiv:2306.12870.

Kaqian Zeng, Zhao Li, and Xiujuan Wang. 2025. Emoji-driven sentiment analysis for social bot detection with relational graph convolutional networks. *Sensors*, 25(13).

Ming Zhou, Dan Zhang, Yuandong Wang, Yangli-Ao Geng, and Jie Tang. 2023. Detecting social bot on the fly using contrastive learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 4995–5001.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7793–7804.

A Appendix

A.1 Baselines

We compare TRUST with a range of competitive baselines widely used in social bot detection.

DeeProBot (Hayawi et al., 2022) A neural bot detector that encodes profile metadata and text and fuses them for classification.

EvolveBot (Yang et al., 2013) A feature engineering method that designs robust metadata and behavior features for traditional classifiers.

RoBERTa (Liu et al., 2019) A pre trained transformer language model fine tuned on user text as a strong text only baseline.

BGSRD (Guo et al., 2022) A hybrid BERT plus GCN model that combines semantic and structural predictions for bot detection.

SATAR (Feng et al., 2021a) A self supervised Twitter account representation framework that encodes tweets, profile attributes, and neighbors, then is fine tuned for bot detection.

GAT (Veličković et al., 2018) An attention based GNN that learns importance weights over neighbors for non uniform message aggregation.

SAGE (Hamilton et al., 2017) An inductive GNN that samples local neighborhoods and applies learnable aggregators to generate node embeddings.

HGT (Hu et al., 2020) A transformer style GNN for heterogeneous graphs with type specific attention over nodes and relations.

BotRGCN (Feng et al., 2021c) A relational GNN on heterogeneous follow graphs that uses relation types and multimodal features for bot detection.

SHGN (Lv et al., 2021) A simplified heterogeneous GNN that decouples relation specific transformations and structure learning.

RGT (Feng et al., 2022a) A relational graph transformer with relation aware attention for Twitter style heterogeneous graphs.

CACL (Chen et al., 2024) A community aware heterogeneous graph contrastive learning framework that mines hard positives and negatives from community structure.

ESA-BotRGCN (Zeng et al., 2025) An emoji-driven multi-modal detection framework that integrates emoji-to-text transformation, sentiment quantification, and heterogeneous social graph modeling for robust bot detection.

A.2 Hyperparameter Settings

Tables 5–10 summarize the full hyperparameter settings for the six GNN backbones.

Hyperparameter	Cresci-15	TwiBot-20	TwiBot-22
batch_size	1024	1024	1024
num_neighbors	256	256	256
lr	0.001	0.001	0.0002
dropout	0.4	0.3	0.5
λ_{edl}	0.5	0.5	0.5
λ_{esu}	2	4	4
original_epochs	300	300	100
purified_epochs	300	300	150

Table 5: Hyperparameter settings of GAT.

Hyperparameter	Cresci-15	TwiBot-20	TwiBot-22
batch_size	1024	1024	1024
num_neighbors	256	256	256
lr	0.001	0.00005	0.0002
dropout	0.4	0.4	0.5
λ_{edl}	0.5	0.5	0.5
λ_{esu}	4	4	4
original_epochs	200	300	100
purified_epochs	200	300	150

Table 6: Hyperparameter settings of SAGE.

Hyperparameter	Cresci-15	TwiBot-20	TwiBot-22
batch_size	1024	1024	1024
num_neighbors	256	256	256
lr	0.001	0.0005	0.0002
dropout	0.1	0.4	0.5
λ_{edl}	0.5	0.5	0.5
λ_{esu}	2	4	4
original_epochs	200	300	100
purified_epochs	300	300	150

Table 7: Hyperparameter settings of HGT.

Hyperparameter	Cresci-15	TwiBot-20	TwiBot-22
batch_size	1024	1024	1024
num_neighbors	256	256	256
lr	0.001	0.0003	0.0002
dropout	0.5	0.2	0.5
λ_{edl}	0.5	0.5	0.5
λ_{esu}	2	4	4
original_epochs	300	250	100
purified_epochs	300	150	150

Table 8: Hyperparameter settings of BotRGCN.

Hyperparameter	Cresci-15	TwiBot-20	TwiBot-22
batch_size	1024	1024	1024
num_neighbors	256	256	256
lr	0.0005	0.0001	0.0002
dropout	0.2	0.2	0.5
λ_{edl}	0.5	0.5	0.5
λ_{esu}	2	4	4
original_epochs	200	300	100
purified_epochs	300	300	150

Table 9: Hyperparameter settings of SHGN.

