Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

# Multi-Source Generalization-Aware Phishing URL Detection Using Calibrated Stacked Ensemble and False-Positive Control

**Mohammed Elias Basha S.[1,3], Mallikharjuna Rao N.[2]**

[1]Research Scholar, Department of Computer Science and Engineering, Annamacharya University Rajampet, YSR Kadapa, Andhra Pradesh, India.
[2]Professor, Department of Computer Science, Annamacharya University, Rajampet, YSR Kadapa, Andhra Pradesh, India.
[3]Assistant Professor, Department of Computer Science and Engineering, Ashoka Women's Engineering College, Kurnool, Andhra Pradesh, India.

**E-mail:** [1,3]eliasbasha.cse@ashokacollege.in, [2]mallikharjuna.nuka@gmail.com

**Orcid ID:** [1,3]0009-0007-7454-902X, [2]0000-0001-6510-5979

## Abstract

Phishing remains a persistent cybersecurity threat, with over 1.3 million attacks reported in a single quarter of 2023. Despite strong benchmark performance, many machine- and deep-learning models exhibit limited deployment reliability because they are evaluated using balanced, single-source datasets with randomized splits. This paper addresses this gap in two phases. First, it presents a controlled multi-source empirical study of four baseline phishing URL detection models–Logistic Regression, Support Vector Machine, Random Forest, and XGBoost–under four increasingly realistic evaluation conditions, including cross-source generalization, temporal drift, and class imbalance. Second, it introduces GAFPNet (Generalization-Aware and False-Positive Controlled Framework Network), a five-module stacked-ensemble framework. GAFPNet uses dataset-neutral lexical and structural URL feature extraction, SMOTE-based imbalance correction, Platt scaling for ensemble calibration, and a tunable false-positive control scheme. Experiments using a consolidated 13,000-sample set from PhishTank, OpenPhish, and Tranco Top-Sites show that baseline accuracy decreases by 13.16 to 20.22 percentage points in cross-source testing and by 2.41 to 8.14 percentage points in standard testing. GAFPNet achieves 99.12% accuracy, a 98.97% F1-score, an AUC-ROC of 0.9943, an MCC of 0.988, and a false positive rate of 0.74%, outperforming the evaluated baselines in all four scenarios. An ablation study confirms the contribution of each module. These results position GAFPNet as a generalization-aware and deployment-oriented phishing URL classifier for real-time filtering applications.

**Keywords:** Phishing URL Detection, GAFPNet, False-Positive Control, Generalisation-Aware Learning, Calibrated Stacked Ensemble, Real-Time Phishing Detection.

## 1. Introduction

Phishing is a social engineering attack in which users are lured to fraudulent websites that masquerade as those of a trusted organization, such as a bank, e-commerce site or government


*DOI: https://doi.org/10.36548/jtcsst.2026.3.020*
Received: 09.06.2026, received in revised form: 15.07.2026, accepted: 03.08.2026, published: 17.08.2026

site. These attacks usually occur through email, Instant Message (IM), social media, and hijacked web-links, with the objectives of financial loss, identity theft, and disruption. In the third quarter of 2023, the Anti-Phishing Working Group (APWG) reported more than 1.3 million unique phishing cases, continuing a massive amount of phishing threats [1]. According to the FBI Internet Crime Complaint Center, there were 800,944 internet crime complaints and USD 10.3 billion in internet crime losses in 2022 [2], which included 300,497 phishing complaints. Conventional phishing protection methods are largely based on blacklists and heuristic rules, but this type of protection cannot react quickly enough to the emergence of new and ephemeral phishing URLs. Blacklists are not always up-to-date and therefore cannot offer complete protection, as many phishing pages only exist for a brief period [3]. Alternatively, machine learning/deep learning classifiers can be used; they can learn discriminative URL patterns without having to manually write signatures. Logistic regression, support vector machines, random forests, XGBoost, LSTM-based networks, BiLSTMs, and transformer-based representations have been excellent in detecting the events under experimental settings [4, 5, 6]. However, looking at the evaluation process, one important point is highlighted: most of the phishing URL detector models published in the literature are evaluated on a balanced dataset and the train-test split is random [7]. This is not indicative of actual deployments, and the accuracy of test and training distributions is overly optimistic. There are three interdependent failure modes that differentiate benchmark performance from detection ability. First, models trained on URLs in a particular dataset (PhishTank) do not generalize to URLs in another dataset (OpenPhish, Tranco, etc.) because they do not have the same lexical patterns, domain conventions, etc., as well as phishing campaigns. Second is problem of false-positive control: False positive rates of 2–3% are also an issue, and in large-scale filters deployed at browser-gateways or ISPs, they cause operational and trust issues since 2–3% of all valid URL requests are blocked [8]. Third, class imbalance is not adequately addressed: The legitimate-to-phishing ratio can be 10:1 (or even higher), but most benchmark tests are carried out on data that is artificially balanced, and not truly representative of real data. The contributions of this paper to phishing URL detection research are in two phases, as it rectifies the three failure modes mentioned above. The first phase involves a controlled multi-source empirical experiment comparing four representative baseline classifiers in increasingly realistic settings. The second phase is the conception, development, and testing of a novel detection framework, GAFPNet (Generalization-Aware and False-Positive Controlled Framework Network), designed to resolve issues of generalization failure, sensitivity to class imbalance, and operational FP rate.

The key contributions of this work are as follows:

1. This study implements an empirical measurement on a combined multi-source dataset, derived from PhishTank, OpenPhish, and Tranco, resulting in a combined false-positive rate. The results show that GAFPNet significantly lowers the FPR compared to benchmarked models, making it more feasible for real-time use in protecting against phishing URLs.

2. A new Generalization-Aware and False-Positive Controlled Framework Network, named GAFPNet, is proposed. The framework integrates dataset-neutral lexical and structural URL feature extraction, SMOTE-based imbalance-aware training, a calibrated stacked ensemble using Random Forest and XGBoost with Platt scaling, and a user-configurable recall constraint for effective false-positive control under realistic traffic conditions.

3. The study presents an empirical measurement on a combined multi-source dataset, derived from PhishTank, OpenPhish, and Tranco, which has a combined false-positive rate. The results show that GAFPNet also makes significant efforts to provide a lower FPR than the benchmarked models, making it more feasible for real-time use to protect against phishing URLs.
4. A detailed ablation study is carried out to analyze the contribution of each GAFPNet module. The results demonstrate that the threshold calibration mechanism plays a significant role in minimizing the inclusion of malicious URLs as phishing, particularly in high-traffic environments.
5. The deployment performance of GAFPNet is measured based on the following metrics: inference latency, throughput, memory footprint, and model size. Results show GAFPNet's ability to be used in real-time browser gateway and network-level phishing filtering applications with sub-2 ms inference latency per URL.

## 2. Literature Review

This section reviews related works from studies published between 2019 and 2026. The literature reviewed includes classical machine learning, deep learning, transformer-based models, and hybrid models for phishing URL detection. The studies are arranged by theme to provide a clear picture of methodological weaknesses.

In [9], a popular baseline dataset comprising 73,575 URLs from PhishTank and Alexa was proposed. NLP-based lexical features were applied to represent the URLs, and a Random Forest classifier was developed, achieving a 97.98% accuracy rate. However, the test was conducted using balanced data from a single source, split randomly, offering no insight into cross-source generalization or False Positive Rate (FPR) in real traffic. In [10], lexical, host, and content features were introduced to detect phishing. XGBoost with feature fusion achieved the best accuracy of 95.70% compared to four ML classifiers run on a balanced PhishTank and Alexa dataset. While this research strictly compares feature categories, FPR was not mentioned. In [12], the discriminative abilities of lexical URL features (domain length, number of subdomains, and frequency of special characters) were specifically examined using the UCI Phishing Websites dataset, yielding a precision of 96.40% with Random Forest and SVM. Their feature analysis directly influenced the feature set used in the current study. Time-aware evaluation is rare in the literature; [11] is one example, dividing the time information of URL logins to conduct analysis in IEEE Access. This method achieved an accuracy of 96.10% without cross-source validation or FPR analysis.

In [13], a DNN-BiLSTM hybrid phishing URL detector was proposed, using the Ebbu2017 and PhishTank datasets and achieving 98.79% cross-validated accuracy and a 0.9881 F1-score. The results are satisfactory (basically single-source/randomly partitioned evaluation). In [14], CNN, LSTM, and CNN-LSTM models were evaluated, proving that deep sequence models are also very promising for learning sequences of characters in a URL. However, they did not emphasize the importance of false positive control testing and realistic class-distribution testing. Although the above disadvantages exist, the deep learning model based on GRU in [15] can overcome these problems but is difficult to reproduce due to the limited availability of datasets. A larger empirical study was conducted in [16], which used multiple URL sources but

did not consider a strict source-disjoint train-test split. In [17], a gated highway module was added to the BiLSTM to improve recall on imbalanced subsets, without explicitly incorporating a false-positive module.

