International Journal of Safety and Security Engineering
Vol. 16, No. 5, May, 2026, pp. 1075-1089
Journal homepage: http://iieta.org/journals/ijsse

Adaptive Control-Based Phishing Detection with Model-Based Learning under Feature Drift

Sundos F. Jabbar[1], Karrar Imran Ogaili[2], Hasan Abdulameer Hasan[2*], Rasha Hussein Joudah[2]

[1] Department of Information Networks, College of Information Technology, University of Babylon, Hillah 51002, Iraq
[2] Department of Cybersecurity, College of Information Technology, University of Babylon, Hillah 51002, Iraq

Corresponding Author Email: hasan.abdulameer@uobabylon.edu.iq

https://doi.org/10.18280/ijsse.160513

Received: 28 February 2026
Revised: 7 May 2026
Accepted: 25 May 2026
Available online: 31 May 2026

Keywords:
adaptive phishing detection, control-based learning, feature drift mitigation, dynamic thresholding, model stability, AI-driven cybersecurity

ABSTRACT

Fixed-rule baseline models struggle to detect phishing attacks when their structure, language, and behavior change. As such, the present study proposed an adaptive control (AC)-based phishing detection framework that uses feature correction, dynamic threshold adjustments, and controlled parameter updates to adapt its behavior. More specifically, it uses a unified method to evaluate lexical, structural, and linguistic cues and make consistent decisions when the patterns of phishing attacks change across the batches. Its behavior was found to remain stable, even when structural and linguistic changes occurred. More specifically, the proposed AC-based phishing detection framework was found to be accurate 99.12, 98.87, and 97.56% of the time, respectively; its recall was 99.05, 98.74, and 97.18%, respectively; and its F1 score was 99.14, 98.88, and 97.54%, respectively. Therefore, AC can be used to reliably detect phishing attacks in environments with ever-changing feature distributions. However, as the proposed AC-based phishing detection framework was tested solely using curated public datasets, these results should be interpreted with that in mind.

1. INTRODUCTION

Phishing attacks are perpetrated using pages and text messages that closely mimic the appearance and behavior of authentic online platforms [1]. At present, both the scale and diversity of automated phishing kits and the systematic reuse of phishing attack templates have substantially increased [2]. As such, conventional fixed-rule detection systems, that are developed using decision rules from historical datasets, are structurally vulnerable to these changes. This is because such models fail to detect the new features that are characteristic of present-day phishing attacks [3, 4]. Recent studies on phishing trends and cybercrime response further show that prevention and detection remain constrained by evolving attack timing, user-facing deception, and response latency [5, 6]. AI-driven email-security research also links phishing defense to scalable big-data analysis of mail streams [7]. Perpetrators also systematically exploit the structure of uniform resource locators (URLs) as well as embedded links. This method successfully overcomes legacy filters, that depend on familiar lexical patterns [8]. Web-based threat-signal studies using predictive analytics and feature attribution similarly confirm that URL and page-level indicators remain central to phishing discrimination [9]. Fixed-rule detection systems are also vulnerable to feature-level drift as they lack adaptive capabilities. This is because, as the statistical correlation between the input features and the class labels change across the batches, they produce higher false-positive or false-negative rates [10]. As such, these vulnerabilities remain a constant challenge for system designers [8].

The same deployment problem is visible in broader AI-security and security-control literature, including cyber-physical systems, maritime-security AI, and ERP-integrated IoT environments, where data-source quality, operational context, and control reliability affect practical adoption [11-13]. System designers use two types of approaches to address these challenges, namely, (1) static machine and deep learning models and (2) drift-aware and adaptive approaches. Static machine and deep learning models perform very accurately when tested using curated benchmark datasets. However, the feature representations, decision thresholds, and parameters of these models cannot be adjusted when the input distribution changes [14-18]. Meanwhile, drift-aware and adaptive approaches use re-training, feature substitution, and weight adjustments to, partially, overcome these limitations. However, these solutions are reactive and lack coordinated, closed-loop regulation at the feature, decision, and parameter levels [19]. Therefore, such systems fail to simultaneously manage input representation drift, rising false-positive rates, and unstable gradients.

As such, the present study proposed an adaptive control (AC)-based phishing detection framework that uses a controlled, dynamic process that adheres to structured feedback laws. It, primarily, differs from extant models in terms of architecture. More specifically, standard machine learning models learn a fixed map from historical datasets, while drift-aware approaches react and adapt, but do not implement simultaneous coordinated, closed-loop regulation at the feature, decision, and parameter levels. Meanwhile, the proposed AC-based phishing detection framework uses (i)

feature-level correction to decrease deviation from the benign manifold when the input distribution changes; (ii) dynamic threshold adaptation to regulate false-positive behavior according to the error rates; and (iii) parameter-level controlled updates to create gradients using drift-sensitive gain matrices. Furthermore, these three control layers operate within a unified stability monitoring function to provide a measurable indicator of adaptation at the system-level.

The proposed AC-based phishing detection framework is more than just a classification model. It is also a detection tool that can be integrated into existing security systems. Phishing attacks target users, primarily, via entry points, such as emails, web searches, and network gateways. Adaptive approaches can be implemented at each of these attack entry points. For instance, the proposed AC-based phishing detection framework can be implemented in an email filtering pipeline, a secure web gateway, or a network monitoring platform. At these entry points, it will receive the raw messages or request data, extract unified feature representations, produce calibrated phishing probability (pt) scores, and feed operational error signals back to maintain stable phishing attack detection even when the input distributions change. Therefore, the proposed AC-based phishing detection framework satisfies demand for safety and security by providing constant, reliable protection in environments with ever-changing feature distributions.

The research questions of the present study include (1) can a control-guided model adapt its detection decisions when phishing behaviors change across the batches? (2) how does structured feature tracking facilitate stable phishing detection when feature distributions change? (3) how does updating controlled parameters decrease error accumulation and improve the robustness of detecting hitherto unknown phishing attacks?

The rest of the present study is organised as follows. Section 2 evaluates extant phishing detection models and identifies research gaps, while Section 3 describes the proposed AC-based phishing detection framework. Section 4 discusses the datasets used in the present study as well as the feature extraction process and experiment settings. Section 5, on the other hand, analyses and compares the performance of the proposed AC-based phishing detection framework with that of other approaches, while Section 6 provides the conclusion and directions for future research.

2. LITERATURE REVIEW

Multiple studies have used machine learning, deep learning, and hybrid models to detect phishing attacks. Although this has led to the development of increasingly accurate classifiers that perform well on benchmark datasets, maintaining reliable phishing attack detection, in environments with ever-changing feature distributions, remains a hitherto unaddressed problem. Therefore, extant methods were reviewed and classified into four groups, before their strengths and limitations were assessed. The results of this review revealed gaps in the research that inspired the development of the proposed AC-based phishing detection framework.

2.1 Conventional machine learning approaches

Conventional supervised learning models have been widely employed for phishing attack detection using structured URL, Hypertext Markup Language (HTML), and domain-based features. Studies based on two University of California, Irvine datasets demonstrated that Random Forest combined with information-gain feature selection achieved an accuracy of approximately 97% in multi-class classification settings [20]. Similar findings were reported using the Kaggle URL dataset, where tree-based models, including Decision Trees, Random Forest, and Extreme Gradient Boosting (XGBoost), consistently outperformed probabilistic and margin-based classifiers when lexical URL features were used [21]. The effectiveness of Random Forest was further confirmed through evaluations incorporating URL characteristics, domain age, HTTPS indicators, and other phishing-related website features, achieving an accuracy of 97.6% [22].

More recent studies have reported even higher performance levels. Using the PhishOFE dataset and a comprehensive feature set consisting of URL, HTML, and derived attributes, CatBoost achieved an accuracy of 99.45% [17]. Likewise, a hybrid feature representation combining URL-based and hyperlink-based indicators attained an accuracy of 99.17% when implemented with XGBoost [15]. These findings indicate that ensemble learning methods can achieve excellent phishing detection performance when coupled with carefully engineered structured features. However, because these supervised models learn fixed decision boundaries during training, their effectiveness may deteriorate significantly when attackers modify URL structures, HTML compositions, or lexical patterns. Consequently, regular retraining is often required to maintain detection performance in rapidly evolving phishing environments.

2.2 Deep learning approaches

Deep learning models have been extensively applied to capture complex sequential and hierarchical patterns in URL strings, webpage content, and text messages. A convolutional neural network–long short-term memory (CNN–LSTM) hybrid model achieved an accuracy of 98.9% on a spoofing website dataset [23]. Transformer-based approaches have also been explored, with a probabilistic Bidirectional Encoder Representations from Transformers (BERT)-based network improving URL-stream detection performance by 4% when confidence-interval estimation was incorporated [24]. Similarly, a convolutional neural network enhanced with multi-head self-attention achieved 97.82% accuracy after being trained on a generative adversarial network-balanced dataset [25].

Hybrid deep learning architectures have also demonstrated strong performance in phishing detection. A BERT–LSTM model achieved 99.55% accuracy for email phishing detection [26], while a variational autoencoder-enhanced and grid-search-optimized CNN hybrid model achieved 99.44% accuracy on custom URL data [27]. These findings show that deep learning models can achieve high detection accuracy on benchmark datasets by learning complex representations from URLs, webpage structures, and message content. However, such models are computationally demanding and often require full or partial retraining to adapt to newly emerging phishing patterns. Moreover, they generally lack explicit mechanisms for regulating false-positive rates or moderating gradient dynamics under feature-drift conditions.