Hyperparameter	Cresci-15	TwiBot-20	TwiBot-22
batch_size	1024	1024	1024
num_neighbors	256	256	256
lr	0.001	0.001	0.0002
dropout	0.1	0.2	0.5
λ_{edl}	0.5	0.5	0.5
λ_{esu}	4	4	4
original_epochs	100	300	100
purified_epochs	200	300	150

Table 10: Hyperparameter settings of RGT.

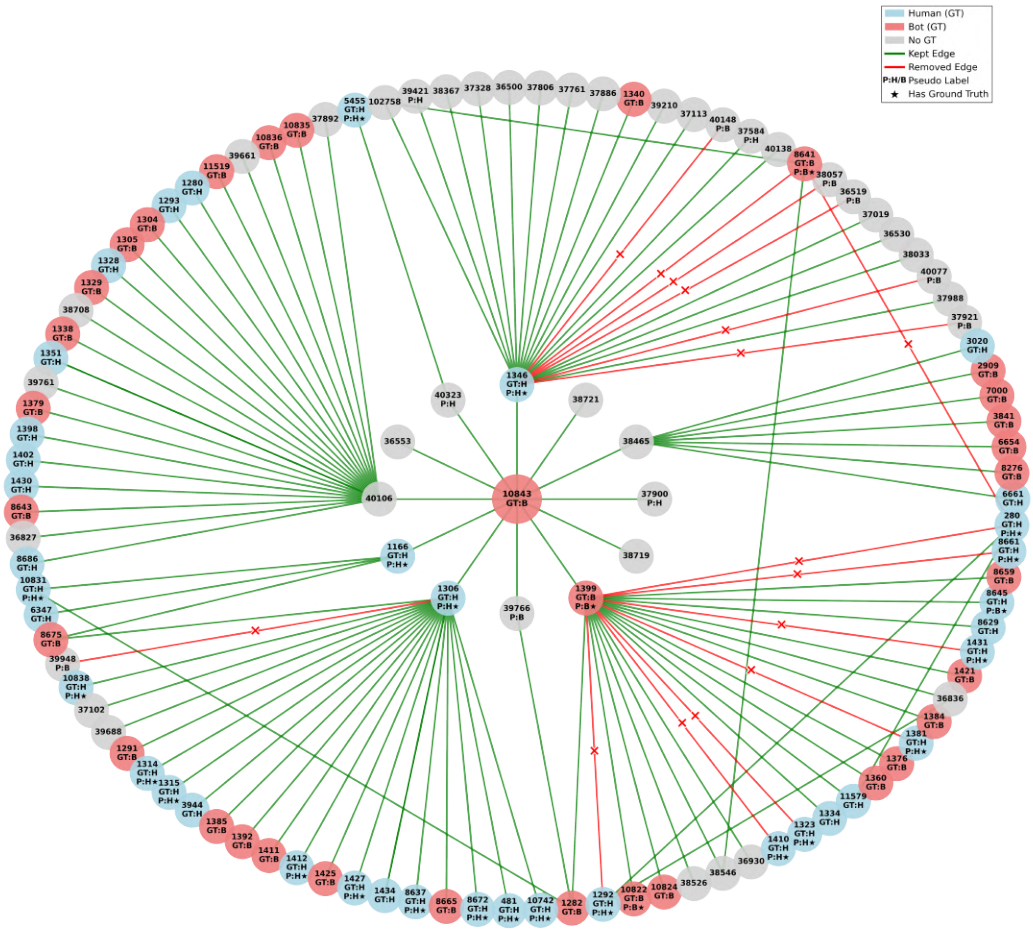


Figure 6: Complete 2-hop neighborhood subgraph centered at node 10843, corresponding to the simplified visualization in Figure 5.

A.3 Full Case Study

This appendix provides the full visualization for the example in Section 4.9 on TwiBot-20 with RGT+TRUST. The central node 10843 is a bot that is misclassified as human by the same model trained on the original graph without uncertainty-guided pseudo-label generation and structure purification, but becomes correctly classified as bot after applying TRUST. Figure 6 reports the complete 2-hop neighborhood subgraph of node 10843, while Figure 5 provides a simplified view for readability by displaying only 2-hop neighbors that are incident to removed edges.

We further quantify how graph structure purification affects different types of labeled edges. Since unlabeled nodes do not have ground-truth labels, we compute the following statistics only on edges connecting two ground-truth labeled nodes, which enables us to categorize these edges as human–human, human–bot, or bot–bot. In this full subgraph, structure purification predominantly removes heterophilous human–bot connections, with a removal rate of 34.62% (9/26). By contrast, within this 2-hop subgraph, we do not observe removals on labeled human–human (0/16) or labeled bot–bot edges (0/9). This pattern aligns with the purpose of our purification rule: reducing heterophilous edges around the target node and yielding a cleaner neighborhood for message passing, which helps correct the prediction of node 10843.