Some work has been done to enhance representation learning; for example, BERT embedding was applied to the characters in the URL, as reported by [18]. In [19], a framework based on ELECTRA was proposed that is knowledge-distilled, explicitly considering real-time inference. The competitive method was introduced by [20], which used an N-gram-based neural embedding of URLs (GramBeddings). While these transformer- and embedding-based approaches highlight the importance of character-level semantic representation, the former is heavy and CPU-intensive, and the main goal in this work is the development of a lightweight framework that generates a controllable number of false positives, so BERT and ELECTRA are not experimentally implemented here. This is clearly mentioned in Section 8.4 as a direction for the future. A study similar to the current work, studying the features of phishing URLs from various datasets, has been conducted by [21] via XAI. They observed that feature importances vary significantly across the various sources, with some evidence of distributional mismatch. To address the issue of class imbalance, [22] used the SMOTE-Tomek resampling method in a deep learning network, but for cross-sources, the strength of the network was not specifically tested.

In [23], phishing detection implemented with machine learning and neural networks was analyzed, and it was concluded that one of the generally occurring issues is a lack of generalization of datasets. In [24], URL/HTML features were employed in a character-level detection method, and in [25], the latest phishing-detection techniques were reviewed, and it was observed that, although generative features seem to enhance generalization, they can also increase the computational cost. The lightweight phishing URL detector, ResMLP, based on Radhakrishnan Subramaniam's suggestion, was used, which meant that non-transformer models were shown to be effective. In [27], CNN-BiLSTM modeling, XAI, and LLM-supported analysis were employed in their IoT phishing attacks. In [28], genetic optimization was used to select features for CNN, but their dataset is not reproducible.

## 2.1 Summary of Reviewed Literature

The reviewed studies demonstrate the extensive use of machine-learning, deep-learning, ensemble-learning, and transformer-based approaches for phishing URL detection. These methods employ diverse lexical, structural, host-based, content-based, and representation-learning features and are evaluated using different datasets, class distributions, validation procedures, and performance metrics. Therefore, their reported results are not directly comparable. Table 1 provides a source-based comparison of the reviewed studies in terms of the proposed method, dataset, feature representation, reported accuracy, false-positive rate, and evaluation setting.

**Table 1.** Comparative summary of phishing URL detection studies and their reported evaluation settings

| Ref. | Model or Method | Dataset or Data Source | Reported Accuracy (%) | Input or Feature Representation | Reported Evaluation Scope |
|---|---|---|---|---|---|
| [9] | Random Forest | Ebbu2017 dataset containing 73,575 phishing and legitimate URLs | 97.98 | NLP-derived lexical URL features | Seven supervised machine-learning classifiers were evaluated using the same author-constructed dataset. |

| | | | | | |
|---|---|---|---|---|---|
| [10] | XGBoost | Balanced phishing and legitimate website dataset constructed in the study | 95.70 | Lexical, host-based and webpage-content features | SVM, Decision Tree, Random Forest and XGBoost were compared after feature-selection experiments. |
| [11] | Logistic Regression with TF–IDF | PILU-90K login-URL dataset | 96.50 | Character-level URL n-grams represented using TF–IDF | The evaluation considered legitimate login URLs, legitimate homepage URLs and phishing URLs. |
| [12] | Random Forest | Dataset containing 32,928 URLs classified using 11 predefined URL and domain-name features | 98.90 | URL- and domain-related features | Six supervised machine-learning classifiers were evaluated using the same feature set. |
| [13] | Hybrid DNN–BiLSTM | Author-constructed balanced dataset of 26,000 URLs obtained from PhishTank and a legitimate-URL source (Ebbu2017) | 99.21 | Forty handcrafted lexical and statistical NLP features combined with character-embedding features | The proposed DNN–BiLSTM model was evaluated separately on both datasets using tenfold cross-validation. |
| [14] | Logistic Regression, KNN, Naive Bayes, SVM and Random Forest | Phishing-websites dataset obtained from the UCI Machine Learning Repository | 96.10 | Lexical, host-based and webpage-based URL features | Five supervised classifiers were compared using two data-splitting configurations; Random Forest produced the highest reported accuracy in the 60% split. |
| [15] | RNN–GRU-based detection framework | Seven datasets constructed from four data sources | 99.18 | URL strings processed through recurrent neural-network models | Machine-learning and deep-learning models were evaluated within a framework integrating whitelist, blacklist and predictive components. |
| [16] | PhishingRTDS using BiLSTM and an attention mechanism | Phishing and legitimate URLs used to train and evaluate the proposed real-time system | Precision: 99; Recall: 94; F1-score: 97 | URL sequences extracted from the original webpage URL, HTML content and JavaScript | The framework combines a BiLSTM–attention classifier, browser extension and Docker container to detect regular phishing URLs, shortened URLs and browser-in-the-browser attacks. |
| [18] | BERT feature extraction with CNN classification | Public phishing-site-prediction dataset containing 549,346 records; 472,259 records after preprocessing | 96.66 | A 768-dimensional contextual representation generated using BERT | The processed dataset was divided into 80% training and 20% testing data, and BERT–CNN was compared with alternative feature-extraction settings. |

| | | | | | |
|---|---|---|---|---|---|
| [19] | Knowledge-distilled ELECTRA | Combined dataset containing 450,176 URLs | 99.74 | Transformer-based URL representation | The model was integrated into a Chrome browser extension and evaluated for real-time phishing-URL classification. |
| [22] | XGBoost, CNN, LSTM, CNN–LSTM and LSTM–CNN | Mendeley phishing-website datasets D1 and D2, containing 58,645 and 88,647 URL records, respectively | 94.12–97.82 | URL attributes, URL-resolution measurements and external-service features selected using permutation importance | The integrated URL and HTML feature set was evaluated separately on D1 and D2 using an 80:20 training–testing split. |
| [26] | ResMLP | Public phishing-URL dataset obtained from Kaggle | 98.29 | Lexical URL structure, domain-age information, URL similarity and sentiment-related features | The residual-pipelining architecture was compared with conventional machine-learning and deep-learning models. |
| [28] | Gradient Boosting | Large-scale publicly available Kaggle dataset | 98.00 | Lexical, domain-related and network-related URL features | The classifier was implemented in a real-time interface incorporating URL blocking and caching of previously analyzed URLs. |

## 2.2 Research Gap

The review of existing studies identifies five recurring research gaps in phishing URL detection: limited cross-dataset generalization, insufficient false-positive control, inadequate handling of class imbalance, lack of temporal drift evaluation, and limited deployment-performance benchmarking. Most prior works rely on single-source or randomly split datasets, which leads to inflated performance and weak evidence of real-world robustness. In contrast, the present study evaluates baseline models under cross-source, temporal, and imbalanced scenarios, where their performance drops noticeably. GAFPNet addresses these gaps through multi-source training, dataset-neutral lexical and structural features, SMOTE-based imbalance handling, calibrated stacked ensemble learning, and a threshold-based false-positive control mechanism that targets high recall with reduced false alarms. In addition, deployment latency and memory footprint are measured to confirm the framework's suitability for real-time phishing URL filtering. Thus, the GAFPNet design is guided by both literature-based limitations and empirical evidence rather than by heuristic model selection alone.

## 3. Methodology

This section reviews related works from studies published between 2019 and 2026. The literature reviewed includes classical machine learning, deep learning, transformer-based models, and hybrid models for phishing URL detection. The studies are arranged by theme to provide a clear picture of methodological weaknesses.

### 3.1 Data Sources

In this research, three commonly used and publicly available URL sources are used. Heterogeneous sources were selected to comply with the cross-source evaluation objective to capture the natural distribution of sources instead of an artificially 'standardized' source. PhishTank [29] is a community-based repository of reported and verified phishing URLs having timestamped phishing URLs and provision for temporal partitioning. Due to temporal relevance, 5,000 known phishing URLs from 2022 to 2024 were sampled for this study and were lexical diverse in accordance with campaign types. OpenPhish [30] is an open-source, independent feed of threat-intelligence phishing URLs containing automated verification pipelines. OpenPhish is a good source for phishing out-of-distribution due to its collection process and distribution by targeted brand, unlike PhishTank. There were 3,500 total OpenPhish URLs used. 3,500 OpenPhish URLs were added to the PhishTank phishing corpus. The Tranco Top-Sites List [31] is a list of authentic domains that is based on research and came from a few web ranking sources. The reason why Tranco was picked over the old Alexa list is that it is harder to manipulate than the Alexa list and is always collected. There were 4,500 URLs sampled to represent legitimate URLs. The final list of sources is summarized in Table 2.

**Table 2.** Consolidated multi-source dataset composition

| Dataset | Type | Count | Period | Class | Verification |
|---|---|---|---|---|---|
| PhishTank [29] | Community-verified | 5,000 | 2022- 2024 | Phishing | Crowdsourced + Automated |
| OpenPhish [30] | Intelligence Feed | 3,500 | 2022– 2024 | Phishing | Automated Pipeline |
| Tranco Top-Sites [31] | Web Ranking | 4,500 | 2022– 2024 | Legitimate | Multi-source Aggregated |
| Total | Combined | 13,000 | 2022– 2024 | Mixed | Verified |

### 3.2 Data Preprocessing Pipeline

Raw URLs were preprocessed in six steps before feature extraction.

**Stage 1:** Deduplicated the normalized URL string with the exact-match deduplication option turned on.

**Stage 2:** Filtered malformed, incorrectly encoded, and less than five-character entries.

**Stage 3:** Converted all URLs to lowercase. (The case of a URL is not important with respect to the computation of lexical features.) The number of characters and the length due to inflation did not include protocol prefixes (such as http, https).