2.3 Hybrid and ensemble approaches

Hybrid and ensemble models have also been developed for phishing attack detection to improve feature coverage and

decision robustness. A hybrid generative adversarial network–convolutional neural network–long short-term memory–white shark optimization model achieved an accuracy of 97.94% on intrusion-detection-evaluation URL data [28]. Ensemble-based approaches have also shown strong performance, with Random Forest combined with ensemble scoring achieving 99% accuracy and producing more stable decisions than individual deep learning models [16]. Other studies have explored feedforward neural networks, dense neural networks, wide-and-deep architectures, and TabNet models combined with anti-phishing scores to balance accuracy, false-positive rate, and detection time [29]. Reinforcement-learning-based ensemble methods have further been applied to email phishing detection, where an ensemble deep Q-network model achieved 95% accuracy on real email streams [30]. These studies demonstrate that hybrid and ensemble architectures can improve phishing detection by combining multiple feature representations, classifiers, or optimization strategies. However, most existing models have primarily been evaluated under static conditions. Their adaptive capacity generally depends on data augmentation or gradual retraining rather than structured closed-loop control.

2.4 Drift-aware and adaptive learning approaches

Drift-aware and adaptive learning approaches have also been developed for phishing attack detection to address changes in attack patterns over time. Machine-learning-operations-based models that combine drift detection, Shapley Additive Explanations (SHAP)-guided feature replacement, and automated retraining have achieved post-drift F1 scores close to 0.995, with a drift-handling delay of approximately 18 s [19]. Explainable feature-selection methods based on SHAP and Local Interpretable Model-Agnostic Explanations (LIME) have also been shown to maintain high detection accuracy while reducing processing cost on web phishing datasets [14]. In addition, studies on robust, adaptive, and adversarial machine learning for cybersecurity have highlighted persistent research gaps related to drift robustness and feature sensitivity in phishing detection [18]. Although these approaches represent progress in adaptive phishing detection, they remain largely reactive to drift. Adaptation is usually performed through retraining, feature substitution, or weight adjustment only after performance degradation has occurred. Moreover, these methods do not provide coordinated closed-loop control across feature representations, decision boundaries, and parameter updates.

2.5 Research gaps

The review revealed a consistent gap in the architecture of all four groups. More specifically, conventional and deep learning models have no structured mechanisms with which to adapt feature drifts during implementation; hybrid and ensemble models are static in their decision-making logic; and drift-aware approaches lack the closed-loop coordination necessary to simultaneously regulate feature representation quality, decision boundary stability, and parameter update dynamics in a single unified framework. Therefore, the present study proposes an AC-based phishing detection framework that uses structured feedback laws to manage these three sources of detection instability.

As summarized in Table 1, existing phishing detection studies show strong benchmark performance but remain limited by the absence of continuous closed-loop adaptation under feature drift.

Table 1. Comparison of selected phishing detection studies across conventional, deep learning, hybrid, and adaptive methods

Ref	Dataset	Methodology	Key Limitation	Key Results
[15]	Custom 6,000 URLs	Hybrid XGBoost using URL and hyperlink features	No real-time adaptation or drift-handling mechanism	Accuracy 99.17%, precision/TPR 98.81%, FPR 0.49%
[23]	UCL Spoofing; PhishTank	CNN-LSTM hybrid for spoofing website URL detection	Does not address class imbalance or compare broader hybrid alternatives	UCL accuracy 98.9%; PhishTank accuracy 96.8%
[20]	UCI Datasets 1 and 2	Twenty-four classifiers with InfoGain feature selection	Rule-learning strategy showed the weakest performance	Best results: FilteredClassifier/J48 on Dataset 1; Random Forest on Dataset 2 above 97%
[17]	PhishOFE dataset, 101,063 URLs	CatBoost with comparative URL, HTML, and derived-feature analysis	Limited adversarial and drift-oriented testing	CatBoost accuracy 99.45% using the complete feature set
[21]	Kaggle 11,054 URLs	DT, RF, SVM, XGBoost, NB, and KNN using selected URL and HTML/JS features	Cannot detect compromised-domain attacks; RF is complex and time-consuming	RF accuracy 95.6%; DT accuracy 95.4%
[30]	Real emails, 5,000; synthetic samples, 1,000	Deep Q-Network reinforcement-learning phishing detector	High computational cost and exposure to adversarial risk	Accuracy 95%, precision 96%, recall 94%, AUC 0.96
[19]	PhishBench 2.0; PhishHaven URL data	Hybrid MLOps framework with SHAP-guided feature replacement and drift detection	High audit/explainability overhead; mainly URL-focused validation	Post-drift F1 approximately 0.995; drift-handling latency about 18 s
[14]	Web phishing dataset, 11,430 samples	SHAP and LIME explainable feature selection with RF and XGBoost	No continuous closed-loop adaptive-control mechanism	RF accuracy 97.41%; XGBoost accuracy 97.21%
[27]	Custom 27,534 balanced URLs; PhishTank secondary data	VAE-enhanced CNN with Enhanced Grid Search Optimisation	Limited cross-dataset validation; weak against embedded-object substitution	Accuracy 99.44%; precision 99.98%; F1-score 99.32%
[26]	SpamAssassin/UCI-derived phishing email corpus	BERT-LSTM hybrid email phishing detector	Requires large labelled corpora, extensive preprocessing, and high compute	Accuracy 99.55%, precision 99.61%, recall 99.55%, F1-score 99.24%

3. PROPOSED METHODOLOGY

The proposed AC-based phishing detection framework operates across three levels, namely, (1) feature representation correction, (2) decision threshold adaptation, and (3) parameter updates. It uses a unified representation of URL, HTML, behavioral, and linguistic cues, as well as structured feedback laws to maintain stable phishing detection, even when input distributions change across the batches.

In a multi-stage security processing pipeline, the proposed AC-based phishing detection framework would occupy the detection and adaptation layers. At the input stage, raw artefacts, such as email messages, hypertext transfer protocol requests, and URL submissions from the mail transfer agent, web proxy, or network sensor, are passed to a feature extraction module. The normalised representation (x_t) is processed by a feature-level AC block, before it is sent to a multilayer perceptron (MLP)-based detector. Here, the pt is evaluated against the adaptive threshold (τ_t) to generate a binary decision. Messages classified as phishing content are quarantined or flagged for review, while other traffic proceeds uninterrupted. At the end of each batch processing cycle, the parameter-level AC block uses a controlled gradient to update the detector weights, then includes the newly-labelled samples in the framework sans full re-training.

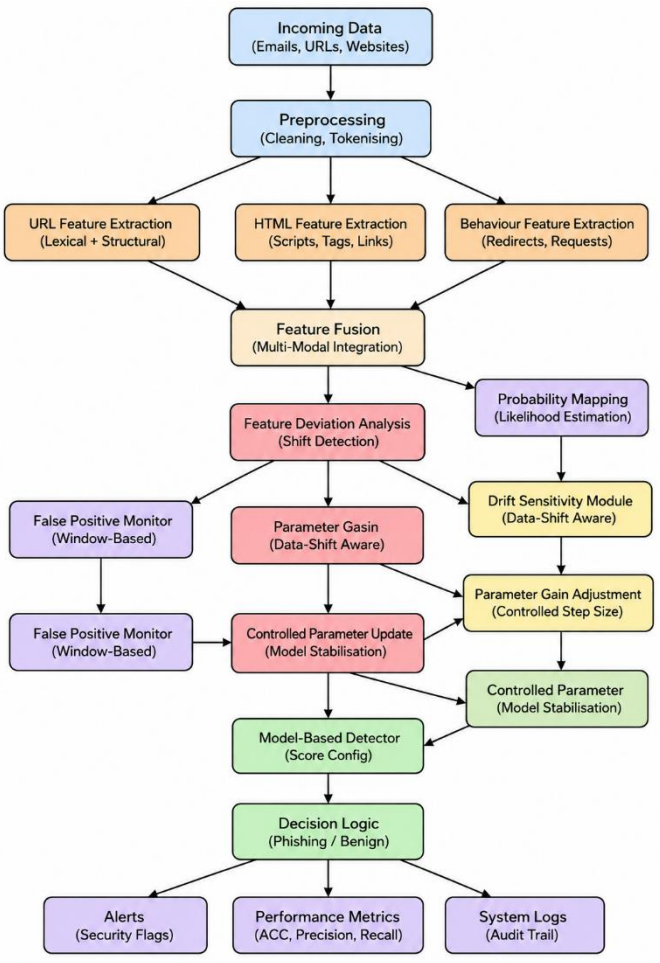

Figure 1. The architecture of the proposed adaptive control-based phishing detection framework

Figure 1 depicts the architecture of the proposed AC-based phishing detection framework. As seen, the features of incoming samples are extracted and added to the base layers of the AC. The feature-level AC block modifies the sample representations according to the drift behavior, then the threshold-level AC block moderates the decision boundary fluctuations when the false-positive rate changes. The parameter-level AC block then conditions the framework's updates according to the drift conditions. These controlled signals are fed into the model-based detector to yield the final decision. Lastly, the output layer generates alerts, performance metrics, and logs of the framework's behavior.

It illustrates the vertical workflow of the processes, namely, the acquisition of raw input via feature extraction and fusion, the feature-level control, the threshold-level control, the parameter-level control, the final decision, and the performance logging. The dashed feedback paths indicate closed-loop signals that link the output layer to each AC block.