A.4 Confidence Calibration

Reliable uncertainty estimation is important in TRUST because pseudo-label selection and graph purification are both driven by model confidence. Therefore, in addition to detection metrics, we further evaluate calibration quality by reporting Maximum Calibration Error (MCE) (Guo et al., 2017).

Method	Cresci-15				TwiBot-20			
	Acc.	F1	MCC	MCE	Acc.	F1	MCC	MCE
HGT+Entropy	96.82	97.54	93.24	0.7304	86.14	87.48	72.06	0.3276
HGT+**TRUST**	**97.01**$^{\uparrow 0.19}$	**97.67**$^{\uparrow 0.13}$	**93.60**$^{\uparrow 0.36}$	**0.3246**$_{\downarrow 0.4058}$	**87.49**$^{\uparrow 1.35}$	**88.99**$^{\uparrow 1.51}$	**75.04**$^{\uparrow 2.98}$	**0.2949**$_{\downarrow 0.0327}$
BotRGCN+Entropy	97.38	97.95	94.37	0.8792	85.88	87.38	71.60	0.2507
BotRGCN+**TRUST**	**97.76**$^{\uparrow 0.38}$	**98.24**$^{\uparrow 0.29}$	**95.18**$^{\uparrow 0.81}$	**0.3802**$_{\downarrow 0.4990}$	**87.57**$^{\uparrow 1.69}$	**88.94**$^{\uparrow 1.56}$	**75.07**$^{\uparrow 3.47}$	**0.2468**$_{\downarrow 0.0039}$
GAT+Entropy	97.01	97.68	93.63	0.5588	84.87	86.61	69.62	0.5124
GAT+**TRUST**	**98.13**$^{\uparrow 1.12}$	**98.53**$^{\uparrow 0.85}$	**95.98**$^{\uparrow 2.35}$	**0.4280**$_{\downarrow 0.1308}$	**85.46**$^{\uparrow 0.59}$	**87.28**$^{\uparrow 0.67}$	**70.97**$^{\uparrow 1.35}$	**0.2374**$_{\downarrow 0.2750}$
SAGE+Entropy	97.20	97.82	94.00	0.6521	87.15	88.76	74.44	0.4077
SAGE+**TRUST**	**97.94**$^{\uparrow 0.74}$	**98.38**$^{\uparrow 0.56}$	**95.58**$^{\uparrow 1.58}$	**0.3111**$_{\downarrow 0.3410}$	**87.66**$^{\uparrow 0.51}$	**88.97**$^{\uparrow 0.21}$	**75.21**$^{\uparrow 0.77}$	**0.2913**$_{\downarrow 0.1164}$
SHGN+Entropy	96.45	97.26	92.45	0.8382	86.56	87.91	72.94	0.2991
SHGN+**TRUST**	**97.57**$^{\uparrow 1.12}$	**98.10**$^{\uparrow 0.84}$	**94.79**$^{\uparrow 2.34}$	**0.4205**$_{\downarrow 0.4177}$	**87.66**$^{\uparrow 1.10}$	**89.14**$^{\uparrow 1.23}$	**75.38**$^{\uparrow 2.44}$	**0.2766**$_{\downarrow 0.0225}$
RGT+Entropy	97.01	97.67	93.60	0.6787	85.55	87.06	70.90	0.7143
RGT+**TRUST**	**97.94**$^{\uparrow 0.93}$	**98.38**$^{\uparrow 0.71}$	**95.58**$^{\uparrow 1.98}$	**0.4401**$_{\downarrow 0.2386}$	**87.83**$^{\uparrow 2.28}$	**89.11**$^{\uparrow 2.05}$	**75.54**$^{\uparrow 4.64}$	**0.2206**$_{\downarrow 0.4937}$

Table 11: Results with MCE on Cresci-15 and TwiBot-20 across backbones.

MCE is defined as:

$$\mathrm{MCE} = \max_{m \in \{1,\ldots,M\}} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|, \tag{12}$$

where B_m denotes the set of samples whose prediction confidence falls into the m-th confidence bin. MCE measures the largest calibration gap over all bins, i.e., the worst miscalibrated region. This metric is particularly relevant in our setting, since poorly calibrated predictions may introduce unreliable pseudo-labels and thus affect subsequent graph purification. In our implementation, MCE is computed using TorchMetrics (`BinaryCalibrationError`)[1] with `norm="max"` and `n_bins=10`.