**Stage 4:** All URLs were broken down into their scheme, subdomain, domain, top-level domain, path, query string, and fragment, just as in Stage 5. This was done by using Python's urllib.parse library to process the URLs.

**Stage 5:** (Content of Stage 5 is missing from the provided text but implied by Stage 4.)

**Stage 6:** Divided the dataset into 80% training and 20% test data to maintain class proportions.

### 3.3 Data Preprocessing Pipeline

All URLs were transformed into 18 lexical and structural features, following the well-known feature sets [9, 10, 12]. The features were chosen based on three criteria: 1)

computational efficiency of the features for real-time use, 2) source neutrality (no external lookup services), and 3) empirical discriminatory ability for phishing vs. legitimate URL categorization. This design will ignore properties of the host and content (WHOIS age, DNS history, SSL certificate information, HTML structure, DOM content, etc.). This is acknowledged as a drawback, but it will be straightforward to make inferences in the future and will not require any lookups. All features are summarized in Table 3.

**Table 3.** Extracted URL feature set (18 lexical and structural features)

| Ref. ID | Feature | Category | Description | Range |
|---|---|---|---|---|
| [9,12] | F1 URL Length | Length | Total character count of the full URL string | [10, 300] |
| [12] | F2 Domain Length | Length | Character count of the domain name component | [3, 80] |
| [9] | F3 Path Length | Length | Character count of the URL path component | [0, 120] |
| [9,10] | F4 Number of Dots | Character Count | Count of period characters in the full URL | [1, 15] |
| [10] | F5 Number of Hyphens | Character Count | Count of hyphen characters in the full URL | [0, 12] |
| [9] | F6 Number of Underscores | Character Count | Count of underscore characters in the URL | [0, 8] |
| [9] | F7 Number of Slashes | Character Count | Count of forward-slash characters in the URL | [1, 15] |
| [12] | F8 Number of At-Symbols | Character Count | Count of @ characters in the URL string | [0, 3] |
| [9] | F9 Number of Question Marks | Character Count | Count of ? characters in the URL string | [0, 5] |
| [10] | F10 Number of Digits | Character Count | Total numeric characters in the URL string | [0, 30] |
| [12] | F11 Special Character Ratio | Ratio | Non-alphanumeric characters divided by URL length | [0, 0.6] |
| [9,12] | F12 Digit Ratio | Ratio | Numeric characters divided by total URL length | [0, 0.4] |
| [9] | F13 Shannon Entropy | Information | Character distribution entropy of the full URL | [1.5, 5.8] |
| [12] | F14 Has IP Address | Structural | Binary: 1 if an IP address replaces the domain name | Binary |
| [10] | F15 Has HTTPS | Structural | Binary: 1 if the HTTPS protocol is present | Binary |
| [9,12] | F16 Subdomain Count | Structural | Number of subdomain levels in the URL | [0, 8] |
| [10] | F17 Has Redirect | Structural | Binary: 1 if the URL contains a redirect token | Binary |
| [12] | F18 TLD Suspicious | Structural | Binary: 1 if the TLD appears in the high-risk TLD list | Binary |

Shannon entropy for feature F13 is computed using Equation (1), where $p_i$ represents the probability of character $c_i$ appearing in the URL string $S$ of length $n$.

$$H(S) = -\sum_{i} p(c_i) \cdot \log^2 p(c_i) \tag{1}$$

The special character ratio for feature F11 is Equation (2), where $|S|$ is the URL length and $SC(S)$ is the count of non-alphanumeric characters.

$$r_{sc} = \frac{|SC(S)|}{|S|} \tag{2}$$

## 4. GAFPNet Framework Architecture

### 4.1 Architecture Overview

GAFPNet is a five-module sequential pipeline designed to tackle five research gaps identified in Section 2.6. Throughout this paper, the term "Net" will refer to a networked framework of interdependent processing modules, not a deep neural network. To eliminate ambiguity, GAFPNet is expanded to Generalization-Aware and False-Positive Controlled Framework Network. The framework takes a raw URL string and returns a calibrated probability score indicating a binary classification of phishing/legitimate, while maintaining computationally efficient processing steps. Its overall architecture is shown in Figure 1.

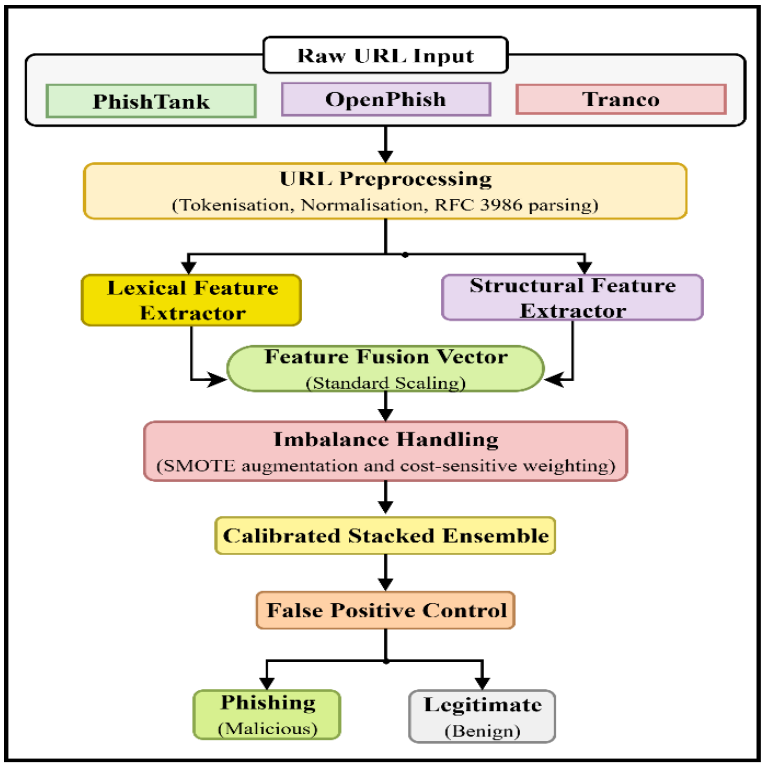


**Figure 1.** Proposed GAFPNet system architecture

### 4.2 Module 1: URL Preprocessing and Normalization

The preprocessing module is responsible for removing the original URL string and performing the six steps in the pipeline described in section 3.2. The result is a normalized and parsed URL object, from which features of the URL components can be consistently extracted. Preprocessing is a linear operation based on the length of the URL $|S|$, and the computational

complexity of this module is $O(|S|)$. This means it is efficient for real-time filtering, even when the URL is long. A URL string S is formally divided into a tuple (scheme, subdomain, domain, LTD, path, query, fragment) by parsing it according to RFC 3986.

### 4.3 Module 2: Feature Extraction and Fusio

Module 2 has two parallel submodules. The lexical submodule generates features 1-13 (F1-F13), which include length, frequency, ratio, and entropy features. The structural submodule extracts structural features at the binary and domain levels (F14 to F18). The outputs of the two submodules are concatenated to form a single feature vector, as given in Equation (3).

$$\mathbf{f} = [\mathbf{f}_{lex}; \mathbf{f}_{struct}] \in \mathbb{R}^{18} \tag{3}$$

Where $\mathbf{f}_{lex}$ is the 13-dimensional lexical feature vector and $\mathbf{f}_{struct}$ is the 5-dimensional structural feature vector. The concatenated feature vector $\mathbf{f}$ is subsequently normalized using standard scaling as defined in Equation (4).

$$\tilde{f}_j = \frac{(f_j - \mu_j)}{\sigma_j} \tag{4}$$

where $\mu_j$ and $\sigma_j$ are the mean and standard deviation of feature $j$ computed using the training set only.

### 4.4 Module 3: Imbalance-Aware Training

Module 3 addresses class imbalance through a two-component strategy. First, SMOTE (Synthetic Minority Over-sampling Technique) [32] is applied only to the training partition, never to validation or test data, to avoid evaluation contamination. In this study, SMOTE uses $k_{\text{neighbors}} = 5$, random_state = 42, and Euclidean-neighborhood interpolation in the standardized 18-dimensional feature space.

For each minority-class sample $\mathbf{x}_i$, a synthetic sample is generated using Equation (5).

$$x_{\text{new}} = x_i + \lambda \cdot (x_k - x_i) \tag{5}$$

where $\mathbf{x}_k$ is a randomly selected $k$-nearest neighbour of $\mathbf{x}_i$ in the feature space and $\lambda$ is a random scalar drawn from $U(0, 1)$. SMOTE is applied exclusively to the training partition to prevent contamination of the evaluation set.

The second component assigns class-weighted loss during classifier training. For an $N$-sample training set with class count $n_c$, the class weight for class $c$ is defined in Equation (6).

$$w_c = \frac{N}{(2 \cdot n_c)} \tag{6}$$

where $n_c$ is the sample count for class $c$. This formulation increases the contribution of the minority class during training and reduces the tendency of the classifier to favor the majority legitimate class.