3.1 Notation and base mapping

Each incoming-sample-at-time (t) is segregated into three feature groups, namely, (1) URL features (x^{url},t), (2) HTML features (x^{html},t), and (3) behavioral features (x^{beh},t). Eq. (1) is then used to link these feature groups into a single raw feature vector:

$$z_t = [x^{url} \quad x_t{}^{html} \quad x_t^{beh}] \in \mathbb{R}^d \tag{1}$$

where, z_t is the linked raw input vector, the superscript ($\top$) denotes vector transposition and is used only to indicate that the feature sub-vectors are concatenated into a single column-oriented raw feature vector, and d is the dimension of the total features. A scaling operator (S($\cdot$)) is then used to yield a normalised vector:

$$x_t = S(z_t) \tag{2}$$

where, $x_t \in \mathbb{R}^d$ may be produced by zero-mean unit-variance standardisation or robust percentile-based scaling. The $S(\cdot)$ is only added at the training phase and at the inference. No test-set statistics are used in the normalisation stage. The model-based detector then maps x_t to a scalar detection score (s_t) using a parametric function:

$$s_t = f_\theta(\tilde{x}_t) \tag{3}$$

which might be a shallow neural unit or another differentiable mapping. Eq. (3) defined the internal activation before probability conversion. The parameter vector θ collected all trainable weights. The choice of $f\theta(\cdot)$ remained flexible, since the AC design acted outside this mapping. This separation preserved generality while keeping a clear model structure. The pt then follows a logistic map:

$$\mathcal{p}_t = \sigma(S_t) = \frac{1}{1 + exp(-s_t)} \tag{4}$$

where, $\mathcal{p}_t \in (0, 1)$ and $\sigma(S_t)$ are the standard logistic sigmoid functions. This probability interacts with the τ_t in Section 3.4 and yields a smooth, differentiable surface for gradient-based parameter updates.

3.2 Base detection model and architectural specification

The function $f\theta(\cdot)$, seen in Eq. (3), is implemented as a tri-layer MLP with fully linked layers. A deliberately shallow

architecture was used to ensure that the AC laws handled the feature drift instead of the representational depth. It was also used to ensure reproducibility on hardware with limitations.

The input is the controlled feature vector ($\tilde{x}_t \in \mathbb{R}^d$) from the feature-level AC block (Section 3.3). The first hidden layer then maps the d-dimensional input to 256 units, with a rectified linear unit. The second hidden layer reduces it to 128 units, with the help of yet another rectified linear unit. A dropout layer (rate = 0.3) is applied after each hidden layer during the training phase. The output layer comprises a single linear unit, yielding s_t, per Eq. (3), with pt obtained using Eq. (4).

The training comprises binary cross-entropy loss (Eq. (13)) with the Adam optimisation algorithm, at a learning rate of 1×10^{-3} and default momentum of 0.9 and 0.999. The standard Adam adaptive gradient optimisation is not applied directly. Instead, the raw gradient (g_t) (Eq. (14)) passes through the parameter-level AC block and yields the controlled update (u_t^{θ}) (Eq. (15)). This is then applied, as per Eq. (16). The training is conducted in small batches of 32 samples each for, at most, 100 epochs. It is stopped early if the validation loss fails to improve after 10 consecutive epochs. The weights are initialised using the He uniform scheme. Ridge regression, at a coefficient of 1×10^{-4}, is applied to all the fully linked layers. All the experiments were conducted using Python© 3.10, with PyTorch© 2.1, on a workstation equipped with an Intel© Core™ i7 processor, 32 gigabytes of random-access memory, and an NVIDIA© GeForce RTX™ 3060 graphics processing unit.

The base classifier, the MLP, is architecturally distinct from the three AC layers. The feature-level AC block transforms x_t into $\tilde{x}_t$ upstream in the network, and the threshold-level AC block adjusts τ_t downstream of the sigmoid output, without modifying the network's weights. Lastly, the parameter-level AC block conditions each weight update using the gain matrix (K_t^{θ}) before it is applied to θ. The three control layers do not add learnable parameters to the base model to ensure clean architectural separation between the classifier and the control mechanism.

3.3 Feature-level adaptive control

3.3.1 Benign reference and feature error

The proposed AC-based phishing detection framework records a running estimate of the benign feature centre ($\mu_t^b \in \mathbb{R}^d$), which tracks the expected number of representations of legitimate samples. When a sample is labelled benign ($y_t = 0$), the reference is updated using an exponential moving average:

$$\mu_t^b = (1-\beta)\mu_{t-1}^b + \beta_{xt}\,, \qquad y_t = 0 \tag{5}$$

where $\beta \in (0,1)$ is the benign-based reference learning rate. Here, when $y_t = 1$, it indicates phishing. The feature error is then calculated using Eq. (6):

$$e_t^x = x_t - \mu_t^b \tag{6}$$

where $e_t^x \in \mathbb{R}^d$ is the size of the displacement of the current sample from the benign manifold. A large $\| e_t^x \|$ indicates that the location of the sample is far away from expected benign structure.

3.3.2 Feature control input and corrected representation

A feature correction input (u_t^x) is then computed to adjust the sample representation, before it is sent to the classifier:

$$u_t^x = -K_t^x e_t^x \tag{7}$$

where, $K_t^x \in \mathbb{R}^{d\times d}$ is a positive semi-definite feature gain matrix. The negative integer ensures that the correction moves in the opposite direction of the displacement, away from the benign centre. The corrected feature vector that is sent to the classifier is then:

$$\tilde{x}_t = x_t + u_t^x \tag{8}$$

where, $\tilde{x}_t$ replaces x_t as the input to $f\theta(\cdot)$ in Eq. (3). The gain matrix then adapts using a rank-one update:

$$K_{t+1}^x = K_t^x + \eta_x e_t^x (e_t^x)^T \tag{9}$$

where $\eta_x > 0$ is the gain adaptation rate. This rank-one update increases the gain along the feature dimensions, where displacement from the benign centre will occur the most. It also concentrates corrective efforts where the most drift has occurred.

Algorithm 1. Feature drift identification and control

Input: sample S_t, label y_t, current benign center μ_t^b, current gain K_t^x
Output: corrected feature $\tilde{x}_t$ (and updated μ_{t+1}^b, K_{t+1}^x)
1. Extract feature groups from S_t and form z_t.
2. Compute the scaled/normalized feature vector (Eq. (2)).
3. If $y_t = 0$(benign) then update the benign center (Eq. (5)).
Else: $\mu_{t+1}^b = \mu_t^b$.
4. Compute the feature error (Eq. (6)).
5. Compute the control input (Eq. (7)).
6. Compute the corrected feature (Eq. (8)).
7. Update the gain (Eq. (9)).
Pass $\tilde{x}_t$ to the next stage for probability estimation.

The label-conditional updating, that occurs in Step 3, ensures that the benign reference is not contaminated by phishing-labelled samples. The rank-one gain update, that occurs in Step 7, concentrates corrective efforts on feature dimensions that have been repeatedly displaced. This serves as the foundation for the stability analysis described in Section 3.6.

3.4 Threshold-level adaptive control

3.4.1 Dynamic decision threshold

Here, the detector applies a time-varying $\mathcal{T}_t$ to the $\mathcal{p}_t$:

$$\hat{\mathcal{y}}_t = \begin{cases} 1, & \mathcal{p}_t \geq \mathcal{T}_t \\ 0, & \mathcal{p}_t < \mathcal{T}_t \end{cases} \tag{10}$$

where, $\hat{\mathcal{y}}_t \in \{0, 1\}$ is the predicted label. A positive tracking error, relative to the target false-positive rate, will increase the threshold, while a negative tracking error will decrease it:

$$e_t^{\tau} = \phi_t^{fp} - \phi^{target} \tag{11}$$

where, ϕ_t^{fp} is the false-positive rate, estimated over a sliding window of length (W), and ϕ^{target} is the specified target. The integral-form threshold update rule is described in Eq. (12):

$$\tau_{t+1} = \tau_t + k_\tau \, e_t^\tau \tag{12}$$

where $K_\tau > 0$ is the threshold adaptation gain. In practice, τ_t is limited to a safe operating interval to prevent the occurrence of boundary positions that are below the transient noise.

Algorithm 2. Threshold adaptation (false-positive feedback control)

Inputs (at time t):
- p_t: predicted phishing probability for sample t
- $\hat{y}_t$: predicted label for sample t(from the current threshold)
- y_t: true label for sample t
- Sliding window of recent samples of length W(for estimating false-positive statistics)
- ϕ^{target}: target false-positive rate
- k_τ: threshold adaptation gain

Internal state:
- τ_t: current decision threshold

Outputs:
- None (the algorithm updates the internal threshold state τ_t)

Procedure:
1. Receive $(p_t, \hat{y}_t, y_t)$.
2. Update false-positive statistics over the most recent window of length W, and compute the observed false-positive rate ϕ_t^{fp}.
3. Compute threshold tracking error: $e_t^\tau = \phi_t^{fp} - \phi^{target}$:
4. Update the decision threshold: $\tau_{t+1} = \tau_t + k_\tau \, e_t^\tau$
5. Apply the updated threshold τ_{t+1}in the probability-based decision rule for subsequent samples:

$$\hat{y}_{t+1} = \begin{cases} 1, & p_{t+1} \geq \tau_{t+1} \\ 0, & p_{t+1} < \tau_{t+1} \end{cases}$$

End

Algorithm 2 then adjusts the threshold based on the observed false-positive feedback. The sliding window then stabilises the estimate against short-term fluctuations. It is noteworthy that the threshold does not need to be manually recalibrated when it is implemented.

3.5 Parameter-level adaptive control

The parameter-level AC block minimises a binary cross-entropy loss at each time step:

$$L_t = -[y_t log(p_t) + (1 - y_t) log(1 - p_t)] \tag{13}$$

where, $L_t \in \mathbb{R} \geqslant 0$ is the cross-entropy loss determined using $\tilde{x}_t$ from Eq. (8). The gradient of the loss with respect to θ is:

$$g_t = \nabla_\theta L_t \tag{14}$$

In a feature drift condition, g_t can indicate high variance or oscillatory behavior. This renders unconditioned gradient steps unreliable. Therefore, a parameter-level control input is used to moderate this:

$$u_t^\theta = -K_t^\theta g_t \tag{15}$$

where, $K_t^\theta \in \mathbb{R}^{p \times p}$ is a drift-sensitive parameter gain matrix. The gain is specified in a drift-damped form to ensure that the update magnitude is damped when the distributional drift is high and returns to a standard fixed-rate gradient step when the drift vanishes:

$$K_t^\theta = \frac{K_0^\theta}{1 + \eta_\theta D_t} \, I_p \, , \theta_{t+1} = \theta_t + u_t^\theta \tag{16}$$

where, $\eta_\theta > 0$ is the parameter-gain damping rate, D_t is the drift magnitude defined in Eq. (17), and K_0^θ is the nominal drift-free gain. When D_t increases, the effective gain K_t^θ decreases, thereby damping parameter updates under distributional drift. When $D_t \to 0$, K_t^θ approaches K_0^θ. The second part of Eq. (16) applies the resulting controlled update to θ.

