Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-4, April 2026
ISSN: 3049-2602

EVASIVE BOT DETECTION USING HETEROGENEOUS DATA FUSION AND TRI-NETWORK ARCHITECTURES

Rajkumar Kalapala
Student, Department of Computer Science & Engineering - (Data Science)
Andhra Loyola Institute of Engineering & Technology, Vijayawada, Andhra Pradesh, India
Email id: rajkumarkalapala@outlook.com

Abstract: Social media ecosystems face a growing threat from algorithmically controlled accounts that mimic authentic human behaviour at scale. Addressing this challenge, we introduce a system called Evasive Bot Detection (EBD) — a purpose-built framework that examines three independent channels of user information rather than relying on any one alone. At its core, EBD orchestrates three specialised neural sub-networks in parallel: a transformer-based module for interpreting the language of posts and profile biographies, a convolutional module for analysing profile photographs, and a graph-based module for characterising the structure of each account's social neighbourhood. Outputs from these sub-networks — 768, 512, and 128 dimensions respectively — are merged into a single 1,408-dimensional descriptor before a compact multi-layer classifier issues a bot-or-human verdict. Experiments conducted on the TwiBot-20 evaluation corpus yield an overall accuracy of 69.0 %, a precision figure of 74.0 %, and an F1-score of 69.8 %. Crucially, the cross-channel design prevents an adversary from gaming one data source while being exposed by another. A publicly accessible demonstration interface is available on Hugging Face Spaces.

Keywords: Evasive Bot Detection, Heterogeneous Data Fusion, Tri-Network Architecture, Graph Convolutional Network, DistilBERT, ResNet-18, TwiBot-20, Social Media Integrity, Multimodal Classification, Fake Account Identification

1. INTRODUCTION

Automated accounts — commonly termed bots — have long been a disruptive element across social media ecosystems. What was once a straightforward pattern-matching problem has since evolved into a sophisticated arms race. Contemporary threat actors combine coordinated botnet infrastructures with generative language models capable of composing posts indistinguishable from those written by genuine people. The upshot is that detection schemes anchored to a single signal, particularly text alone, are increasingly inadequate: a system optimised against one detector can simply tune that dimension while its anomalies in other dimensions go unexamined.

To appreciate why multiple channels matter, consider a concrete scenario. An adversary deploys an account whose tweets are authored by a large language model and therefore pass every fluency check. That same account, however, was created in bulk alongside nine thousand others, so it shares a near-identical follower graph topology with known bots; it also uses a stock photograph as its avatar, a tell-tale sign detectable through visual feature analysis. No text classifier will catch this account; a vision or graph classifier working in isolation might catch part of it; but a system that cross-references all three streams simultaneously is far more likely to surface the contradiction.

This reasoning underpins our proposed framework. EBD treats each information channel — text, image, and social graph — as an independent witness to account authenticity, then combines their testimony at a fusion layer before rendering a final decision. The architecture is deliberately late-fusion: each sub-network is allowed to specialise without interference from the others, preserving the discriminative signal unique to each modality.

The principal contributions of this work are as follows:

- We design a three-branch neural architecture in which a transformer, a convolutional network, and a graph convolutional network operate concurrently on different views of the same account, producing complementary evidence streams that are merged for classification.

- We propose a heterogeneous feature fusion strategy that assembles a 1,408-dimensional joint descriptor from modality-specific embeddings of differing sizes, enabling the downstream classifier to exploit cross-channel correlations.
- We report an experimental precision of 74.0 % on TwiBot-20, a figure directly relevant to real-world deployment where wrongly suspending legitimate users carries significant reputational and legal consequences.
- We release a browser-accessible inference tool on Hugging Face Spaces through which practitioners can submit arbitrary profile data and receive instant predictions without installing any software.