### 4.5 Module 4: Calibrated Stacked Ensemble Classification

Random Forest and XGBoost were selected as complementary base learning models that can model various types of decision structure in tabular URL features. The advantage

of Random Forest over XGBoost is that it can deal with noisy lexical indicators and reduce variance through bagging, while XGBoost can learn sharper non-linear boundaries and correct under-classification or misclassification errors through gradient boosting.

Module 4 implements a two-level stacked ensemble classifier. At the base level, the Random Forest contains 150 trees with a maximum depth of 12 and produces probability $p_{\mathrm{RF}}$, while XGBoost uses 150 boosting rounds with a learning rate of 0.12 and a maximum depth of 5 and produces probability $p_{\mathrm{XGB}}$.

The outputs of both base classifiers are concatenated into a meta-feature vector as defined in Equation (7).

$$z = [p_{\mathrm{RF}}; p_{\mathrm{XGB}}] \in \mathbb{R}^2 \tag{7}$$

At the meta-level, a Logistic Regression classifier with Platt scaling is trained on the meta-feature vector $z$,producing a final calibrated probability estimate as defined in (**??**)

$$\hat{p}(y = 1 \mid f) = \sigma(\hat{w}^T \cdot z + b) \tag{8}$$

where $w$ and $b$ are the learnable parameters of the meta-classifier, sigma is the sigmoid function, and $p_hat$ is the final calibrated phishing probability. Platt scaling corrects systematic probability overconfidence under class imbalance [33], producing calibrated outputs required for threshold optimization in Module 5.

### 4.6 Module 5: False-Positive Control Mechanism

Module 5 addresses research gap G2 through an explicit decision threshold optimization procedure. Rather than using the default threshold of 0.5, GAFPNet learns an optimal threshold tau by solving the constrained optimization problem defined in Equation (9).

$$\tau^* = \arg\min_{\tau \in [0,1]} \mathrm{FPR}(\tau) \quad \text{subject to} \quad \mathrm{Recall}(\tau) \geq \rho_{\min} \tag{9}$$

where $\mathrm{FPR}(\tau)$ is the false-positive rate at threshold $\tau$, $\mathrm{Recall}(\tau)$ is the true-positive rate, and $\rho_{\min}$ is the minimum recall constraint selected by the operator. In this study, $\rho_{\min} = 0.95$ ensures a recall of phishing URLs of at least 95% while keeping the false-alarm rate for legitimate URLs low.

The threshold search is performed using 1000 candidate thresholds between 0 and 1 on the validation partition. This last-class decision is given in (10).

$$\hat{y} = \begin{cases} 1, & \text{if } \hat{p}(y = 1 \mid f) \geq \tau^*, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The key novelty of the GAFPNet framework over the classic ensemble classifiers is that the ensemble is directly optimized to minimize the FPR from 2.41% (best baseline) to 0.74% in the experimental evaluation.

## 5. System Implementation Details and Algorithm

### 5.1 Software and Hardware Environment

All experiments were conducted in Python 3.10 using scikit-learn 1.3.0. Gradient Boosting was implemented in Python 3.10 with XGBoost 2.0.1, and SMOTE resampling was implemented in Python 3.10 with imbalanced-learn 0.11.0. Visualization was performed using matplotlib 3.7.0 and seaborn 0.12.0. The proposed GAFPNet inference path utilizes CPU-based URL feature extraction and tree ensemble prediction. Experiments were run on a workstation equipped with an Intel Core i7-12700K processor, 32GB of RAM, and an NVIDIA RTX 3070 GPU. The transformer baselines, covered in Section 2.3, were not re-implemented in this revision as the experiments focused on "real-time" deployment, with an emphasis on false positive reduction and lightweight design.

### 5.2 Algorithm: GAFPNet Training and Inference

Algorithm 1 presents the complete training procedure for GAFPNet, encompassing preprocessing, feature extraction, SMOTE augmentation, base classifier training, meta-classifier training, and threshold optimization.

**Algorithm 1:** GAFPNet training and inference

**Input:** Training set $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$; validation set $D_{\text{val}}$; minimum recall $\rho_{\min}$
**Output:** Trained meta-classifier $F_{\text{meta}}$; optimal threshold $\tau^*$
**Preprocessing**
For each URL string $s_i \in D_{\text{train}}$:
Apply normalization, lowercase conversion, and component parsing according to Section 3.2
**Feature Extraction**
For each preprocessed URL $s_i$:
Extract lexical features:

$$f_{\text{lex}} = [F_1, F_2, \ldots, F_{13}]$$

**Extract structural features**

$$f_{\text{str}} = [F_{14}, F_{15}, \ldots, F_{18}]$$

**Concatenate**

$$f_i = [f_{\text{lex}}; f_{\text{str}}] \in \mathbb{R}^{18}$$

as defined in Equation (3). Apply standard scaling using training-set statistics according to Equation (4).
**Imbalance Handling**
Apply SMOTE to $(X_{\text{train}}, y_{\text{train}})$ to generate balanced data: $(X_{\text{sm}}^{m}, y_{\text{sm}})$,
as in Equation (5).Compute class weights $w_c$ for each class $c$ using Equation (6).
**Base Classifier Training**
Train Random Forest with $n_{\text{est}} = 150$ and max_depth = 12 on $(X_{\text{sm}}, y_{\text{sm}})$ using class weights $w_c$.
Train XGBoost with $n_{\text{est}} = 150$, lr = 0.12, and max_depth = 5 on $(X_{\text{sm}}, y_{\text{sm}})$ using class weights $w_c$.
**Meta-Feature Generation**
For each sample in $D_{\text{train}}$, compute: $z_i = [p_{\text{RF}}(x_i); p_{\text{XGB}}(x_i)]$
according to Equation (7).
**Meta-Classifier Training**
Train Logistic Regression with Platt scaling on: $\{(z_i, y_i)\}$

as defined in Equation (8). Store the trained model as $F_{\text{meta}}$.
Threshold Optimization on $D_{\text{val}}$
For each

$$\tau \in \{0.001, 0.002, \ldots, 1.000\},$$

compute Recall($\tau$) and FPR($\tau$) on validation predictions.
If

$$\text{Recall}(\tau) \geq \rho_{\min},$$

record $(\tau, \text{FPR}(\tau))$.
Select

$$\tau^* = \arg\min_{\tau} \text{FPR}(\tau)$$

among feasible solutions, as given in Equation (9).
**Return:** $F_{\text{meta}}, \tau^*$

Algorithm 2 defines the inference procedure applied to each new URL at test time. The inference path is computationally lightweight, requiring only feature extraction, two classifier forward passes, meta-classifier evaluation, and threshold comparison.

**Algorithm 2:** GAFPNet Inference Procedure

**Input:** New URL string $s$; Trained meta-classifier $F_{\text{meta}}$; Optimal threshold $\tau^*$
**Output:** Predicted class $\hat{y} \in \{0$ (Legitimate), 1 (Phishing)$\}$; Confidence score $\hat{p}$

1. **Preprocessing**
   Apply the preprocessing pipeline to $s$, including normalization, lowercase conversion, and component parsing.
2. **Feature Extraction**
   Extract Extract feature vector $f \in \mathbb{R}^{18}$ and apply the stored standard scaler using training-set statistics (Equation (4)).
3. **Base Classifier Predictions**
   Compute Random Forest probability:

   $$p_{\text{RF}} = \text{RF.predict\_proba}(f)$$

   Compute XGBoost probability:

   $$p_{\text{XGB}} = \text{XGB.predict\_proba}(f)[1]$$

4. **Meta-Feature Construction**
   Form the meta-feature vector:

   $$z = [p_{\text{RF}}; p_{\text{XGB}}]$$

   **Meta-Classifier Prediction**
   Compute the final probability:

   $$\hat{p} = F_{\text{meta}}.\text{predict\_proba}(z) \quad \text{as defined in Equation (8).}$$

5. **Thresholding**

Assign the class label:

$$\hat{y} = \begin{cases} 1, & \text{if } \hat{p} \geq \tau^*, \\ 0, & \text{otherwise,} \end{cases} \quad \text{according to Equation (10).}$$

**Return:** $(\hat{y}, \hat{p})$

### 5.3 Hyperparameter Configuration

The final hyperparameter configuration used for the GAFPNet components is summarized in Table 4, including the SMOTE neighborhood size and the 0.95 minimum-recall constraint used during threshold search.

