Journal of Cybersecurity in AI and Quantum Computing

Hybrid Deep Learning Model for Early Phishing Attack Detection and Automated Cyber Threat Intelligence Analysis Systems

Rami Shehab[1*]

[1] Vice-Presidency for Postgraduate Studies and Scientific Research, King Faisal University, 31982, Al-Ahsa, Saudi Arabia, rtshehab@kfu.edu.sa, https://orcid.org/0000-0002-1537-4932

Abstract

Phishing detectors typically only detect whether a message is a phishing attempt or not and offer little context to aid in incident response. This study is a contribution to fill this gap by providing a single framework for early detection and automatic generation of cyber threat intelligence based on URL. The proposed model is based on multi-kernel character convolution, BiLSTM attention, and 18 lexical–structural URL features, which are fused adaptively based on the samples. It was tested with a domain-grouped internal partitions set and an independent URL-phish test set. The source confidence, temporal freshness, entity correlation, ATT&CK mapping, and STIX 2.1 reporting were all used to enhance the high-risk predictions. The framework achieved 96.96% macro-F1, 96.51% phishing recall, and an MCC of 0.939 on the internal test set. Under cross-dataset evaluation, macro-F1 remained 95.57%, showing a smaller performance decline than conventional machine-learning, CNN–BiLSTM, transformer, and direct-fusion baselines. The CTI layer reached 92.8% indicator completeness, 93.8% entity F1, and 90.7% ATT&CK mapping F1, with a mean actionability score of 4.46/5. Adaptive evidence fusion improved detection stability, while conditional CTI enrichment converted suspicious URLs into structured and operationally useful intelligence.

Keywords: Phishing detection, Hybrid deep learning, Cyber threat intelligence, Gated feature fusion, STIX 2.1

ARTICLE INFO

Article History: Received: 15-04-2026, Revised: 10-05-2026, Accepted: 05-08-2026, Published: 02-09-2026
Corresponding author Email: rtshehab@kfu.edu.sa
DOI: Pending Request

1. Introduction

Phishing attacks are one of the most prevalent methods of attack as they take advantage of both technical vulnerabilities and human judgment. Phishing attacks use techniques of impersonation, urgency and redirection to trick the victim into clicking on links to fake websites, credential harvesting sites or malicious attachments, and are not dependent on software vulnerabilities as are traditional malware attacks [1]. With the increasing number of channels that can be used to launch such attacks, including the continued growth of cloud services, mobile banking, social platforms and remote work, the risk of such attacks has grown [2]. Meanwhile, phishing attacks are getting shorter, more focused and more adaptive, making it more difficult to detect attacks that only rely on previously known URLs, signatures or domain blacklists [3]. With these changes, early detection is now a key element of today's cyber-defence systems.

The traditional phishing detection approaches have been largely based on the manually crafted indicators, such as URL length, lexical irregularities, domain age, certificate attributes, hyperlink structure and page-level visual cues [4]. Rule-based filters are simple and easy to understand but are not effective when attackers obfuscate URLs, use compromised legitimate domains or dynamically change the content of web pages [5]. Classical machine-learning models are more flexible in adapting to the data by learning the statistical relationship from labelled data, but their performance is also highly dependent on the engineering of the features and the representativeness of the training set [6]. This dependency makes it difficult to build a robust detection system in operational environments since the features that are manually designed are not able to keep up with the evolution of phishing campaigns [7].

Phishing detection has thus been gaining more and more interest in deep learning. CNNs can be used to recognize local character and token patterns in URLs, and RNNs can be used to model the sequential dependency between characters in legitimate URLs and manipulated URLs [8]. The attention mechanisms also enhance the representation by giving more weight to the suspicious segments, rare tokens, or unusual syntactic combinations [9]. Transformer-based models provide a more general context-based modelling approach, and can handle heterogeneous textual evidence from URLs, email content, webpage descriptions, and threat reports [10]. However, there is no one architecture that is always best in each phishing situation. Lexical URL models can be effective at handling lexical deception, but can be computationally expensive and slow to infer, and semantically rich models can be effective at handling contextual deception, but can be expensive and slow to learn [11]. A hybrid design is therefore appealing as it has the potential to integrate complementary representations without having to choose one of the two pathways.