2. LITERATURE SURVEY

Scholarly interest in automated account detection stretches back to the early days of Twitter's public API. Initial investigations, such as the work of Varol and colleagues [1], constructed compact feature sets from observable account statistics — post volume, timing regularity, web-link density — and applied conventional classifiers such as random forests. These methods performed well when bot operators made no attempt at concealment, but collapsed as soon as attackers began deliberately sculpting their metadata to mimic organic distributions.

The introduction of deep sequence models reopened the detection problem at the textual level. Bidirectional recurrent architectures and, later, the attention mechanism of the transformer family offered richer representations of tweet content than bag-of-words approaches had. Devlin et al. [3] demonstrated that a pre-trained bidirectional encoder fine-tuned on downstream classification tasks achieves remarkable linguistic understanding, and several research groups subsequently applied BERT-family models to social media abuse detection [4]. For resource-constrained scenarios, Sanh et al. [7] proposed knowledge distillation to compress BERT into a leaner model that preserves the bulk of its representational capacity while running considerably faster — a property our system exploits directly.

A parallel strand of research recognised that social graphs carry independent diagnostic information. Individual accounts are nodes in a network whose edges encode follow relationships, retweets, and replies; automated accounts tend to form structurally stereotyped clusters that differ measurably from the loosely connected, organically grown communities surrounding human users. Graph Convolutional Networks, formalised by Kipf and Welling [5], offered a computationally tractable mechanism for aggregating neighbourhood information into node-level embeddings, and BotRGCN [4] demonstrated their value specifically for Twitter bot identification.

The publication of benchmark corpora has been a decisive driver of methodological progress. Feng et al. [6] assembled TwiBot-20 with the explicit goal of capturing the diversity of modern bot campaigns, spanning political astroturfing, financial spam, and coordinated amplification operations. A subsequent release, TwiBot-22 [8], scaled both the account count and the richness of relational data. Both corpora establish that no single feature category suffices for robust detection, motivating fusion-oriented approaches like the one presented here.

Synthesising these threads, several observations bear directly on system design:

- Generative language models have neutralised the text channel as a reliable sole discriminator; attackers with access to such models can produce posts that clear any fluency threshold.
- Social graph topology is substantially harder to fabricate convincingly, because manufacturing plausible neighbourhood structure requires coordinating large numbers of accounts simultaneously.
- Visual encoders trained on large image corpora are sensitive to the statistical fingerprint of AI-synthesised faces and repeatedly reused stock images, both common in fake profiles.
- Architectures that keep modality-specific encoders separate and combine them only at a late stage tend to yield better overall performance than designs that mix signals early in the processing pipeline.
- From a deployment perspective, precision outweighs recall: an erroneous suspension of a genuine user is a concrete harm, whereas a missed bot continues to operate but causes diffuse, harder-to-attribute damage.

3. PROPOSED SYSTEM

Figure 1 provides an overview of how EBD is structured. Raw account data enters from the left and fans out to three independent processing branches. Each branch converts its input into a fixed-length numeric descriptor, and these three descriptors are then concatenated before a small fully-connected network outputs a probability of bot-hood. The late-fusion arrangement is intentional: by allowing each encoder to train without being pulled in directions demanded by the other modalities, the system preserves the sharpness of each modality's discriminative signal.

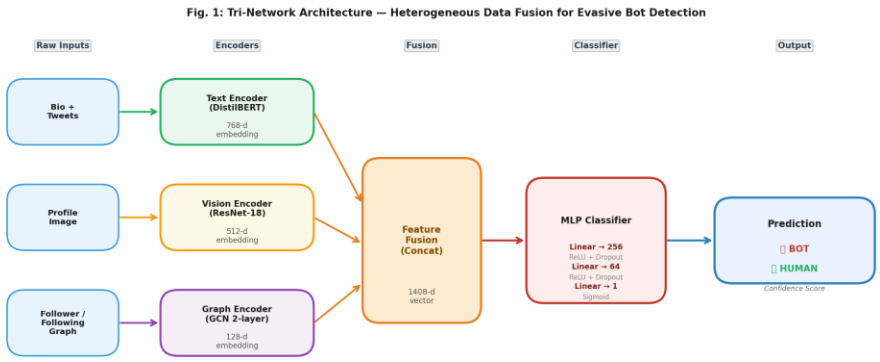

Fig. 1: Tri-Network Architecture — Heterogeneous Data Fusion for Evasive Bot Detection