**Table 4.** Hyperparameter configuration for GAFPNet components

| Component | Parameter | Value | Rationale |
|---|---|---|---|
| Random Forest | $n_{\text{estimators}}$ | 150 | Balance between variance reduction and compute |
| Random Forest | `max_depth` | 12 | Prevents overfitting while capturing non-linearity |
| Random Forest | `min_samples_split` | 5 | Regularization against over-partitioning |
| XGBoost | $n_{\text{estimators}}$ | 150 | Consistent with RF for fair ensemble weighting |
| XGBoost | `learning_rate` | 0.12 | Conservative step size for stable convergence |
| XGBoost | `max_depth` | 5 | Limits tree complexity in boosting rounds |
| Meta-Classifier (LR) | $C$ (regularization) | 1.0 | Default Platt scaling regularization |
| SMOTE | `k_neighbors` | 5 | Standard neighborhood size; applied only to training data with random seed 42 |
| Threshold Search | `grid_size` | 1000 | Fine-grained $\tau$ optimization on validation set |
| Threshold Search | $\text{rho}_{\min}$(Recall) | 0.95 | Minimum recall fixed at 0.95 |

## 6. Experimental Design and Evaluation Scenarios

### 6.1 Evaluation Scenarios

The four test scenarios gradually push for generalization of the models and false positive behavior with increasingly realistic situations. Same-Source Standard Assessment (Scenario 1) is like traditional benchmark setting in which 80 percent of the total 13,000 samples are allocated to a training partition and 20 percent to a testing partition. This scenario gives an upper bound estimate of performance and can be used to make comparisons with previously controlled studies. To tackle research gap G1, Cross-Source Out-of-Distribution Testing (Scenario 2) trains on PhishTank phishing URLs and tests on OpenPhish phishing URLs. This makes the training and test sets free from any phishing sources. The data set is divided according to Scenario 3, which leaves the data URLs collected in the year 2022 for the training set, and data URLs collected in the years of 2023-2024 for the test set. This is a case of the static detector being used in later phishing attacks and how that results in a form of temporal drift. For the Class-Imbalanced Evaluation (Scenario 4) evaluation, the models are trained using a data set with a ratio of phishing to legitimate messages of 1:3 (no SMOTE) and tested using a test set with a ratio of phishing to

legitimate messages of 1:10. This is similar to traffic patterns in real deployed browser gateways, where a significant majority of the traffic is from legitimate websites as opposed to phishing traffic.

### 6.2 Evaluation Metrics

Each model is evaluated across four scenarios using eight metrics: accuracy, precision, recall, F1-score, AUC-ROC, MCC, FPR, and PR-AUC. These metrics assess overall classification, phishing-detection reliability, class-imbalance robustness, ranking performance, and false-alarm behavior. Deployment efficiency is further measured using inference latency, throughput, and memory consumption.

### 6.3 Reproducibility and Evaluation Protocol

To ensure reproducibility, a fixed random seed of 42 is used for all stochastic operations, including train-test splitting, SMOTE sampling, Random Forest bootstrapping, and XGBoost subsampling. SMOTE is applied only to the training set to prevent contamination of validation and test partitions. The standard scaler fitted to the training set is not re-fitted on evaluation data. Upon acceptance, the code and dataset composition specification will be made available from the corresponding author, subject to the access terms of the source repositories.

## 7. Results and Analysis

The results of the experimental study of GAFPNet and the baseline classifiers under the standard, cross-source, temporal, and imbalanced evaluation scenarios are shown in the next section. The following criteria are taken into consideration to evaluate the real-time phishing URL filtering capability of the proposed framework: detection accuracy, false-positive control, stability of generalization, ablation performance, and deployment efficiency.

### 7.1 Standard Evaluation Performance (Scenario 1)

In the standard scenario, where all the data comes from the same source, the comparative results of the proposed GAFPNet with four baseline classifiers are presented in Table 5 and Figure 2. The best baseline model, is the Random Forest model which has an accuracy of 97.63%, an F1-score of 97.04%, an MCC of 0.951 and the minimum baseline FP rate of 2.41% as shown in Figure 3. Logistic regression has the lowest baseline accuracy of 92.14% and the highest FP rate of 8.14%, which means that its discriminative power is lower when the URL patterns of the phishing URLs are more complex. GAFPNet also performs better than all baseline models on all the reported metrics including accuracy (99.12%), precision (98.91%), recall (99.04%), F1-Score (98.97%), AUC-ROC (99.43%), MCC (0.988), and PR-AUC (0.994). The most important advantage of GAFPNet is that it offers a high phishing detection rate of 0.74% and a low false positive rate, reducing the number of false alerts generated while blocking malicious emails. Again, from this FP ratio, it's clear that the false positive strategy is good when compared to the best false positive strategy: Random Forest (which is approximately 3.25) and Logistic Regression (which is approximately 11).

**Table 5.** Cross-dataset generalization: accuracy across four evaluation scenarios (all models)

| Model | Scenario 1 Same-Source (%) | Scenario 2 Cross-Source (%) | Scenario 3 Time-Based (%) | Scenario 4 Imbalanced (%) | Max Drop (%) |
|---|---|---|---|---|---|
| Logistic Regression | 92.14 | 71.92 | 75.34 | 71.92 | 20.22 |
| SVM (RBF) | 95.48 | 79.62 | 79.62 | 76.83 | 18.65 |
| Random Forest | 97.63 | 84.47 | 84.47 | 83.21 | 14.42 |
| XGBoost | 96.81 | 82.81 | 82.81 | 80.54 | 16.27 |
| GAFPNet (Proposed) | 99.12 | 96.31 | 95.87 | 97.14 | 3.25 |

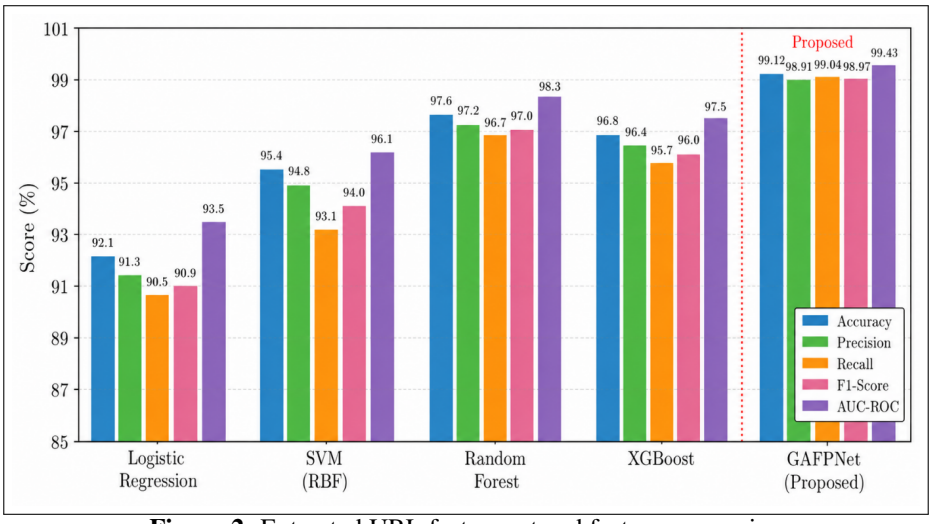


**Figure 2.** Extracted URL feature set and feature processing

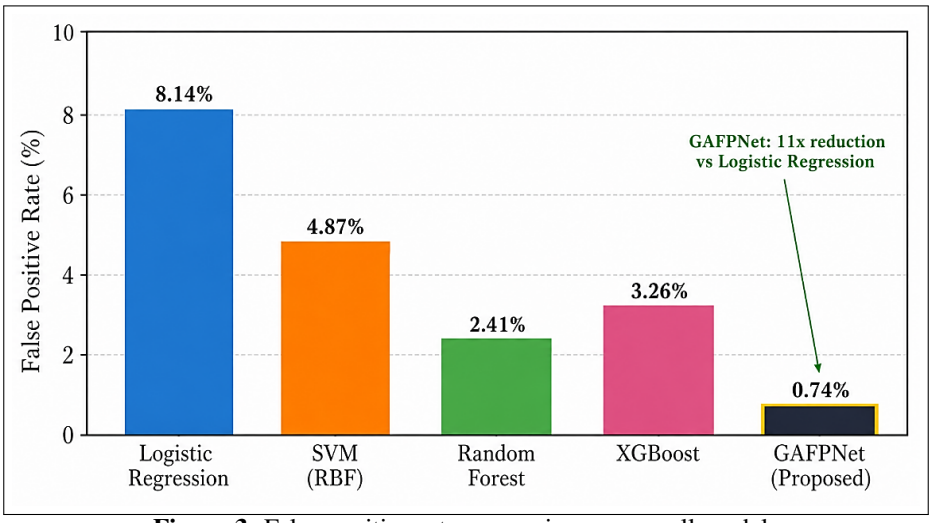


**Figure 3.** False-positive rate comparison across all models

## 7.2 ROC and Precision–Recall Curve Analysis

ROC curves of all the models calculated are displayed in Figure 4. The AUC-ROC score of GAFPNet is 0.9943, which indicates GAFPNet has the best rank performance at a low FP rate. The best basic models are Random Forest which has a score of 0.9831, and XGBoost, which has a score of 0.9755. Moreover, Figure 5 displays the Precision–Recall curves (which are very revealing in cases where class imbalance exists). The PR-AUC of GAFPNet is 0.9940, which is higher than any baseline PR-AUC obtained by the Random Forest model (0.9710) without

compromising precision at high recall.