Algorithm 3 combines drift information from the feature-level AC block, before sending it to the parameter update step, to retain learning strength while ensuring that the destabilising parameter does not increase when the inputs change.

Algorithm 3. Parameter control and update

1: Input: Controlled feature $\tilde{x}_t$, label y_t, parameters θ_t
2: Compute score and probability using Eqs. (3) and (4)
3: Evaluate loss L_t via Eq. (13)
4: Compute gradient g_t using Eq. (14)
5: Update parameter gain K_t^θ based on drift metrics (derived from feature block)
6: Compute control input u_t^θ using Eq. (15)
7: Update parameters with Eq. (16)

3.6 Drift quantification and stability function

The proposed AC-based phishing detection framework defines the D_t based on the movement of the benign feature centre ($\mu_t^b \in \mathbb{R}^d$) over a W to clearly calculate the distributional change:

$$D_t = \left\| \mu_t^b - \mu_{t-w}^b \right\|_2 \tag{17}$$

where, $D_t \in \mathbb{R} \geqslant 0$ is the scalar drift magnitude, μ_t^b is the current benign centre, μ_{t-w}^b is the benign centre over a W from the previous step, and $\| \cdot \|$ is the Euclidean norm. As a high D_t will trigger an equally strong adaptation in K_t^x, Eq. (16) is used to calculate an equally damped effective parameter gain. The composite stability monitoring function is described in Eq. (18):

$$V_t = (e_t^x)^T P e_t^x + \lambda_T (e_t^x)^2 + \lambda_\theta \|g_t\|_2^2 \tag{18}$$

where, $P \in \mathbb{R}^{d \times d}$ is the positive-definite weighting matrix and λ_τ, $\lambda_\theta > 0$ are the scalar weights. Here, Eq. (18) is used as an engineering monitoring metric, not a formal Lyapunov certificate. It aggregates feature displacement errors, threshold tracking errors, and gradient energy into a single scalar indicator of the state of the system-level adaptation. Eq. (19) describes a practical stability requirement:

$$V_{t+1} - V_t \leq -\delta V_t \tag{19}$$

where, $\delta > 0$ is a user-specified decay parameter. Eq. (19) requires the stability function (V_t) to consistently decrease when the drift is within bounds. It also determines the learning rates and gain matrices during the experiments.

4. EXPERIMENT SETUP

Three public datasets, comprising varying phishing attack entry points, were used. Dataset 1 was a phishing websites dataset compiled by Mohammad and McCluskey [31]. It contains the URL and HTML attributes of IP-based links, abnormal anchors, suspicious forwarding, and embedded redirects. Dataset 2 was a phishing websites dataset compiled by Vrbančič [32]. It contains more recent webpages, with longer lexical patterns, structural cues, and HTML variations. Dataset 3 was a phishing email and text message dataset compiled by Tijjani [33]. It contains messages categorised by urgency, authority, and persuasion type. Therefore, these three datasets comprise phishing behaviors across URL, webpage, email, and text messages.

All the samples were subjected to single feature extraction, which yielded URL, HTML, behavioral, and text-based vectors. These vectors were normalised and combined into a unified representation (z_t), as per Eq. (1). Each dataset was stratified and split 80:20 to preserve class balance. The input for the training phase was batch-based. This was done to enable the adaptive components to respond to the emerging feature patterns in sequence. It also mimics real drift conditions, in which the phishing patterns of URLs, webpages, and text messages change.

The performance metrics measured were accuracy, precision, recall, and F1. The false-positive behavior was used to determine the threshold control, while batch-level D_t was used to monitor the distributional changes. Lastly, the stability function value was used to track the behavior of the AC block across the batches.

The fixed-rule baseline model had the same MLP-based architecture, feature representation, and training configuration as the proposed AC-based phishing detection framework (Section 3.2). However, all three AC components were disabled to better isolate the effects of AC from the base classifier (MLP). A static threshold of 0.5 was used throughout the evaluation phase. Apart from that, standard gradient updates were also used *sans* gain modulation. Four additional static classifiers were also used, namely, random forest, gradient boosting, support vector machine with radial basis function kernel, and logistic regression.

All the models were tested using the same 80:20 stratified split of each dataset and the same unified feature representation. The hyperparameters for the random forest and gradient boosting classifiers were selected using grid search over estimator count {100, 200, 500}, maximum depth {5, 10, 20, None}, and minimum leaf samples {1, 2, 5}. Meanwhile, the regularisation parameter (C) was searched over {0.1, 1, 10, 100} and the kernel coefficient γ was searched over {scale, auto, 0.01, 0.1} for the support vector machine with radial basis function kernel classifier. Lastly, the regularisation strength was searched over {0.001, 0.01, 0.1, 1, 10} for the logistic regression classifier. All the grid searches used a held-out 10% validation subset that had been set aside before the 80:20 stratified split of each dataset. The MLP hyperparameters of the fixed-rule baseline model and the proposed AC-based phishing detection framework were identical. This was done to ensure that differences in performance could be conclusively attributed to the presence or absence of AC.

The control-specific hyperparameters for the AC layers were set to be constant across Dataset 1 [31], Dataset 2 [32], and Dataset 3 [33]. Table 2 lists all the values.

Table 2. Control hyperparameter values used in all experiments

Parameter	Value	Description
P	0.05	Benign-reference learning rate (Eq. (5))
η^x	0.01	Feature gain adaptation rate (Eq. (9))
η_θ	0.001	Parameter gain damping rate (Eq. 16)
k_τ	0.1	Threshold adaptation gain (Eq. (12))
W	50 samples	Sliding window for FP rate and drift (Eqs. (11) and (17))
ϕ^{target}	0.02	Target false-positive rate (Eq. (11))
λ_τ	1.0	Stability weight for threshold error (Eq. (18))
λ_θ	0.5	Stability weight for gradient energy (Eq. (18))
Δ	0.001	Stability decay parameter (Eq. (19))
K_0^x	0.1 I_d	Initial feature gain matrix
K_0^θ	0.01 I_p	Nominal parameter gain (Eq. (16))

5. RESULTS AND ANALYSES

5.1 Evaluation strategy and overall performance

5.1.1 Evaluation protocol

The purpose of the evaluation was to determine how the proposed AC-based phishing detection framework adjusted its behavior in relation to the changing input patterns of Dataset 1, Dataset 2, and Dataset 3. All the datasets were subjected to single feature extraction, which yielded a single-structured representation comprising URL, HTML, behavioral, and text cues. The data was segregated into batches to approximate real-world drift conditions and observe the performance of the proposed AC-based phishing detection framework across different input distributions. Each batch of data passed through the feature-, threshold-, and parameter-levels' control units. The effect of each level was determined by examining the corrected features, threshold movements, and controlled parameter updates. Apart from that, the accuracy, precision, recall, and F1 score were recorded, alongside the batch-level drift magnitudes, false-positive rates, and stability function values.

As shown in Figure 2, the workflow processes Dataset 1, Dataset 2, and Dataset 3 through a common feature extractor, followed by parallel baseline and adaptive-control evaluation paths. Batch-level performance, drift, and stability metrics are then logged for comparative analysis.

5.1.2 Performance comparison with existing studies

The fixed-rule baseline model processed the unified feature space sans AC. As such, its performance plummeted as the URL, HTML, and behavioral cues drifted across the batches. However, the feature correction, threshold offsets, and controlled parameter updates used in the proposed AC-based phishing detection framework yielded higher accuracy, precision, recall, and F1. The most significant improvements were seen when processing Dataset 1 and Dataset 2, where the feature-level AC block decreased the structural drift most effectively. When processing Dataset 3, the recall improved significantly as the feature controller corrected the linguistic variations.

As shown in Figure 3, a comparison of the classification performances of the strongest published results and the proposed AC-based phishing detection framework across Dataset 1, Dataset 2, and Dataset 3.

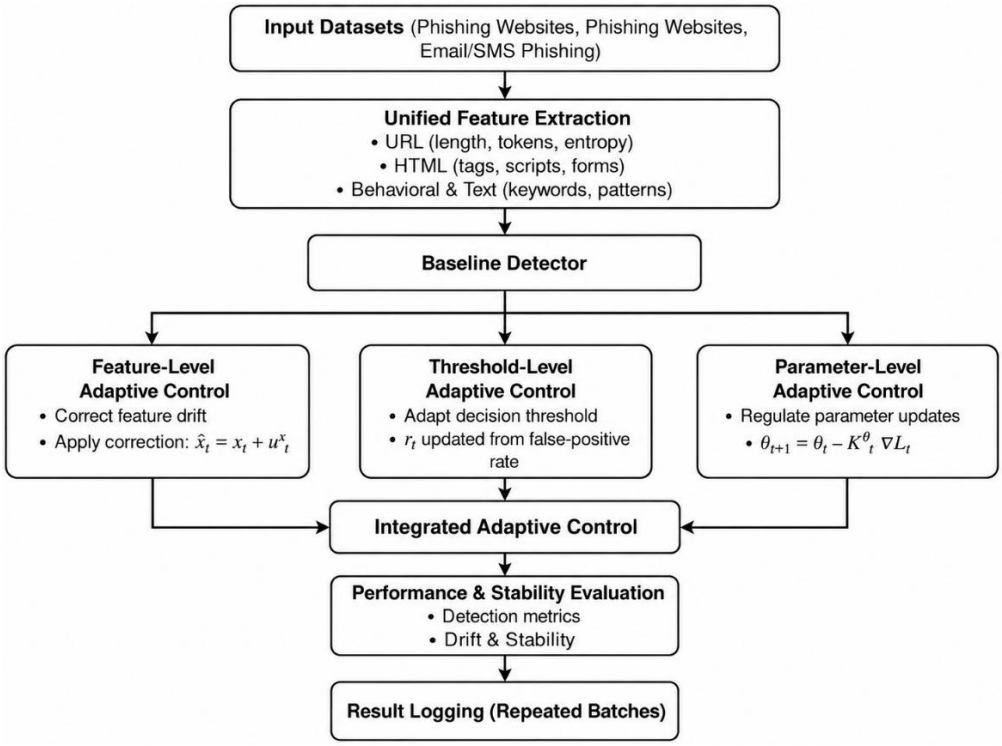

Figure 2. Evaluation workflow of the proposed framework.