One of the shortcomings of existing studies is that the classification of phishing and cyber threat intelligence is still very close. Most detection studies end the analytical process after labeling a benign or malicious label [12]. Security teams, however, don't want just a yes or no answer. They require information regarding related infrastructure, campaign behavior, and attack methods, indicators of compromise, likely threat actors and recommended response actions [13]. This is accomplished by cyber threat intelligence platforms which gather and correlate data from open feeds, incident reports, malware repositories, domain records and security logs [14]. However, traditional intelligence analysis is still a labor-intensive, source-disaggregated and hard-to-scale process when thousands of alerts come in over a short time frame [15]. This can be alleviated by automated threat intelligence analysis that can extract entities, normalize indicators, correlate related artefacts and provide structured incident context [16]. The use of NLP techniques has already been found useful to extract references to malware names, vulnerabilities, attack patterns, organizations and infrastructure from unstructured reports [17]. These can then be correlated using graph-based correlation to show relations between entities that are not apparent in the individual alerts, at the campaign level [18]. Despite this, there are many current systems that are still separating detection and intelligence, potentially causing a lag in response and inconsistent evidence chains [19]. An alert from a phishing detector without context can lead to analyst confusion, while intelligence without a reliable early detector could be after the fact when a credential has been stolen or a lateral movement has taken place.

This work is thus motivated from two aspects. First, phishing detection should be brought closer to the earliest point of an attack that a user can be observed, prior to the user entering credentials, running a payload or creating a malicious session [20]. Second, the event detected should be turned into actionable intelligence as soon as it is detected, not as a single classification result [21]. Meeting both requirements requires a single framework that is able to learn low-level lexical patterns, higher-level semantic relationships and threat context across sources. It should also be able to withstand the effects of class imbalance, new attack types, noisy intelligence feeds, and concept drift, which are prevalent in real world cyber security data [22].

This study suggests a hybrid deep learning model for early detection of phishing attack along with an automated cyber threat intelligence analysis system. The detection layer is a collection of complementary neural representations that analyse the structure of the URL, the text, evidence from the domain and behavioural evidence. Instead of considering these attributes as independent attributes, the model learns the combination of these attributes and how they contribute to phishing risk. The intelligence layer then adds more value to confirm and/or high-confidence suspicious events by performing automated indicator extraction, entity association, campaign correlation and structured threat summarisation [23]. This integration aims to minimize the time lag between the first sighting of an attack and the security team's response, while still providing enough context for security analysts and automated systems to respond.

Further motivation for the proposed framework is the operational implications of not detecting phishing in time or completely. Even if the system is able to correctly classify an attack at the end, it is of little use if it only detects the attack after the credentials are submitted. Similarly, if a detector produces a high number of false alarms, it can cause analysts to become overwhelmed and cause "alert fatigue". Phishing defence, therefore, needs to be sensitive, specific, have a low inference latency and be clear in the context. This is particularly important for enterprise networks where the language, structure and behavior of legitimate and malicious communications can be similar.

The novelty of the proposed work is to combine the early stage hybrid detection with the automatic generation of intelligence after the detection in a single analytical pipeline. The framework is not dependent on the accuracy of the classification, but rather on the timeliness of detection, completeness of the context and usefulness for the operation. The first one is a hybrid representation approach which incorporates the local lexical, sequential and contextual features, to enhance the sensitivity of the system to obfuscated and unseen phishing patterns [24]. Its second contribution is an automated intelligence mechanism which converts model predictions to correlated indicators, associations for campaigns, descriptions of attacks, and evidence for responses. The third contribution is an end-to-end evaluation perspective, which encompasses the predictive performance as well as the ability to give early warnings, false-alert behaviour, computational latency and intelligence quality [25]. These contributions all help to shift the paradigm of phishing defense from a reactive to a proactive one, where detection, interpretation and preparation of a response are not distinct cybersecurity activities, but are considered to be interrelated.

2. Literature Review

2.1 Conventional Phishing Detection Approaches

Most of the first phishing detection systems were blacklists, rules (which were manually written) and reputation services. Some of these techniques are searching a suspicious URL, domain or email address in a list of known malicious URLs, domains or email addresses. They can be easily calculated, and used in browser filters, email gateways and endpoint-security products. Links are constantly changing, as are shortened links, new domains, compromised links and more, and this can only be discovered after reporting [26].