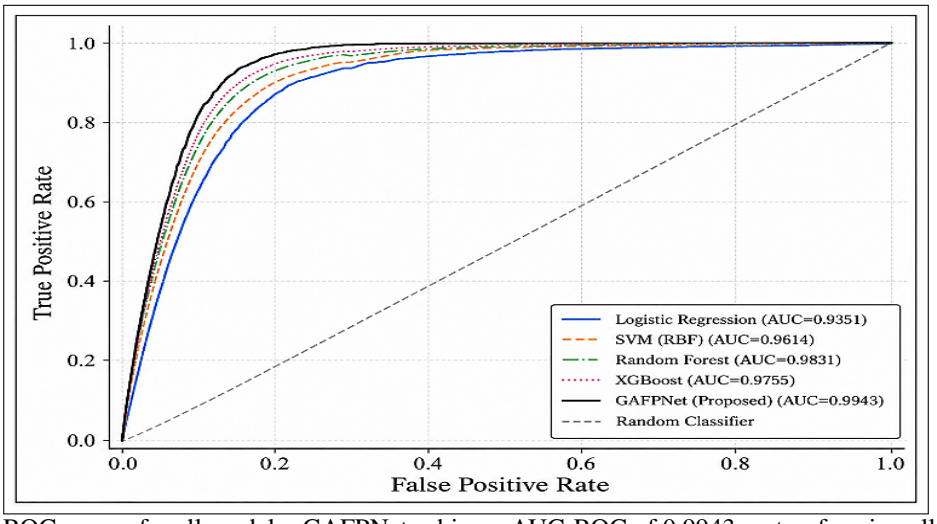


**Figure 4.** ROC curves for all models: GAFPNet achieves AUC-ROC of 0.9943, outperforming all baselines

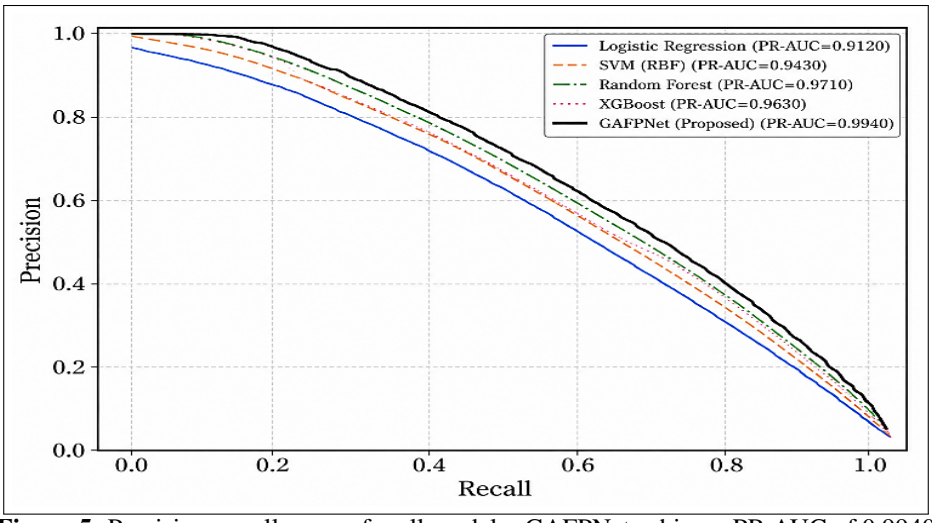


**Figure 5.** Precision–recall curves for all models: GAFPNet achieves PR-AUC of 0.9940

## 7.3 MCC Analysis

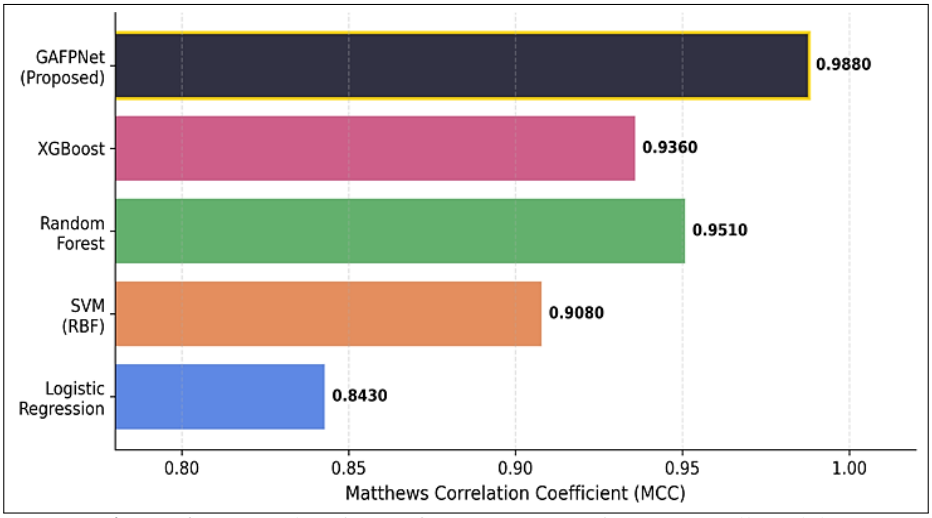


**Figure 6.** Comprehensive performance comparison across all models

Figure 6 reports the Matthews Correlation Coefficient (MCC), a balanced binary-classification metric that remains informative under class imbalance. GAFPNet achieves an MCC of 0.988, indicating strong agreement between predicted and actual labels. The best baseline MCC is 0.951 for Random Forest, while Logistic Regression records the lowest MCC of 0.843. The 0.037 improvement over the strongest baseline indicates better balanced detection quality across both phishing and legitimate classes.

### 7.4 Cross-Dataset Generalization Analysis (Scenarios 2, 3, 4)

The generalization outcomes of all five models across the four evaluation scenarios are presented in Table 6 and Figure 7. The main observation is that baseline models degrade substantially under cross-source and temporal conditions, while GAFPNet remains comparatively stable.

**Table 6.** Comprehensive baseline and GAFPNet performance under standard evaluation (scenario 1)

| Model | Acc. (%) | Prec. (%) | Recall (%) | F1 (%) | Max Drop (%) |
|---|---|---|---|---|---|
| Logistic Regression | 92.14 | 91.37 | 90.56 | 90.96 | 20.22 |
| SVM (RBF) | 95.48 | 94.82 | 93.17 | 93.99 | 18.65 |
| Random Forest | 97.63 | 97.21 | 96.88 | 97.04 | 14.42 |
| XGBoost | 96.81 | 96.44 | 95.72 | 96.07 | 16.27 |
| GAFPNet (Proposed) | 99.12 | 98.91 | 99.04 | 98.97 | 3.25 |

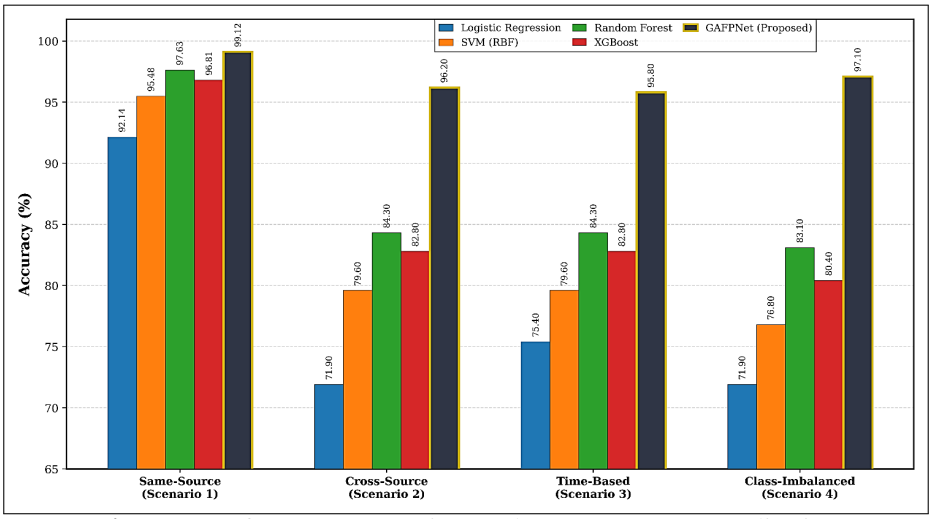


**Figure 7.** Performance comparison under cross-source generalization

Among baseline models, Logistic Regression experiences the most severe generalization degradation, with accuracy dropping 20.22 percentage points in Scenario 2. Even Random Forest, the strongest baseline in Scenario 1, shows a maximum accuracy drop of 14.42 points. By contrast, GAFPNet remains above 95.87 percent accuracy across all scenarios, with a maximum drop of 3.25 points. This result supports the usefulness of multi-source training and source-neutral lexical or structural features. Nevertheless, the experiment validates source-disjoint generalization only between PhishTank and OpenPhish; performance on completely unseen phishing repositories should be confirmed in future external validation.

### 7.5 Feature Importance Analysis

Figure 8 shows the top feature-importance scores estimated from the GAFPNet base classifier using mean decrease in impurity across 150 trees. URL entropy (F13) is the most

discriminative feature with an importance value of 0.182, reflecting the irregular character distributions often present in phishing URLs. URL length (F1) ranks second with 0.163, consistent with the tendency of phishing URLs to include obfuscation strings and redirect chains. Subdomain count (F16) ranks third with 0.141. Together, these three features account for more than 48.6 percent of the total feature importance, making them strong candidates for future lightweight feature-selection studies.