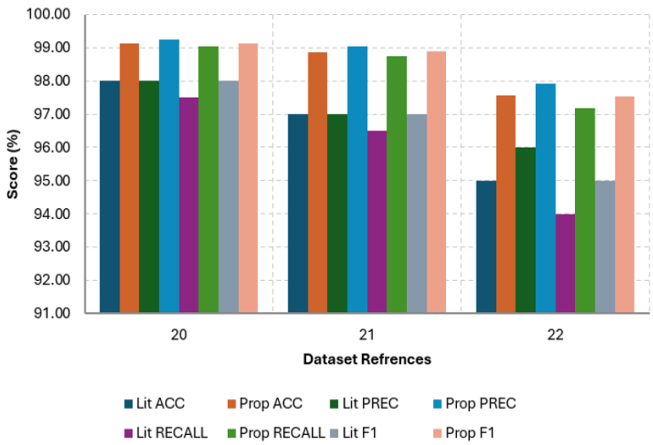

Figure 3. Comparison of literature and proposed method performance across datasets

Table 3. Comparison between strongest literature results and the proposed adaptive control method

Ref	M	Acc	Prec	Rec	F1
[31]	Pub*	97–99	97–99	97–98	97–98
Prop	99.12	99.24	99.05	99.14	
[32]	Pub	96–99	96–98	96–98	96–98
Prop	98.87	99.03	98.74	98.88	
[33]	Pub	95–97	95–97	95–97	95–97
Prop	97.56	97.92	97.18	97.54	

Note: * Pub: ranges as reported in prior studies; protocols may differ. Prop: proposed method (this work).

As seen in Figure 3, the proposed AC-based phishing detection framework achieved the highest and most consistent performance in terms of accuracy, precision, recall, and F1. Operationally, this balanced metric profile is more valuable than isolated accuracy improvements as it reflects reliable detection behavior, instead of performance that has been optimised for a single criterion using a fixed testing dataset.

Table 3 compares the strongest literature results with the proposed adaptive control-based phishing detection framework across the three evaluation datasets.

5.2 Adaptive control behavior under feature drift

5.2.1 Feature-level adaptive control

The feature-level controller decreased the size of the displacement between the incoming feature vectors and the benign centre. Therefore, it stabilised the decision boundaries when the URL and HTML patterns changed across the batches. Apart from that, the drift responses confirm that the adjustment decreased the effects of irregular lexical and structural changes. It also improved the separation between phishing and benign samples when the distributions were changing.

Table 4 describes how drift behavior changed after applying feature-level control, and Figure 4 presents the drift magnitude across batches for both stages. The behavior before control showed larger changes in the benign feature centre. The adaptive feature block reduced these movements and stabilised the drift curve. This stability supported more consistent feature representations during detection.

The batch-wise D_t (Eq. (17)) of Dataset 1 before and after the feature-level AC was implemented.

Table 4. Feature drift behavior before and after feature-level control

Ref	Before Control (High Drift)	After Control (Reduced Drift)
[31]	Elevated shift in URL and HTML cues	Lower displacement from benign centre
[32]	Irregular lexical and structural variations	Stable movements across Batches
[33]	Fluctuations in behavioural and text cues	Reduced deviation during sequences

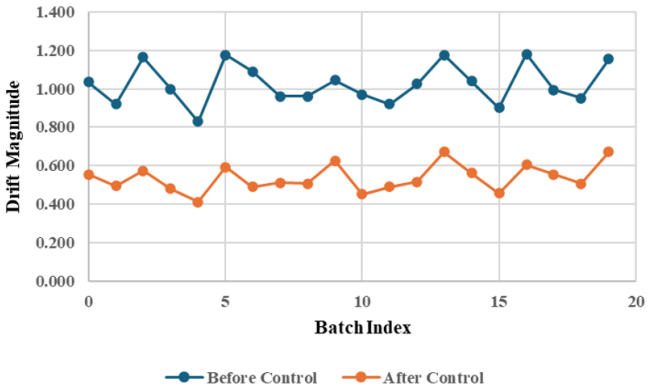

Figure 4. Drift response before and after feature-level control

As seen in Figure 4, before the AC were implemented, the inter-batch variation of the Dt was significant, particularly in the early batches of Dataset 1 and Dataset 2. However, after the feature-level AC block was implemented, the drift curve was smoother and the trajectory of the amplitude was lower. This decrease was consistently observed in Dataset 1, Dataset 2, and Dataset 3. Therefore, the mechanism was generalised across URL-, webpage-, and text-based phishing environments.

5.2.2 Threshold-level adaptive control

The interactions of the feature- and threshold-level AC blocks were assessed by comparing the results when both the AC blocks were implemented simultaneously and when each AC block was implemented on its own. When both the AC blocks were implemented, the feature-level AC block stabilised the input representations, while the threshold-level AC block reinforced it. This was done by adjusting the decision boundary in relation to the residual false-positive behavior that prevailed even after the feature correction. Across most of the batches of Dataset 1 and Dataset 2, the false-positive rates were lower, and the recall was higher when both the AC blocks were implemented. Therefore, both the AC blocks are complementary, rather than redundant. In Dataset 3, however, where false-positive monitoring cannot be performed, the feature-level AC block was the primary stabiliser, while the threshold-level AC block had no effect due to the absence of benign samples. Therefore, the performance of the proposed AC-based phishing detection framework plummets gracefully when one of the AC blocks cannot be implemented.

5.2.3 Effect of the threshold-level adaptive control block

The threshold-level AC block modified the decision boundary in relation to variations in the false-positive rate at the batch-level. In Dataset 1 and Dataset 2, both of which contain phishing and benign samples, the threshold-level AC block increased the threshold when the false-positive ratio exceeded the target and vice versa. Therefore, it adjusts the sensitivity without destabilising the results. In Dataset 3, which contains only phishing messages, the threshold-level AC block yielded consistent recall across linguistic variations. Therefore, it is still valuable, even when false-positive monitoring is unavailable.

The batch-wise threshold results are presented in Table 5, showing that the adaptive threshold remained close to the fixed threshold of 0.50 while adjusting slightly across batches in response to changes in the false-positive rate.

A comparison of the evolution of the τt in Dataset 1, Dataset 2, and Dataset 3, and the fixed baseline threshold (0.50).

Figure 5 illustrates the closed-loop threshold behavior. As seen, in the initial batches, where the false-positive rate exceeded the target, the threshold was > 0.50 (Eq. (12)). However, as the false-positive rate stabilised, the threshold decreased to restore sensitivity. Furthermore, as the trajectory was smooth and bounded, it proves that the threshold-level AC block dampens short-term noise without producing unstable oscillations. This would decrease the likelihood of unnecessary quarantines occurring in operational filtering systems.

Table 5. Batch-wise threshold adjustment using Dataset 1 and Dataset 2

Batch	FP Rate	Adaptive Threshold	Fixed Threshold
1	0.031	0.5015	0.50
2	0.028	0.5029	0.50
3	0.021	0.5010	0.50
4	0.018	0.4991	0.50
5	0.019	0.4980	0.50

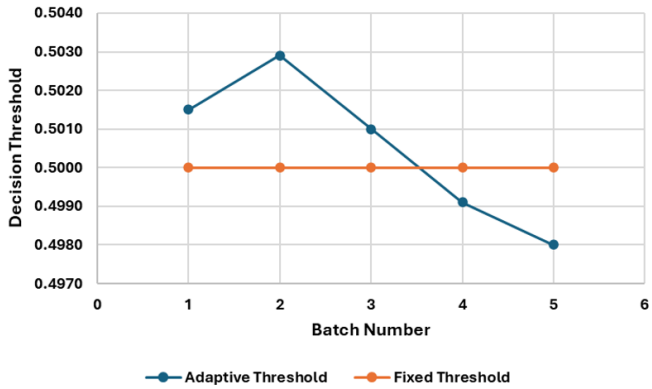

Figure 5. Adaptive movement of the decision threshold across batches

5.2.4 Parameter-level control and stability behavior

The V_t (Eq. (18)) was monitored across all the batches to evaluate the combined effect of feature correction, threshold adjustment, and parameter updates. In Dataset 1 and Dataset 2, the stability decreased across the batches. This indicates that the AC blocks compensate for changing URL and HTML structures. Furthermore, sans adaptation, the stability curve was jagged due to the uneven movement of the benign centre, the irregular false-positive behavior, and the unstable gradients. However, when all three AC blocks were implemented, the stability curve was smoother and less jagged. Apart from that, in Dataset 1, Dataset 2, and Dataset 3, the stability decreased monotonically or nearly-monotonically. Therefore, the controlled drift behavior matched the improvements in accuracy, precision, and recall. Comparison with fixed-threshold and static-update studies is integrated into the stability analysis below.

Extant studies, that used fixed thresholds or static parameter updates, reported erratic phishing detection behavior when the feature distribution changed. They also reported higher false-positive rates and lower recall scores when the URL, HTML, or text patterns changed across the batches. Meanwhile, the proposed AC-based phishing detection framework yielded smoother decisions in Dataset 1, Dataset 2, and Dataset 3. More specifically, the feature-level AC block decreased drift-induced displacement, the threshold-level AC block stabilised

the decision boundary, while the parameter-level AC block attenuated gradient oscillations. The comparative patterns across studies are summarised in Table 6.