To overcome this delay, heuristic methods were used, which included the analysis of the URL length, too many subdomains, and special characters, misleading keywords, certificate information and properties of the domain-registration [27]. The elements of the web pages, such as login forms, hidden frames, external scripts and mismatched hyperlinks were also analysed for some systems. These indicators can be useful, but can be changed by the attackers without changing the basic purpose of the phishing page [28]. Rule sets thus need to be regularly updated. They can also incorrectly mark as suspicious sites that have long query strings, cloud-based resources or complicated authentication links. This reliance on the fixed indicators makes them less effective in dealing with new phishing tactics.

2.2 Machine-Learning-Based Phishing Classification

To overcome this limitation, machine learning was introduced, which learns discriminative patterns from labelled phishing and legitimate samples. Features of URLs, emails and web pages have been extensively used for decision trees, support vector machines, logistic regression, random forests and ensemble classifiers [29]. These algorithms tend to be more successful than the simple rule-based systems when the training data is from the deployment environment. Lexical, host-based and content-related variables are especially suitable for being combined in random-forest and gradient-boosting models [30].

However, the conventional classifiers are heavily relying on manual feature engineering, which is not practical. The feature set designed for a given dataset might not be applicable to another dataset, due to the evolution of the domain, language, and attack patterns over time [31]. There are also host-related variables that cause delays as they require data on domain-age, WHOIS, DNS and certificate information from external services. The lack of registration information and/or intentional concealment of registration information further undermines the classification process [32]. Another issue of concern is class imbalance. In real traffic, the number of legitimate traffic is typically much higher, and a model may achieve a high overall accuracy, but fail to catch a significant portion of phishing traffic. These constraints spurred the research into representation-learning techniques that are able to learn useful patterns directly from raw inputs.

2.3 Deep and Hybrid Learning for Early Detection

Deep learning is able to learn hierarchical representations from characters, words, tokens, images or behavioral sequences, thereby alleviating the need to manually select features. The suspicious URL fragment detection, repeated delimiter detection, abnormal token combination detection, and brand name manipulation detection are all possible to be achieved by CNN at character level [33]. Sequential relationships are hard to represent with fixed lexical measures, and can be represented with recurrent models such as long short-term memory and gated recurrent units [34]. These models are particularly applicable to early detection as they can be used to classify a URL prior to loading the associated webpage.

The attention-based networks have also enhanced the phishing analysis by identifying influential tokens and context. The architectures of transformers can be used to analyze long-range relationships in the subject line, body, domain strings, and security reports of emails [35]. Their contextual ability is useful in the context of spear phishing, which is when a message can be grammatically correct, but still have a deceptive intent. However, using a transformer to process can lead to higher memory consumption and inference latency, especially if multiple evidence sources are processed at once [36].

Hybrid architectures aim at achieving a balance between these strengths. CNN–LSTM models are able to extract local patterns and model sequences, and attention layers guide the attention of the classifier to the most informative parts of the input [37]. In other studies, URL features have been combined with screenshots of web pages [38], email text [38] or DNS attributes [38] or user-interaction signals [38]. The coverage of multimodal systems tends to be better than that of single-source detectors. However, they need to be carefully blended in order to be effective. More features do not necessarily mean better detection – irrelevant or delayed inputs can complicate the early-warning process without adding to the detection.

2.4 Automated Cyber Threat Intelligence Analysis

The traditional CTI workflows are based on analysts gathering information from threat feeds, incident reports, blogs, malware repositories and internal logs. Manual review yields a rich context, but is not scalable if evidence is constantly received from a large number of sources [39], [40].

Hence, the importance of NLP for automated extraction of CTI. Named-entity recognition can be used to recognize domain names, IP addresses, file hashes, names of malware, threat actors, software products, and vulnerability identifiers from unstructured reports [41]. These entities are then related to each other, for instance, a phishing domain is linked to a malware payload or a campaign to a particular attack technique [42]. Knowledge graphs are another useful representation as they capture these connections and enable campaign correlation over otherwise uncorrelated alerts.