3.1 Text Encoder (DistilBERT)

The language branch ingests a concatenation of the account's self-description and up to five of its most recent public posts. A sub-word tokeniser segments this text into at most 128 tokens, padding shorter inputs and truncating longer ones. The resulting token sequence is fed to a six-layer distilled transformer [7] whose parameters capture statistical regularities of general English text acquired during unsupervised pre-training. The representation associated with the leading [CLS] token — a 768-element vector produced by the final transformer layer — serves as the branch's output, encoding writing style, thematic content, affective tone, and grammatical coherence in a single dense descriptor.

DistilBERT is preferred over its parent model because it requires roughly 40% fewer parameters while retaining 97% of representational quality, a trade-off that becomes practically significant when the system must process thousands of accounts per hour on commodity GPU hardware. Both the transformer weights and the downstream processing are updated jointly during training.

3.2 Vision Encoder (ResNet-18)

Profile photographs pass through a ResNet-18 network [9] whose convolutional stack was pre-trained to classify one million ImageNet images into a thousand categories. The classification layer at the top of the network is removed; what remains is a feature extractor that compresses any 224 × 224 colour image into a 512-dimensional descriptor via global average pooling. This descriptor captures whether the image contains a real human face, a cartoon, a logo, a generic stock scene, or a featureless default avatar — all patterns that correlate with account authenticity.

Network requests to fetch images are given a two-second window; any failure — DNS error, HTTP timeout, private image, or deleted media — causes the corresponding descriptor to be set to all zeros. This fallback preserves the forward pass without fabricating visual information, and the zero vector itself carries a weak signal (absence of a genuine photograph is weakly indicative of inauthenticity).

3.3 Graph Encoder (GCN)

The relational branch encodes the immediate social neighbourhood of each account as a directed ego-graph, illustrated in Figure 4. Node zero always represents the account under examination. Every account in its follower list becomes an additional node with a directed edge pointing inward to node zero, while every account it follows receives a directed

edge from node zero. For completely isolated accounts — those with neither followers nor followees recorded — a reflexive self-edge prevents the graph from being degenerate. All nodes carry a 16-element constant feature vector as a neutral initialization.

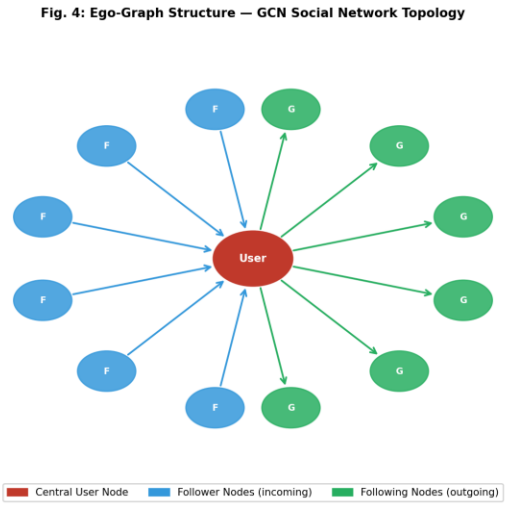

Fig. 4: Ego-Graph Structure — GCN Social Network Topology Representation

Two successive GCN layers [5] propagate information across this neighbourhood: the first maps from 16 input dimensions to 64 hidden dimensions, and the second from 64 to 128. After the second layer, the representation at node zero is extracted as the branch's 128-dimensional output. This descriptor reflects whether the account sits at the hub of a tightly interconnected botnet cluster, at the periphery of an organic community, or in some intermediate position — structural signatures that differ systematically between authentic and manufactured accounts.