Figure 6 proves the stability of the proposed AC-based phishing detection framework. Instability values were calculated as (1 − F1). More specifically, the instability values were 0.0086, 0.0112, and 0.0246, respectively, in Dataset 1, Dataset 2, and Dataset 3. Therefore, its detection behavior was consistent across all phishing modalities. Meanwhile, extant studies reported higher detection behavior instability. Therefore, fixed-rule models are more sensitive to changes in distribution.

Table 6. Contrast between behavior reported in prior studies and the adaptive framework

Behavior Aspect	Prior Studies	Adaptive Framework
Response to feature drift	Sharp drops in accuracy, unstable boundary shifts	Controlled adjustments with reduced drift displacement
Threshold behaviour	Fixed threshold causing false-positive spikes	Smooth adaptive movement aligned with batch behavior
Parameter updates	Oscillatory gradients under changing patterns	Moderated gradient steps providing stable updates
General stability	Sensitive to structural or linguistic variations	Consistent detection across batches

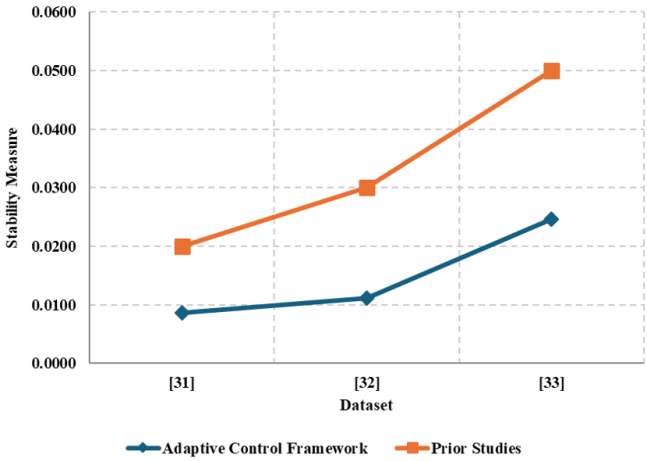

Figure 6. Instability comparison across evaluation datasets

5.3 Dataset-wise, ablation, and error analysis

5.3.1 Dataset-wise performance behavior

Dataset 1, Dataset 2, and Dataset 3 were subjected to unified feature representation. The accuracy, precision, recall, and F1 of the proposed AC-based phishing detection framework were, respectively, 99.12, 99.24, 99.05, and 99.14% in Dataset 1; 98.87, 99.03, 98.74, and 98.88% in Dataset 2; and 97.56, 97.92, 97.18, and 97.54% in Dataset 3. Therefore, the feature-level AC block adjusted the representations in relation to the dataset-specific drift patterns, the threshold-level AC block regulated the false-positive behavior in the two datasets that contain benign samples, and the parameter-level AC block shaped the learning response during the irregular update sequences.

As seen in Figure 7, the performance of the proposed AC-based phishing detection framework was consistently high and balanced in Dataset 1, Dataset 2, and Dataset 3. The precision was slightly higher than the recall in all three datasets. This indicates that the threshold-level AC block moderates the false-positive rate. Furthermore, the narrow spread between the accuracy, precision, recall, and F1 confirms that the proposed AC-based phishing detection framework does not sacrifice recall for accuracy or vice versa. Table 7 reports the dataset-wise accuracy, precision, recall, and F1-score obtained by the proposed framework.

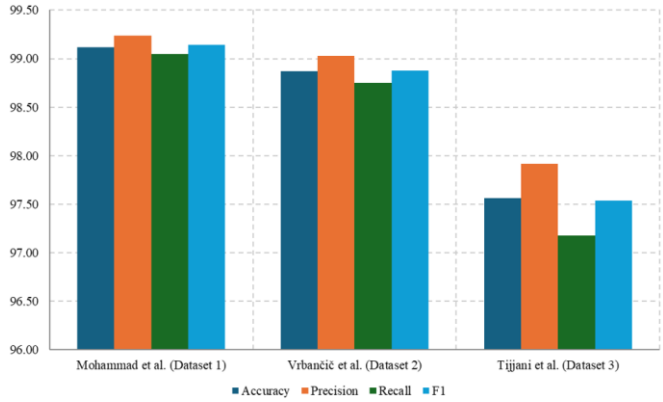

Figure 7. Dataset-wise performance of the proposed framework.

Table 7. Performance of the proposed system across evaluation datasets

Ref	ACCU	Precision	Recall	F1
[31]	99.12%	99.24%	99.05%	99.14%
[32]	98.87%	99.03%	98.74%	98.88%
[33]	97.56%	97.92%	97.18%	97.54%

5.3.2 Ablation analysis of adaptive control blocks

The purpose of the ablation study was to isolated the effect of each AC block. The feature-level AC block decreases distributional drift and improves phishing-benign segregation in the URL and HTML cues. Meanwhile, the threshold-level AC block stabilises false-positive behavior and smooths decision boundaries in datasets that contain both sample types. Lastly, the parameter-level AC block reduces gradient oscillations and produces more stable weight updates. Therefore, in combination, they yield the highest accuracy, precision, recall, and F1 scores and the most stable behavior under drift conditions in Dataset 1, Dataset 2, and Dataset 3. Table 8 summarizes the ablation behavior of the feature-level, threshold-level, and parameter-level adaptive control components, and Table 9 presents the main misclassification patterns observed across the three datasets.

Table 8. Ablation behavior of adaptive components

Component Tested	Observed Behavior
Feature-level control only	Reduced drift across batches and improved separation in URL and HTML cues.
Threshold-level control only	Stabilised false-positive behavior and produced smoother decision paths in mixed datasets.
Parameter-level control only	Reduced gradient oscillation and produced more stable updates in both webpage datasets.
All components combined	Highest accuracy, precision, recall and F1; most SB under drift.

Table 9. Misclassification patterns observed across datasets

Dataset	Observed Misclassification Pattern
[31]	Benign pages with redirect chains and embedded scripts caused false positives.
[32]	Long lexical strings and nested HTML tags in benign pages produced confusion.
[33]	Alert-style phishing messages resembled service notifications and caused false negatives.

5.3.3 Misclassification and error patterns

The false positives observed in Dataset 1 were primarily due to benign webpages that contain long, redirect chains or embedded scripts that resembled phishing patterns. Meanwhile, Dataset 2 contains benign pages that have extended lexical forms or nested HTML tags, which led to, occasional, confusion due to their structural similarity with phishing pages. Lastly, the false negatives observed in Dataset 3 were primarily due to messages that used alert-style phrasing, which is common in legitimate service notifications. However, it is noteworthy that the feature-level AC block mitigated these errors by adjusting the representations in relation to changes in the lexicon or structure, the threshold-level AC block stabilised the decisions when the benign samples fluctuated across the batches, and the parameter-level AC block decreased gradient noise in samples that contain variable cue distributions.

5.4 Robustness, runtime, and generalisation

5.4.1 Robustness under drift conditions

In Dataset 1, Dataset 2, and Dataset 3, the observed drifts were primarily due to changes in the URL structure and embedded redirect behavior; changes to longer lexical forms, HTML depth, and webpage composition; and changes in the linguistic markers used in phishing phrases, respectively. However, it is noteworthy that the feature-level AC block decreased the size of the displacement by adjusting the representations in relation to movements in the benign sample, the threshold-level AC block stabilised the decision boundaries when the benign cues changed in Dataset 1 and Dataset 2, and the parameter-level AC block decreased gradient fluctuations during irregular input sequences. Therefore, in combination, they successfully maintained stable predictions under ever-changing drift conditions in Dataset 1, Dataset 2, and Dataset 3. Table 10 summarizes the observed drift behavior and the corresponding adaptive response of each control component.

5.4.2 Runtime and computational behavior

Two distinct measurement conventions were used to assess runtime. Firstly, a per-sample inference delay, which comprised the forward pass of the base classifier and the lightweight control updates, was executed at prediction time. This was found to be 12 microseconds per sample. Secondly, a drift-handling delay, which is the end-to-end time required to detect drifts and apply corrective actions at the pipeline level. The ±18 seconds reported by Reda et al. [19] was for the latter, not the per-request inference time. Their serving performance was under 50 microseconds at p99. Therefore, as these two figures measure different operational stages, they cannot be directly compared. Table 11 reports the runtime behavior of the proposed method and compares it with the available operational results reported in the literature.

5.4.3 Cross-dataset generalisation behavior

The proposed AC-based phishing detection framework processed samples from three structurally distinct datasets using the same unified feature space. This ensured stable predictions even when the structure of the inputs changed. More specifically, the feature-level AC block decreased the size of the displacement from distinct lexical or structural cues, the threshold-level AC block adjusted the decisions when the benign cues changed in Dataset 1 and Dataset 2, and the parameter-level AC block maintained stable updates when the sequence of the cues varied. Therefore, in combination, they decreased sensitivity to dataset-specific structures and supported generalisation across URL-, webpage-, and text-based phishing environments. It is noteworthy that the present study did not perform cross-dataset evaluation, such as training using one dataset or testing on a structurally distinct one. As such, cross-dataset evaluation is identified as essential future research in Section 6.2.

Table 12 summarizes the cross-dataset generalisation behavior and the adaptive response observed across URL-, webpage-, and text-based phishing environments.

Table 10. Drift behavior and the response of adaptive components

Ref	Observed Drift	Adaptive Response
[31]	URL and redirect variations	Feature-level control reduced displacement; threshold control stabilised false positives.
[32]	Long lexical forms and deeper HTML structures	Feature-level control stabilised representation; parameter control reduced gradient noise.
[33]	Shifts in linguistic patterns and behavioral cues	Feature-level control corrected representation; parameter control improved sequence stability.