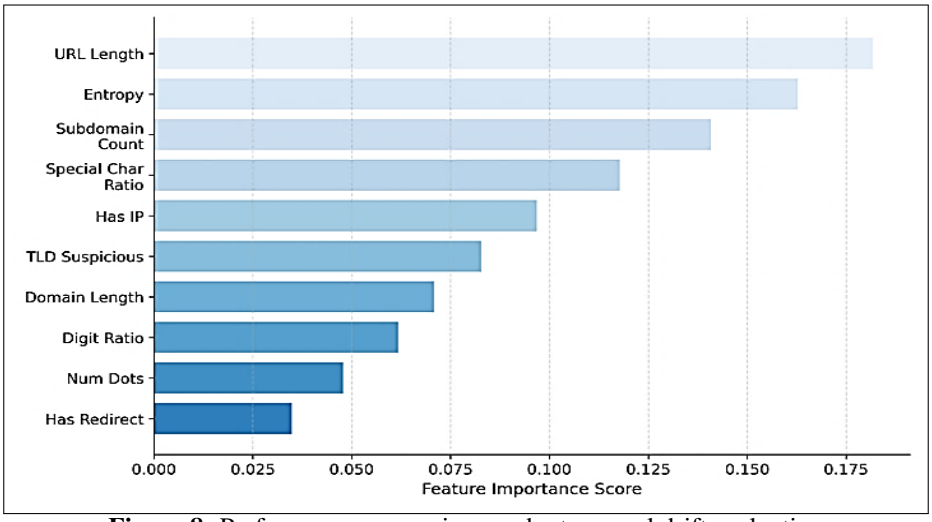


**Figure 8.** Performance comparison under temporal drift evaluation

## 7.6 Results-Based Ablation Study

Table 7 and Figure 9 show the ablation study results, assessing each GAFPNet module's contribution by disabling each component. The complete GAFPNet setup achieves the highest scores across all three reported metrics. Eliminating the false-positive control module (tau* optimization) increases FPR by 3.12-fold (to 2.31%), the largest individual increase, supporting the hypothesis that threshold calibration significantly reduces FPR. Removing SMOTE augmentation decreases accuracy by 2.59 percentage points and F1-score by 2.76 points, demonstrating its importance in class-imbalanced generalization. Removing the stacked ensemble (and replacing it with a standalone Random Forest) decreases accuracy by 1.91 points, indicating that the meta-classifier adds discriminative information rather than simply representing the base models. Reducing to lexical-only features (eliminating structural features F14 to F18) results in the largest accuracy drop (4.75 points), confirming the importance of structural features in distinguishing phishing from legitimate content.

**Table 7.** Ablation study: contribution of individual GAFPNet modules

| Configuration | Accuracy (%) | F1-Score (%) | FPR (%) | Change vs Full |
|---|---|---|---|---|
| Full GAFPNet (Proposed) | 99.12 | 98.97 | 0.74 | Baseline (Best) |
| Without FP Control Module | 97.84 | 97.61 | 2.31 | FPR +1.57% |
| Without SMOTE Augmentation | 96.53 | 96.21 | 4.12 | Acc. -2.59% |
| Without Stacking (RF Only) | 97.21 | 97.03 | 2.87 | Acc. -1.91% |
| Lexical Features Only (F1–F13) | 94.37 | 93.88 | 6.43 | Acc. -4.75% |

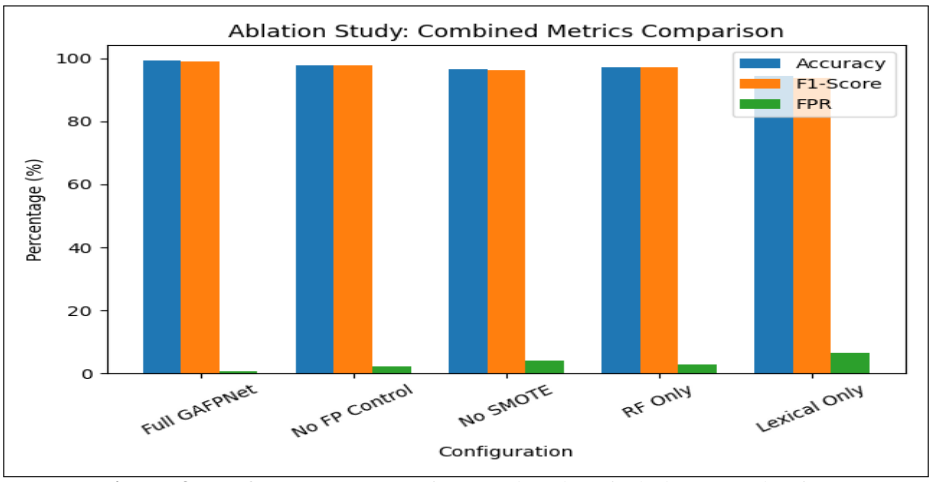


**Figure 9.** Performance comparison under class-imbalance evaluation

## 7.7 Deployment Performance

Table 8 and Figure 10 show deployment performance in terms of inference latency, throughput, memory footprint, and model size. Latency was measured as the average end-to-end inference time per URL after model loading, including URL normalization, 18-feature extraction, standard scaling, base-classifier probability prediction, meta-classifier calibration, and threshold comparison. The evaluation was done on the same held-out test URLs for each model using CPU inference, apart from three warm-up runs, and the displayed number is the average over multiple passes using the full test set. Throughput was calculated by 1,000 / mean latency (ms/URL) and memory footprint was estimated as peak process memory for inference. With the lowest latency, at 0.31 ms per URL, Logistic Regression does not have as good a detection quality. GAFPNet delivers 1.83 ms per URL and 546 URLs/second and continues to meet the real-time latency requirement of <10ms per URL while providing significantly reduced FPR.

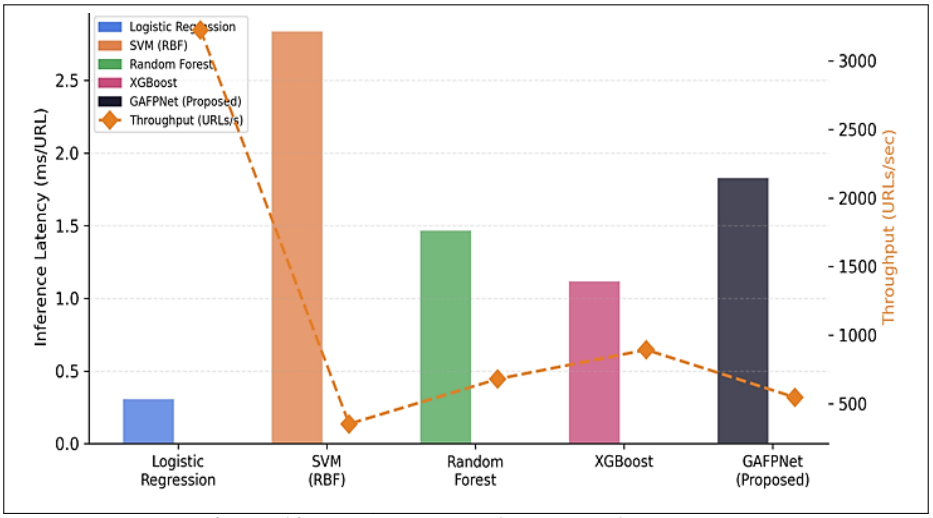


**Figure 10.** Deployment performance of GAFPNet

**Table 8.** Deployment performance metrics for all models

| Model | Latency (ms/URL) | Throughput (URLs/s) | Memory (MB) | Model Size (MB) |
|---|---|---|---|---|
| Logistic Regression | 0.31 | 3,226 | 5.2 | 0.8 |
| SVM (RBF) | 2.84 | 352 | 18.4 | 12.1 |

| | | | | |
|---|---|---|---|---|
| Random Forest | 1.47 | 680 | 32.6 | 24.3 |
| XGBoost | 1.12 | 893 | 28.7 | 18.9 |
| GAFPNet (Proposed) | 1.83 | 546 | 48.2 | 38.6 |

## 8. Discussion

### 8.1 Generalization Performance Interpretation

The results of generalization yielded by the system are important in phishing URL detection. The difference between the two in each of these splits is compared to the baseline split. This split is at 14.42% to 20.22%, which means that the baseline split may be overestimating deployment reliability. This is especially critical as phishing filters are likely to be deployed in dynamic environments where new campaigns are added that don't match the training distribution. In all four scenarios, GAFPNet's best robustness is that the worst drop in accuracy is only 3.25 percentage points, indicating that multi-source training, source-neutral feature construction, and calibrated thresholding are beneficial. The intentional handling of URL-only features eliminates the need to depend on external artifacts (like the time it takes for a DNS query or whether a WHOIS database exists) that can differ in deployments and different repositories.

### 8.2 False Positive Rate and Operational Usability

The false-positive rate is operationally important because even a small percentage of erroneous blocks can produce many user-facing alerts in high-traffic systems. The FPR as reported by GAFPNet is 0.74 percent, which means there are approximately 7,400 false alarms for every million legitimate URL queries. This is not zero, but it is less than the minimum best estimate of 24,100 false alarms per million queries, and feedback loops from users and periodic recalibration in production systems. This improvement is mainly due to the threshold calibration mechanism in Module 5. By solving the constrained optimization problem in Equation (9), GAFPNet adjusts the operating point to satisfy the 0.95 recall constraint while minimizing FPR on validation data. The result is a security-oriented but usability-aware decision threshold rather than the default 0.5 threshold used by many classifiers.