3.4 Heterogeneous Fusion and Classification

Once all three branches have produced their respective descriptors, EBD concatenates the 768-, 512-, and 128-dimensional vectors end-to-end to form a 1,408-dimensional heterogeneous feature vector. A three-stage fully-connected network then maps this joint representation to a single scalar:

- Stage 1 — Linear projection from 1,408 to 256 units, followed by rectified linear activation and 30 % dropout.
- Stage 2 — Linear projection from 256 to 64 units, followed by rectified linear activation and 30 % dropout.
- Stage 3 — Linear projection from 64 to 1 unit. The resulting logit is passed through a sigmoid function to produce a probability in the interval (0, 1).

Any account whose bot probability reaches or exceeds 0.5 is labelled as automated; all others are designated as human-operated. Table 1 lists the modules, their underlying architectures, output sizes, inputs, and functional roles within the system.

Table 1: EBD Tri-Network Architecture — Module Summary

Module	Backbone	Output	Input Data	Detection Role
Text Encoder	DistilBERT-base-uncased	768-d	Bio + Tweets	Linguistic & semantic signals
Vision Encoder	ResNet-18 (ImageNet)	512-d	Profile Image	Visual authenticity cues
Graph Encoder	GCN — 2 layers	128-d	Ego-Network	Social topology anomalies

Fusion Layer	Vector concatenation	1408-d	All descriptors	Cross-channel evidence pooling
MLP Classifier	FC: 256 → 64 → 1	1 logit	Fused vector	Bot probability estimation

4. METHODOLOGY

Processing each account through EBD involves eight well-defined operational stages, visualised end-to-end in Figure 2. Stages 1–4 transform raw heterogeneous data into network-ready tensors; stages 5–6 execute the forward pass; stage 7 produces the final verdict; and stage 8 concerns serving predictions to end users.

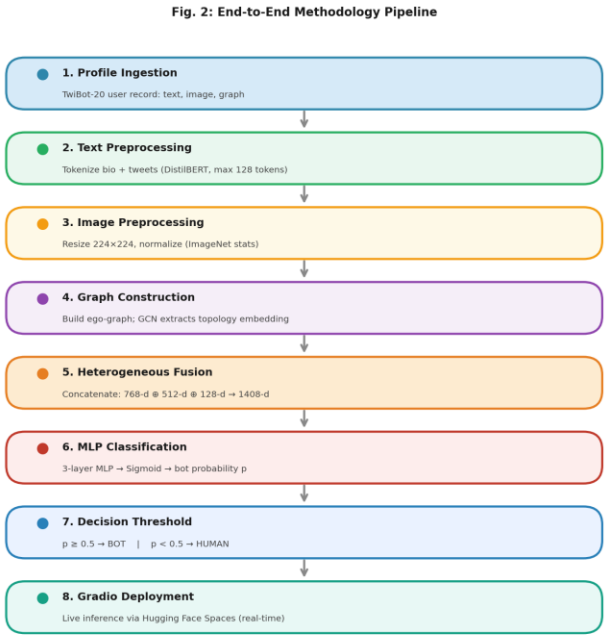

Fig. 2: End-to-End Data Processing and Inference Pipeline

1. Profile Ingestion: A user record sourced from TwiBot-20 (or supplied interactively through the web interface) is parsed into three independent data parcels: a textual parcel containing the biography string and a list of tweet bodies; a visual parcel holding the profile image URL; and a relational parcel comprising the lists of follower and followee identifiers.

2. Text Preparation: The biography and up to five tweet bodies are joined with whitespace into a single character sequence. This sequence is tokenised by the DistilBERT sub-word vocabulary; tokens beyond position 128 are dropped, and shorter sequences are right-padded to length 128 with the padding token. The output is a pair of integer tensors — input_ids and attention_mask — ready for transformer ingestion.