Table 11. Runtime behavior of the proposed method compared with available results

System	Mean inf (ms)	p95 (ms)	p99 (ms)	Thrpt (req/s)	Upd/batch (s)	Drift resp (s)	Peak RAM (MB)
BL	10.5	15.0	22.0	95	0.00	N/A	420
P-Inf	12.0	18.0	26.0	82	0.00	N/A	460
P-Upd	12.0	18.0	26.0	80	0.35	1.20	480
Reda	N/R	N/R	<50	2300	N/R	18.0	N/R

Note: BL = baseline (fixed threshold, no control), P-Inf = proposed (control during inference only), P-Upd = proposed (control + online model update per batch), N/R = not reported. Mean/p95/p99 refer to per-sample inference latency (prediction path).

Table 12. Cross-dataset generalisation behavior and adaptive response

Ref	Feature	Behavior
[31]	URL and HTML cues with redirects and embedded scripts	SB due to reduced drift and controlled threshold updates.
[32]	Longer lexical strings and deeper HTML structures	Consistent predictions supported by feature correction and stable parameter updates.
[33]	Linguistic and behavioral phishing cues	Stable recall due to corrected representation and controlled learning response.

Table 13. Comparison of the proposed adaptive system with existing phishing detection models

Ref	ACCU	Precis	Recall	F1	AUC	Latency
[27]	99.44	99.98	–	99.32	–	–
[26]	99.55	99.61	99.55	99.24	–	–
[17]	99.45	–	–	–	–	–
[16]	99.00	–	–	–	–	–
[15]	99.17	98.81	–	–	–	–
[25]	97.82	99.74	98.89	99.31	–	–
[19]	–	–	–	99.50	–	18000 ms, p99 < 50 ms
[30]	95.00	96.00	94.00	–	0.96	–
Proposed	99.12	99.24	99.05	99.14	0.997	12 ms

5.5 Performance comparison with state-of-the-art models

The performance of extant phishing detection models, that use fixed thresholds or static parameter updates, often plummeted when the feature distributions changed. However, the performance of the proposed AC-based phishing detection framework remained consistent throughout feature corrections, threshold adaptations, and controlled parameter updates. Table 13 compares the performance of the proposed AC-based phishing detection framework with that of extant phishing detection models.

Figure 8 positions the proposed AC-based phishing detection framework within the current performance landscape. As seen, it was accurate 99.12% of the time. As previously mentioned, the delay figures seen in Table 13, more specifically, the 12 ms per-sample inference time of the proposed AC-based phishing detection framework and the 18 second drift-handling delay reported by Reda et al. [19] measure different operational stages. Therefore, they cannot be directly compared. It is, however, noteworthy that the broader value of the proposed AC-based phishing detection framework lies in its ability to yield competitive detection accuracy as well as controlled adaptation, stable threshold behavior, and bounded parameter dynamics.

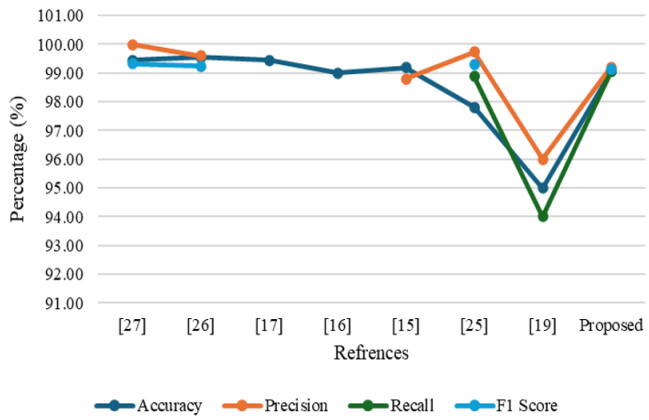

Figure 8. A comparison of the accuracy, precision, recall, and F1 scores of the best extant phishing detection models and the proposed AC-based phishing detection framework

5.6 Overall stability behavior

In Dataset 1, Dataset 2, and Dataset 3, the proposed AC-based phishing detection framework was able to yield steady decision patterns sans abrupt fluctuations, even when the distribution of the URL, HTML, or text cues changed across the batches. It was also able to yield uniform prediction patterns when the structure and lexicon changed. Therefore, the proposed AC-based phishing detection framework does not rely on any single dataset-specific feature. This consistent performance matches the results of the quantitative stability-function analysis (Section 5.6) and affirms the detection reliability of the proposed AC-based phishing detection framework. The supporting system-level stability observations are summarized in Table 14.

Table 14. System-level stability behavior across datasets

Ref	Observation
[31]	Predictions remained uniform across batches despite changes in URL and HTML structures.
[32]	Stable output trends during lexical and webpage-structure variations across the data sequence.
[33]	Consistent detection behavior across different message styles and linguistic patterns.

5.7 Critical interpretation of reported performance

The accuracy, precision, recall, and F1 score of the proposed AC-based phishing detection framework in Dataset 1, Dataset 2, and Dataset 3 were high. It reached 99.12% in Dataset 1, 98.87% in Dataset 2, and 97.56% in Dataset 3. It is noteworthy that several dataset-level factors contributed to these figures. Firstly, Dataset 1, Dataset 2, and Dataset 3 are publicly available. They are curated datasets of phishing and benign samples that have been labelled in controlled conditions. Furthermore, Dataset 1 and Dataset 2 contain structured URL and HTML attributes, with relatively high inter-class separability, while Dataset 3 contains greater linguistic variability as it is categorised according to urgency, authority, and persuasion, which would explain its slightly lower scores. Secondly, as these datasets are structured and curated, the accuracy, precision, recall, and F1 score of the proposed AC-based phishing detection framework reflect its

performance under controlled input conditions. Therefore, its performance should not be generalised to live deployment environments, where the phishing content is more diverse, the labels are noisier, and the benign traffic is more contextually variable.

In terms of the integrity of the evaluation, the validation subset was set aside before any model fitting, feature normalisation, threshold initialisation, or adaptive parameter update. The $S(\cdot)$ was also fitted exclusively on the training set and frozen before inference. Therefore, the validation subset statistics did not influence any normalisation or scaling parameters. Furthermore, the benign reference centre (μtb) was only updated on the training set. Therefore, none of the test-set label information could have influenced any of the AC components. These procedural safeguards were undertaken to substantially decrease the risk of information leakage. However, as the samples in Dataset 1, Dataset 2, and Dataset 3 were drawn from related phishing domains, they have overlapping structural feature vocabularies. As such, there is a possibility that pre-existing inter-sample correlations led to optimistic performance estimates. Therefore, this uncertainty is a, potential, limitation of the present study.

Apart from that, the present study did not perform cross-dataset evaluation, such as training using one dataset or testing on a structurally distinct one. Therefore, future research could assess cross-dataset generalisation by training the proposed AC-based phishing detection framework using URL-based phishing data and evaluating its performance on text-based datasets. Temporally separated partitions, that simulate concept drifts at longer time periods, could also be constructed to evaluate the performance of the proposed AC-based phishing detection framework. It is believed that such evaluations would affirm the deployment-level robustness of the proposed AC-based phishing detection framework better than benchmarked accuracy alone.

Therefore, the reported results are best interpreted as evidence that the proposed AC-based phishing detection framework yields stable and consistent performance solely under the feature distributions of D Dataset 1, Dataset 2, and Dataset 3. Its primary contribution is its ability (1) to stably detect phishing across sequential batches under feature drift conditions, (2) to control and regulate false-positive behavior, and (3) to maintain consistent gradient dynamics properties that are relevant for security engineering systems that are required to operate reliably under ever-changing input conditions.

6. CONCLUSION

The present study proposed an AC-based phishing detection framework that simultaneously executes feature-level correction, threshold-level adaptation, and parameter-level controlled updates to yield stable phishing attack detection in environments with ever-changing feature distributions. Its three AC components, namely, its feature-, threshold-, and parameter-level AC blocks, adhere to structured feedback laws and are monitored using a composite stability function. This yields a quantifiable basis with which to manage the adaptation state of the proposed AC-based phishing detection framework across sequential processing batches.

Three public benchmark datasets, comprising URL-, webpage-, and text-based phishing artefacts, were used to evaluate the performance of the proposed AC-based phishing detection framework. It consistently outperformed the fixed-rule baseline model in terms of accuracy, precision, recall, and F1 score. Its instability values were also lower than that of extant phishing detection models under comparable drift conditions. Furthermore, the results of the ablation analysis affirmed that, when all three AC blocks are implemented simultaneously, they yield stronger and more consistent performance than any single AC block on its own.

Apart from that, the benefits of the proposed AC-based phishing detection framework are most evident at a system level. More specifically, by replacing reactive model updates with coordinated closed-loop controls, it facilitates continuous, auditable, and operationally reliable detection behavior in existing security infrastructure, such as email filtering systems and network monitoring platforms. Lastly, the parameter-level AC block decreases the need for full model re-training by adding new threats via drift-conditioned gradient steps.

6.1 Practical deployment considerations

The current evaluation is conducted on public benchmark datasets, which supports controlled comparison and reproducibility but does not replicate operational deployment conditions. In practice, the proposed framework is designed to function as the detection and adaptation layer within a broader security pipeline, such as an organisational email filtering system, a secure web gateway, or a network monitoring platform. At the input stage, raw artefacts forwarded by a mail transfer agent or network proxy would be processed through the unified feature extractor before entering the adaptive detection module. The feature-level correction mechanism would stabilise input representations as lexical and structural phishing patterns evolve across traffic batches, while the threshold controller would maintain false-positive rates within operationally specified bounds. Per-sample inference latency of 12 milliseconds is compatible with synchronous filtering workflows. Empirical validation under live deployment conditions including real traffic volumes, adversarial inputs, and operational feedback loops remains a necessary direction for future work.