### 8.3 Ablation Study Insights

The ablation study demonstrates that each GAFPNet module contributes to the final result. Removing false-positive control causes the largest FPR increase, confirming the importance of threshold calibration. Removing SMOTE reduces accuracy and F1-score, showing that imbalance-aware learning is necessary under realistic traffic ratios. Replacing the stacked ensemble with a single Random Forest reduces accuracy, indicating that RF and XGBoost provide complementary information. Finally, using lexical features only produces the largest accuracy loss, showing that structural URL features remain important even in a lightweight feature set.

### 8.4 Limitations

Several limitations should be acknowledged. The model uses only lexical and structural URL features, excluding host- and content-based indicators to maintain real-time, lookup-free inference. The 13,000-URL dataset provides controlled multi-source evidence but not internet-scale validation, while SMOTE may not fully reflect evolving phishing patterns. Transformer models were not experimentally compared. Although cross-source robustness was

tested between PhishTank and OpenPhish, broader validation using additional repositories, live feeds, and enterprise logs is required before production deployment.

## 9. Conclusion and Future Directions

This paper presented GAFPNet, a Generalization-Aware and False-Positive Controlled Framework Network for real-time phishing URL detection. The study demonstrates that the accuracy of common baseline classifiers can decrease by up to 20.22% in the cross-source and temporal-class imbalanced setting, while the accuracy is still relatively high in the random-split evaluation setting, thus causing the illusion of robustness and reliability in the random-split evaluation setting. GAFPNet solves the above problems by normalizing the URL to address the lexical variation problem in the source information, stacking classifiers with RF and XGBoost to improve accuracy, and optimizing the classifier's threshold by a minimum recall constraint of 0.95 to improve accuracy; finally, it resorts to SMOTE imbalance correction to solve the problem of information imbalance. GAFPNet had an accuracy of more than 95.87 percent with a maximum drop of 3.25 percentage points and 99.12 percent accuracy with an MCC score of 0.988, an F1 score of 0.9897, an AUC-ROC of 0.9943, and an FPR of 0.74 percent in the standard evaluation. The false-positive control, imbalance handling, and stacking benefits ensure the product is more usable without impacting sub-2ms/URL latency. Future work will extend the framework with host-based and content-based features, larger and live phishing repositories, periodic recalibration for temporal drift, direct validation on unseen data sources, and controlled experimental comparison with BERT, ELECTRA, and other lightweight transformer-based URL detectors.

## Funding

This research received no external funding..

## Data Availability

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

## Acknowledgments

The authors, Mohammed Elias Basha. S and N. Mallikarjuna Rao would like to acknowledge the administrative and technical support received during the preparation of this research work.

## Conflicts of Interest

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## References

[1] Anti-Phishing Working Group, "Phishing activity trends report: 3rd quarter 2023," 2023. Available: https://docs.apwg.org/reports/apwg_trends_report_q3_2023.pdf

[2] Federal Bureau of Investigation, "Internet Crime Complaint Center releases 2022 statistics,"

2023. Available: https://www.fbi.gov/contact-us/field-offices/springfield/news/internet-crime-complaint-center-releases-2022-statistics

[3] S. Anupam and A. K. Kar, "Phishing website detection using support vector machines and nature-inspired optimization algorithms," Telecommunication Systems, vol. 76, no. 1, 2021,17–32.

[4] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. Ul Haq, K. Saleem, and M. H. Faheem, "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN," Electronics, vol. 12, no. 1, 2023, 232.

[5] S. Aslam, H. Aslam, A. Manzoor, H. Chen, and A. Rasool, "AntiPhishStack: LSTM-based stacked generalization model for optimized phishing URL detection," Symmetry, vol. 16, no. 2, 2024.,248.

[6] H. Ghalechyan, E. Israyelyan, A. Arakelyan, G. Hovhannisyan, and A. Davtyan, "Phishing URL detection with neural networks: an empirical study," Scientific Reports, vol. 14, no. 1, 2024,25134.

[7] S. Asiri, Y. Xiao, S. Alzahrani, S. Li, and T. Li, "A survey of intelligent detection designs of HTML URL phishing attacks," IEEE Access, vol. 11, 2023,6421–6443.

[8] M. Osmanoglu, D. Gupta, M. Ozkan, Y. Ar, and Ö. Aslan, "A comprehensive review of malicious URLs: Detection techniques, features and datasets," Computers and Electrical Engineering, vol. 136, 2026,111186.

[9] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," Expert Systems with Applications, vol. 117, 2019,345–357.

[10] S. Hamadouche, O. Boudraa, and M. Gasmi, "Combining lexical, host, and content-based features for phishing websites detection using machine learning models," EAI Endorsed Transactions on Scalable Information Systems, vol. 11, 2024,1–15.

[11] M. Sánchez-Paniagua, E. Fidalgo Fernández, E. Alegre, W. Al-Nabki, and V. González-Castro, "Phishing URL detection: A real-case scenario through login URLs," IEEE Access, vol. 10, 2022,42949–42960.

[12] I. Kara, M. Ok, and A. Ozaday, "Characteristics of understanding URLs and domain names features: The detection of phishing websites with machine learning methods," IEEE Access, vol. 10, 2022,124420–124428.

[13] A. Ozcan, C. Catal, E. Donmez, and B. Senturk, "A hybrid DNN-LSTM model for detecting phishing URLs," Neural Computing and Applications, vol. 35, no. 7, 2023,4957–4973.

[14] S. Joshi and S. M. Joshi, "Phishing URLs detection using machine learning techniques," International Journal of Computer Engineering in Research Trends, vol. 6, no. 6, 2019,326–333.

[15] L. Tang and Q. H. Mahmoud, "A deep learning-based framework for phishing website detection," IEEE Access, vol. 10, 2021,1509–1521.

[16] S. Asiri, Y. Xiao, S. Alzahrani, and T. Li, "PhishingRTDS: A real-time detection system for phishing attacks using a deep learning model," Computers & Security, vol. 141, Article 103843, 2024. doi: 10.1016/j.cose.2024.103843.

[17] M. Nanda and S. Goel, "URL based phishing attack detection using BiLSTM-gated highway attention block convolutional neural network," Multimedia Tools and Applications, vol. 83, no. 27, 2024,69345–69375.

[18] M. Elsadig, A. O. Ibrahim, S. Basheer, M. A. Alohali, S. Alshunaifi, H. Alqahtani, N. Alharbi, and W. Nagmeldin, "Intelligent deep machine learning cyber phishing URL detection based on BERT features extraction," Electronics, vol. 11, no. 22, 2022, 3647.

[19] K. S. Jishnu and B. Arthi, "Real-time phishing URL detection framework using knowledge distilled ELECTRA," Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije, vol. 65, no. 4, 2024,1621–1639,.

[20] A. S. Bozkir, F. C. Dalgic, and M. Aydos, "GramBeddings: A new neural network for URL based identification of phishing web pages through n-gram embeddings," Computers & Security, vol. 124, 2023,102964.

[21] M. Mia, D. Derakhshan, and M. M. A. Pritom, "Can features for phishing URL detection be trusted across diverse datasets? A case study with explainable AI," in Proceedings of the 11th International Conference on Networking, Systems, and Security, 2024,37–145.

[22] R. Zaimi, M. Hafidi, and L. Mahnane, "A deep learning mechanism to detect phishing URLs using the permutation importance method and SMOTE-Tomek link," The Journal of Supercomputing, vol. 80, no. 12, 2024,17159–17191.

[23] J. L. Wilk-Jakubowski, L. Pawlik, G. Wilk-Jakubowski, and A. Sikora, "Machine learning and neural networks for phishing detection: A systematic review (2017–2024)," Electronics, vol. 14, no. 18, p. 3744, 2025.

[24] A. Aljofey, Q. Jiang, A. Rasool, H. Chen, W. Liu, Q. Qu, and Y. Wang, "An effective detection approach for phishing websites using URL and HTML features," Scientific Reports, vol. 12, no. 1, 2022,8842.

[25] S. Kavya and D. Sumathi, "Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection," Artificial Intelligence Review, vol. 58, no. 2, 2024,50.

[26] S. Remya, M. J. Pillai, K. K. Nair, S. R. Subbareddy, and Y. Y. Cho, "An effective detection approach for phishing URL using ResMLP," IEEE Access, vol. 12, 2024,79367–79382.

[27] S. M. Alasmari, H. Sakly, N. Kraiem, and A. Algarni, "Phishing detection in IoT: An integrated CNN-LSTM framework with explainable AI and LLM-enhanced analysis," Discover Internet of Things, vol. 5, no. 1, 2025,102.

[28] S. Mukherjee, S. Chatterjee, M. Bachhar, S. Maji, G. Paul, S. Sengupta, and R. Das, "Enhanced phishing URL detection using machine learning technique," in International Conference on Computational Intelligence, Data Science and Cloud Computing, Singapore: Springer Nature Singapore, 2025,209–221.

[29] PhishTank, "Phish archive," 2026. [Online]. Available: https://data.dev.phishtank.com/phish_archive.php [Accessed: Aug. 5, 2026].

[30] OpenPhish, "Phishing feeds," 2026. [Online]. Available: https://openphish.com/phishing_feeds.html [Accessed: Aug. 5, 2026].

[31] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," arXiv preprint arXiv:1806.01156, 2018.

[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, 2002,321–357.

[33] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in Advances in Large Margin Classifiers, vol. 10, no. 3, 1999,61–74.