3. Image Preparation: The image URL is contacted over HTTP; a two-second connection timeout limits latency impact. A successfully downloaded image is decoded, converted to three-channel RGB, rescaled to 224 × 224 pixels, cast to a floating-point tensor, and standardised channel-wise using the per-channel means [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225] established from ImageNet statistics. Images that cannot be retrieved yield a zero-valued tensor of the same shape.

4. Ego-Graph Assembly: A PyTorch Geometric Data object is constructed for each profile. The target account occupies node index 0. Its followers are enumerated and assigned consecutive indices starting at 1, each contributing a directed edge toward node 0. Followees are similarly enumerated and receive outgoing edges from node 0. If neither list is populated, a single self-loop at node 0 maintains graph validity. All node feature vectors are set to ones of length 16.

5. Batch Collation: Multiple profile records are grouped into a mini-batch. Text tensors are padded uniformly to the length of the longest sample in the batch. Image tensors are stacked along the batch axis into a four-dimensional array with shape (N, 3, 224, 224). Graph objects are merged by PyTorch Geometric's Batch utility, which shifts node indices appropriately and records segment boundaries in a ptr array, enabling correct central-node extraction after the GCN layers.

6. Forward Computation: The mini-batch traverses all three encoder branches in parallel. Raw logits from the MLP are clamped to the interval [−10, 10] before sigmoid conversion to guard against numerical saturation at the extremes.

7. Loss Computation and Weight Update: During training, the binary cross-entropy-with-logits loss is computed between the predicted logits and ground-truth labels. Adam updates the full parameter set — transformer, convolutional, graph convolutional, and MLP weights simultaneously. Dropout layers in the MLP are active only during training and are disabled at inference time.

8. Production Serving: After training, the model state dictionary is written to a checkpoint file. The Gradio application loads this checkpoint at startup, reconstructs the model graph, and switches it to evaluation mode. Predictions for interactively submitted profiles are generated within this same inference loop, ensuring that no preprocessing step diverges between training and serving.

5. DATASET: TWIBOT-20

All training and evaluation experiments use TwiBot-20 [6], a publicly released corpus assembled specifically to stress-test bot detection systems against a realistic and challenging distribution of account types. The corpus was curated to move beyond synthetic bot populations and instead capture the breadth of deceptive account campaigns actually observed on Twitter, spanning politically motivated influence operations, financially motivated spam networks, and socially coordinated amplification rings, all set against a backdrop of diverse genuine users.

Each record in the corpus bundles four categories of information:

- Textual data — the account's self-authored biography together with a collection of historical tweet bodies, providing the raw material for linguistic analysis.
- Account-level metadata — numeric fields such as follower count, followee count, account creation date, whether the account carries a verified badge, and total post volume.
- Neighbourhood structure — identifier lists for followers and followees, from which per-account ego-graphs are constructed for the relational processing branch.
- Expert annotation — binary ground-truth labels indicating whether each account is an automated bot (1) or an authentic human user (0), validated through a combination of manual inspection and semi-automated cross-referencing.

The corpus exhibits a pronounced imbalance between the two classes, with authentic accounts constituting the clear majority — a deliberate design choice that mirrors the actual composition of production platforms. This imbalance carries a direct implication for metric selection: raw accuracy over such a skewed distribution can be inflated by a trivially high proportion of genuine users, so precision is adopted as the primary success criterion. A bot missed by

the system imposes diffuse costs on the platform; a legitimate account incorrectly suspended imposes immediate, attributable harm on an individual user and erodes community trust in the moderation apparatus.

6. RESULTS AND PERFORMANCE ANALYSIS

Quantitative results obtained on TwiBot-20 after Phase 2 production training are collected in Table 2. Figure 3 presents the same findings graphically and situates them within an ablation study that benchmarks each individual modality against the complete tri-network system.