6.2 Limitations and scope of experimental evaluation

The experimental evaluation uses three datasets from the phishing detection domain. While these provide meaningful variation in feature modality, they remain within a single threat class and were collected under curated conditions. The evaluation demonstrates the adaptive framework's behavior across different phishing feature types and supports controlled analysis of drift response within the phishing detection problem, but it does not establish generalisation to the full range of threat classes, attack surfaces, or input distributions present in operational security environments. Broader validation would require live organisational email traffic, web gateway logs, enterprise network flow data, or cross-domain datasets spanning multiple attack categories. Cross-dataset transfer experiments and temporal validation using chronologically separated partitions are identified as necessary future work. Overstating the generality of benchmark results would misrepresent the readiness of a detection system for heterogeneous and adversarially dynamic deployment environments.

Future extensions may include multi-modal analysis,

refined drift metrics, lightweight control modules for constrained hardware, adversarial robustness analysis, and characterisation of longer-term behavioral drift in attacker strategies.

DATA AVAILABILITY STATEMENT

The datasets used in this study are publicly available from the UCI Machine Learning Repository [31], Mendeley Data [32], and Kaggle [33].

STATEMENT ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE

Generative AI tools were used only for language polishing, formatting support, and editorial consistency checks. They were not used to generate research data, fabricate results, create references, or determine scientific conclusions. All scientific content, data interpretation, and final manuscript decisions remain the responsibility of the authors.

REFERENCES

[1] Ali, M.M., Mohd Zaharon, N.F. (2024). Phishing—A cyber fraud: The types, implications and governance. International Journal of Educational Reform, 33(1): 101-121. https://doi.org/10.1177/10567879221082966

[2] Bermani, A.K., Al-Salih, A.M., Jabir, H.A. Privacy-preserving over encrypted data for web page phishing detection. Journal of Discrete Mathematical Sciences and Cryptography, 28(4-B): 1413-1424. https://doi.org/10.47974/JDMSC-2287

[3] Amiri, Z., Heidari, A., Navimipour, N.J., Unal, M., Mousavi, A. (2024). Adventures in data analysis: A systematic review of deep learning techniques for pattern recognition in cyber-physical-social systems. Multimedia Tools and Applications, 83(8): 22909-22973. https://doi.org/10.1007/s11042-023-16382-x

[4] Kheruddin, M.S., Zuber, M.A.E.M., Radzai, M.M.M. (2024). Phishing attacks: Unraveling tactics, threats, and defenses in the cybersecurity landscape. Authorea Preprints. https://doi.org/10.22541/au.170534654.48067877v1

[5] Putra, F.P.E., Zulfikri, A., Arifin, G., Ilhamsyah, R.M. (2024). Analysis of phishing attack trends, impacts and prevention methods: literature study. Brilliance: Research of Artificial Intelligence, 4(1): 413-421. https://doi.org/10.47709/brilliance.v4i1.4357

[6] Taherdoost, H. (2024). Insights into cybercrime detection and response: A review of time factor. Information, 15(5): 273. https://doi.org/10.3390/info15050273

[7] Bauskar, S.R., Madhavaram, C.R., Galla, E.P., Sunkara, J.R., Gollangi, H.K. (2024). AI-driven phishing email detection: Leveraging big data analytics for enhanced cybersecurity. Library Progress International, 44(3): 7211-7224. https://doi.org/10.2139/ssrn.4980647

[8] Opara, C., Chen, Y., Wei, B. (2024). Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. Expert Systems with Applications, 236: 121183. https://doi.org/10.1016/j.eswa.2023.121183

[9] Duah, D., Sarfo, B.K.O., Boakye, C., Duku, D. (2025). Phish or legit? Quantifying web-based threat signals through predictive analytics and feature attribution. International Journal of Professional Business Review, 10(5): e05472. https://doi.org/10.26668/businessreview/2025.v10i5.5472

[10] Innab, N., Abdelgader, A., Awad, M., Abu-Zanona, M., Elzaghmouri, B., Zawaideh, F., Alawneh, M. (2024). Phishing attacks detection using ensemble machine learning algorithms. Computers, Materials, & Continua, 80(1): 1325. https://doi.org/10.32604/cmc.2024.051778

[11] Xing, W., Shen, J. (2024). Security control of cyber–physical systems under cyber attacks: A survey. Sensors, 24(12): 3815. https://doi.org/10.3390/s24123815

[12] Talpur, K., Hasan, R., Gocer, I., Ahmad, S., Bhuiyan, Z. (2025). AI in maritime security: Applications, challenges, future directions, and key data sources. Information, 16(8): 658. https://doi.org/10.3390/info16080658

[13] Atakari, C. (2024). A deep learning-based security model for ERP-integrated IoT in national defense manufacturing environments. International Journal of Emerging Trends in Computer Science and Information Technology, 5(3): 90-98. https://ijetcsit.org/index.php/ijetcsit/article/view/371

[14] Shafin, S.S. (2025). An explainable feature selection framework for web phishing detection with machine learning. Data Science and Management, 8(2): 127-136. https://doi.org/10.1016/j.dsm.2024.08.004

[15] Das Guptta, S., Shahriar, K.T., Alqahtani, H., Alsalman, D., Sarker, I.H. (2024). Modeling hybrid feature-based phishing websites detection using machine learning techniques. Annals of Data Science, 11(1): 217-242. https://doi.org/10.1007/s40745-022-00379-8

[16] Zara, U., Ayyub, K., Khan, H.U., Daud, A., Alsahfi, T., Ahmad, S.G. (2024). Phishing website detection using deep learning models. IEEE Access, 12: 167072-167087. https://doi.org/10.1109/ACCESS.2024.3486462

[17] Kustiawan, Y.A., Ghauth, K.I. (2025). Evaluating the impact of feature engineering in phishing URL detection: A comparative study of URL, HTML, and derived features. IEEE Access, 13: 126756-126768. https://doi.org/10.1109/ACCESS.2025.3579223

[18] Mohamed, N. (2025). Artificial intelligence and machine learning in cybersecurity: A deep dive into state-of-the-art techniques and future paradigms. Knowledge and Information Systems, 67(8): 6969-7055. https://doi.org/10.1007/s10115-025-02429-y

[19] Reda, A., Taie, S.A., Shaheen, M.E. (2025). Hybrid MLOps framework for automated lifecycle management of adaptive phishing detection models. Scientific Reports, 15: 38478. https://doi.org/10.1038/s41598-025-23600-z

[20] Alazaidah, R., Al-Shaikh, A., Al-Mousa, M.R., Khafajah, H., et al. (2024). Website phishing detection using machine learning techniques. Journal of Statistics Applications & Probability, 13(1): 119-129. https://doi.org/10.18576/jsap/130108

[21] Khan, M.A., Hossen, M.S., Bhuiyan, M.A., Hossain, M.L. (2024). Phishing website detection system using machine learning. Journal of Networking and Communication Systems, 7(2): 1-56.

https://doi.org/10.46253/jnacs.v7i2.a1
[22] Alzboon, M.S., Al-Batah, M.S., Alqaraleh, M., Alzboon, F., Alzboon, L. (2025). Phishing website detection using machine learning. SAP Gamification and Augmented Reality, 3: 81-81. https://doi.org/10.56294/gr202581
[23] Ujah-Ogbuagu, B.C., Akande, O.N., Ogbuju, E. (2024). A hybrid deep learning technique for spoofing website URL detection in real-time applications. Journal of Electrical Systems and Information Technology, 11(1): 7. https://doi.org/10.1186/s43067-023-00128-8
[24] Ghalechyan, H., Israyelyan, E., Arakelyan, A., Hovhannisyan, G., Davtyan, A. (2024). Phishing URL detection with neural networks: An empirical study. Scientific Reports, 14(1): 25134. https://doi.org/10.1038/s41598-024-74725-6
[25] Said, Y., Alsheikhy, A.A., Lahza, H., Shawly, T. (2024). Detecting phishing websites through improving convolutional neural networks with Self-Attention mechanism. Ain Shams Engineering Journal, 15(4): 102643. https://doi.org/10.1016/j.asej.2024.102643
[26] Chinta, P.C.R., Moore, C.S., Karaka, L.M., Sakuru, M., Bodepudi, V., Maka, S.R. (2025). Building an intelligent phishing email detection system using machine learning and feature engineering. European Journal of Applied Science, Engineering and Technology, 3(2): 41-54. https://doi.org/10.59324/ejaset.2025.3(2).04
[27] Barik, K., Misra, S., Mohan, R. (2025). Web-based phishing URL detection model using deep learning optimization techniques. International Journal of Data Science and Analytics, 20(5): 4449-4471. https://doi.org/10.1007/s41060-025-00728-9
[28] Albahadili, A.J.S., Akbas, A., Rahebi, J. (2024). Detection of phishing URLs with deep learning based on GAN-CNN-LSTM network and swarm intelligence algorithms. Signal, Image and Video Processing, 18(6): 4979-4995. https://doi.org/10.1007/s11760-024-03204-2
[29] Nayak, G.S., Muniyal, B., Belavagi, M.C. (2025). Enhancing phishing detection: A machine learning approach with feature selection and deep learning models. IEEE Access., 13: 33308-33320. https://doi.org/10.1109/ACCESS.2025.3543738
[30] Jabbar, H., Al-Janabi, S. (2025). AI-driven phishing detection: enhancing cybersecurity with reinforcement learning. Journal of Cybersecurity and Privacy, 5(2): 26. https://doi.org/10.3390/jcp5020026
[31] Mohammad, R., McCluskey, L. (2012). Phishing websites. UCI Machine Learning Repository. https://doi.org/10.24432/C51W2X
[32] Vrbančič, G. (2020). Phishing websites dataset. Mendeley Data. https://doi.org/10.17632/72ptz43s9v.1
[33] Tijjani, A. (2024). Phishing email and SMS dataset with NLP categories. Kaggle Dataset. https://www.kaggle.com/datasets/ahmadtijjani/phishing-urgency-authority-persuasion.