For the probability-based uncertainty baseline, we use predictive entropy. Specifically, TwiBot-22 defines the uncertainty score as

$$\phi = -\big(\hat{y}_0 \log \hat{y}_0 + \hat{y}_1 \log \hat{y}_1\big), \tag{13}$$

where $\hat{y}_0$ and $\hat{y}_1$ denote the predicted probabilities that a node is a genuine user and a bot, respectively. This is the binary form of Shannon entropy. More generally, for a K-class predictive distribution $\mathbf{p} = (p_1, \ldots, p_K)$, the predictive entropy is

$$H = -\sum_{k=1}^{K} p_k \log p_k. \tag{14}$$

A larger entropy value indicates higher uncertainty and lower confidence. This formulation is consistent with the Entropy baseline used in our experiments, where uncertainty is computed purely from softmax probabilities.

Table 11 reports Accuracy, F1-score, MCC, and MCE on Cresci-15 and TwiBot-20 across six GNN backbones. The results show that TRUST consistently achieves lower MCE than the probability-based Entropy baseline, indicating better calibration quality and a more reliable uncertainty signal for pseudo-label selection and graph purification. At the same time, TRUST delivers competitive and often improved detection performance across backbones on both datasets.

A.5 Effectiveness of the Uncertainty Module

We evaluate the contribution of the uncertainty-related components in TRUST, including Evidential Deep Learning (EDL), the Error-Sensitive Uncertainty loss (ESU), and Dynamic Thresholding (DT). Since TRUST uses uncertainty as a reliability signal for pseudo-label selection and graph purification, calibration quality is also important in addition to detection performance; therefore, we additionally report Maximum Calibration Error (MCE) in this ablation study. As shown in Table 12, the full design, i.e., EDL + ESU + DT (TRUST), achieves the best performance on all four metrics. In contrast, static thresholds (e.g., 0.4 or 0.8) lead to unstable and often inferior results, indicating that DT is necessary to balance pseudo-label quality and coverage across different settings. Moreover, removing ESU generally worsens calibration, as reflected by higher MCE values, suggesting that uncertainty should be explicitly shaped to align with prediction correctness before it can serve as a reliable filtering signal for pseudo-label selection

[1] https://lightning.ai/docs/torchmetrics/stable/classification/calibration_error.html

RGT+TRUST	Acc.	F1	MCC	MCE
EDL+0.4	85.00	86.00	71.00	0.3378
EDL+0.8	86.30	87.40	74.00	0.3116
EDL+1.0 (no filtering)	78.11	79.36	56.12	0.3291
EDL+ESU+0.4	85.80	87.31	71.43	0.2429
EDL+ESU+0.8	76.42	78.52	52.41	0.3117
EDL+ESU+1.0 (no filtering)	86.31	87.75	72.45	0.3210
EDL+ESU+DT (TRUST)	**87.83**	**89.11**	**75.54**	**0.2206**

Table 12: Ablation study on the uncertainty module in TRUST with the RGT backbone.

Disconnect ratio	Acc.	F1	MCC
0.1	86.64	87.92	73.09
0.2	86.81	88.15	73.45
0.3	86.31	87.48	72.39
0.4	86.56	88.09	73.05
0.5	86.39	87.94	72.71
0.6	86.05	87.28	71.88
0.7	86.39	87.78	72.60
0.8	86.56	87.83	72.91
0.9	86.31	87.61	72.40
TRUST	**87.83**	**89.11**	**75.54**

Table 13: Comparison with random edge pruning without uncertainty in TRUST using the RGT backbone.

and subsequent graph purification.

A.6 Comparison with Random Edge Pruning

We examine whether the performance gain of TRUST can be reproduced by simply removing edges without uncertainty guidance. To this end, we construct a baseline that randomly disconnects edges at different ratios from 0.1 to 0.9. As shown in Table 13, random edge pruning achieves its best result at 86.81% accuracy when the disconnection ratio is 0.2, but remains consistently below TRUST across all settings, indicating that the improvement cannot be attributed to graph sparsification alone. This is because random disconnection reduces edge density indiscriminately and may remove both harmful and beneficial connections, and therefore cannot systematically suppress deceptive message propagation caused by label-conflicting interactions. In contrast, TRUST uses calibrated uncertainty to concentrate pruning decisions on edges that are most likely to inject contradictory signals into message passing, while preserving structurally informative connections. Therefore, the gain of TRUST comes not from the simple act of removing edges, but from uncertainty-guided and reliability-aware graph purification.