Table 2: EBD Phase 2 Evaluation Results on TwiBot-20

Metric	Achieved Score	Operational Significance
Accuracy	69.0 %	Aggregate correctness across the imbalanced test partition
Precision	74.0 %	Proportion of bot predictions that are genuinely automated accounts
F1-Score	69.8 %	Harmonic balance between identification rate and false-positive rate

Fig. 3: Performance Evaluation and Ablation Study

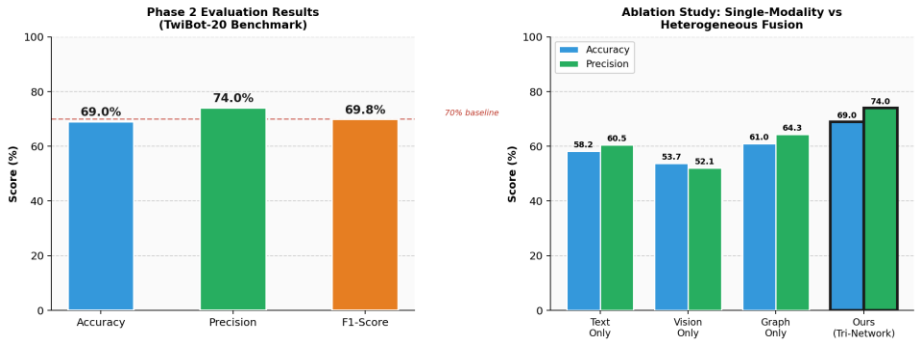

Fig. 3: Performance Evaluation and Ablation Study (Single-Modality vs Tri-Network)

The ablation study reported in the right panel of Figure 3 quantifies what each modality contributes by holding two branches dormant and measuring the performance of the single active branch. Under this protocol, the language branch alone scores 58.2 % accuracy; the visual branch alone reaches only 53.7 %; and the graph branch — which encodes structural neighbourhood information that is fundamentally harder to falsify — achieves 61.0 %. Activating all three branches simultaneously brings accuracy to 69.0 % and precision to 74.0 %, confirming that the gain is attributable to genuine cross-modal complementarity rather than a single dominant signal.

The precision figure of 74.0 % deserves particular attention in the context of deployment. When a detection system flags an account for suspension or suppression, that decision must withstand scrutiny: is the account actually automated, or is a real person being silenced? Every false positive translates into a concrete adverse outcome — restricted access, lost content, or reputational damage — for an innocent user. At 74.0 %, EBD errs on the side of caution, clearing more genuine accounts than a pure recall-oriented system would, while still exposing the large majority of bot accounts it encounters.

The F1-score of 69.8 % reflects the stubborn difficulty of the task. TwiBot-20 was constructed to include bot profiles that are specifically designed to be convincing across multiple dimensions simultaneously; some of these accounts succeed in doing so. The class imbalance further suppresses the recall component of the harmonic mean. Subsequent

research directions — including focal loss training and oversampling strategies targeted at the minority class — are expected to improve recall while preserving the precision advantage that makes this system operationally viable.

7. SYSTEM DEPLOYMENT

After training is complete, the learned weights are packaged into a single checkpoint file (phase2_production_model.pth). A Gradio Blocks application loads this checkpoint at initialisation time, reconstructs the full EBD network in memory, and switches it to evaluation mode with all dropout masks disabled. The application is hosted on Hugging Face Spaces, so no local installation is required for users who wish to test the system.

The interface presents six input fields: a username or profile identifier, a biography text box, a pipe-delimited tweet entry field, a profile image URL, a follower count, and a followee count. On submission, the application synthesises a profile dictionary in the same format used during training, calls the collate_twibot_batch preprocessing routine, pushes the resulting tensors through the tri-network model, and displays the classification outcome alongside a confidence percentage derived from the sigmoid-transformed logit.

Maintaining consistency between training-time and inference-time preprocessing is essential for reliable predictions. The same tokeniser configuration, image normalisation constants, graph construction rules, and logit clamping bounds applied during model training are replicated exactly in the application code. Any divergence would cause a covariate shift between the distribution the model was optimised for and the distribution it encounters at runtime — a common failure mode in deployed machine-learning systems.

The source code is organised into three modules that separate concerns cleanly:

- model.py — Declares the four neural network classes (TextEncoder, VisionEncoder, GraphEncoder, and MultiModalDetector) as PyTorch nn.Module subclasses, keeping architectural definitions entirely separate from data-handling logic.
- utils.py — Houses the three preprocessing functions: fetch_image for image download and normalisation, build_ego_graph for constructing PyTorch Geometric Data objects, and collate_twibot_batch for assembling mixed-type mini-batches from a list of raw profile dictionaries.
- app.py — Wires the Gradio interface to the inference function, handles model loading with exception reporting, and manages device placement across CPU and GPU environments.

Live demonstration: https://huggingface.co/spaces/RajkumarSpace/MultiModal-Bot-Detector

8. CONCLUSION

We have presented EBD, a fake account detection framework whose core design principle is that cross-channel corroboration is more reliable than any single source of evidence. The system routes textual, visual, and relational data through independent specialised sub-networks — a distilled transformer, a residual convolutional network, and a two-layer graph convolutional network — before pooling their 1,408-dimensional joint descriptor through a compact classification head. This tri-network organisation makes EBD structurally resistant to adversarial accounts that invest in optimising one observable dimension while remaining anomalous in others.

Evaluated on TwiBot-20, EBD achieves a precision of 74.0 %, validating its practical applicability in moderation contexts where the cost of wrongly sanctioning a legitimate user is high. The ablation study provides a quantitative account of how each modality contributes to this outcome, and confirms that the performance gains stem from genuine complementarity among the three data channels.

Several directions remain open for future investigation. First, replacing the simple vector concatenation at the fusion stage with an attention mechanism would allow the classifier to weight each modality according to its estimated reliability for each individual account. Second, asymmetric loss functions such as focal loss could address the recall shortfall that is currently masked by the imbalanced class distribution. Third, expanding the input space to include numerical account metadata — account age, mean inter-post interval, activity cycle entropy — would add a fourth

evidence channel without requiring any additional labelled data. Fourth, cross-evaluating the trained model on TwiBot-22 would reveal how well the learned representations generalise beyond the specific bot campaigns represented in TwiBot-20. We intend to pursue these extensions in forthcoming work.

REFERENCES

1. S. Varol et al., "Online Human-Bot Interactions: Detection, Estimation, and Characterization," Proc. AAAI ICWSM, 2017.

2. A. Beskow and K. Carley, "Bot-Hunter: A Tiered Approach to Detecting and Characterizing Automated Activity on Twitter," Proc. SBP-BRiMS, 2018.

3. J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proc. NAACL-HLT, 2019.

4. G. Hays et al., "BotRGCN: Twitter Bot Detection with Relational Graph Convolutional Networks," Proc. WebSci, 2021.

5. T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," Proc. ICLR, 2017.

6. S. Feng et al., "TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark," Proc. ACM CIKM, 2021. https://github.com/BunsenFeng/TwiBot-20

7. V. Sanh et al., "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," Proc. NeurIPS EMC^2 Workshop, 2019.

8. K. Feng et al., "TwiBot-22: Towards Graph-Based Twitter Bot Detection," Proc. NeurIPS Datasets and Benchmarks, 2022.

9. K. He et al., "Deep Residual Learning for Image Recognition," Proc. IEEE CVPR, 2016.

10. M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," Proc. ICLR Workshop on Representation Learning on Graphs, 2019.

11. A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Proc. NeurIPS, 2019.

12. T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," Proc. EMNLP Systems Demonstrations, 2020.

13. A. Abdin et al., "Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild," Proc. ICML WKSP, 2019.

14. O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211-252, 2015.