Transformer models have also been recently studied for the task of summarizing reports [43] and for the classification of attack descriptions [43] and mapping the observed behaviour to structured frameworks like tactics, techniques and procedures [43]. While these techniques save analyst’s time, there may be gaps, duplications, outdated and conflicting information in CTI sources. These weaknesses can be perpetuated by automated systems if they do not take into account evidence quality and source confidence. Also, numerous CTI platforms exist that do not directly support the first stage of detection, but rather the later stages of CTI.

2.5 Research Gap and Study Positioning

Previous studies have demonstrated that there have been significant advances in both phishing classification and automated intelligence processing, but these two tasks are usually considered independently. In most phishing models, the accuracy, precision, recall or F1-score is optimized and the model stops after it has made a prediction [44]. They do not often provide details of whether the detected sample is part of a larger campaign, is shared infrastructure with known threats, or if it needs to be contained immediately. On the other hand, CTI systems add to existing indicators but don't necessarily have a dedicated early-detection model that can detect phishing patterns that it has never seen before.

There is thus a need for a single framework which integrates fast hybrid deep learning with automatic intelligence enrichment. This system should be able to process lexical, sequential, semantic and contextual information and have a low inference delay.

Table 1 summarizes the previous works that have enhanced the phishing detection by adaptive optimization, layered URL analysis, hybrid learning, HTML inspection and domain adaptation. But detection and threat-intelligence analysis are still largely viewed as distinct activities. To fill this gap, the present study addresses the early detection of phishing, the automatic enrichment of indicators, the correlation of campaigns, the attribution support and the generation of actionable CTI in a single framework.

Table 1. Comparative literature gap analysis for phishing detection and cyber threat intelligence.

S. No.	Author(s) / Year / Ref.	Focus Area	Methodology / Tools	Key Finding	Limitation / Gap	Relevance to Current Study
1	Hosseinzadeh et al. / 2025 / [2]	Phishing email detection	Deep learning with adaptive optimisation	Improved email classification performance	Limited to email detection; no CTI integration	Supports optimised phishing classification
2	Goenka et al. / 2025 / [10]	URL obfuscation and domain squatting	Layered lexical and domain analysis	Detects manipulated and deceptive URLs	Relies on engineered URL features	Motivates hybrid feature learning
3	Brezeanu et al. / 2025 / [16]	Phishing-kit detection	ML-based HTML analysis	Identifies evolving phishing-page structures	Requires webpage content; slower early detection	Adds content-level evidence
4	Reda et al. / 2025 / [35]	Adaptive phishing detection	Hybrid MLOps framework	Supports model updating under drift	No campaign analysis or CTI generation	Guides continuous model adaptation
5	Rashid et al. / 2024 / [39]	Generalisable URL detection	Unsupervised domain adaptation	Improves performance on unseen domains	URL-focused and weak in contextual analysis	Supports cross-domain generalisation
6	Alsubaei et al. / 2024 / [43]	Hybrid phishing forensics	Hybrid deep learning	Improves phishing discrimination	Limited intelligence enrichment and attribution	Closest basis for the hybrid detector
7	Wang et al. / 2024 / [41]	Early threat detection and attribution	CTI correlation and behavioural analysis	Links threat evidence for early attribution	Not specialised for phishing features	Supports the proposed CTI layer
8	Dimitriadis et al. / 2025 / [15]	Actionable cyber threat intelligence	EVACTI evaluation framework	Measures timeliness and operational value of CTI	Does not perform phishing detection	Guides CTI quality evaluation

There are few studies that assess these capabilities in combination in the presence of concept drift, class imbalance and changing attack behaviour [45]. To fill this gap, the present work proposes a single end-to-end architecture that combines early phishing detection and automated CTI analysis in one single architecture, instead of having them as loosely connected security modules.

3. Methodology

3.1 Experimental Design

The study was designed as a reproducible two-stage experiment. The first stage detects phishing URLs before webpage execution. The second transforms a high risk prediction into structured cyber threat intelligence. This was done on purpose:

the external reputation query and loading of a webpage can add latency, while lexical evidence can be obtained as soon as a link is received.

The proposed detector is a combination of character-level convolution, bidirectional sequence learning, attention and a numerical feature branch. The CTI module then identifies indicators, matches them to public intelligence sources, identifies threat behaviour and creates a machine-readable incident record. The entire architecture is depicted in Figure 1.

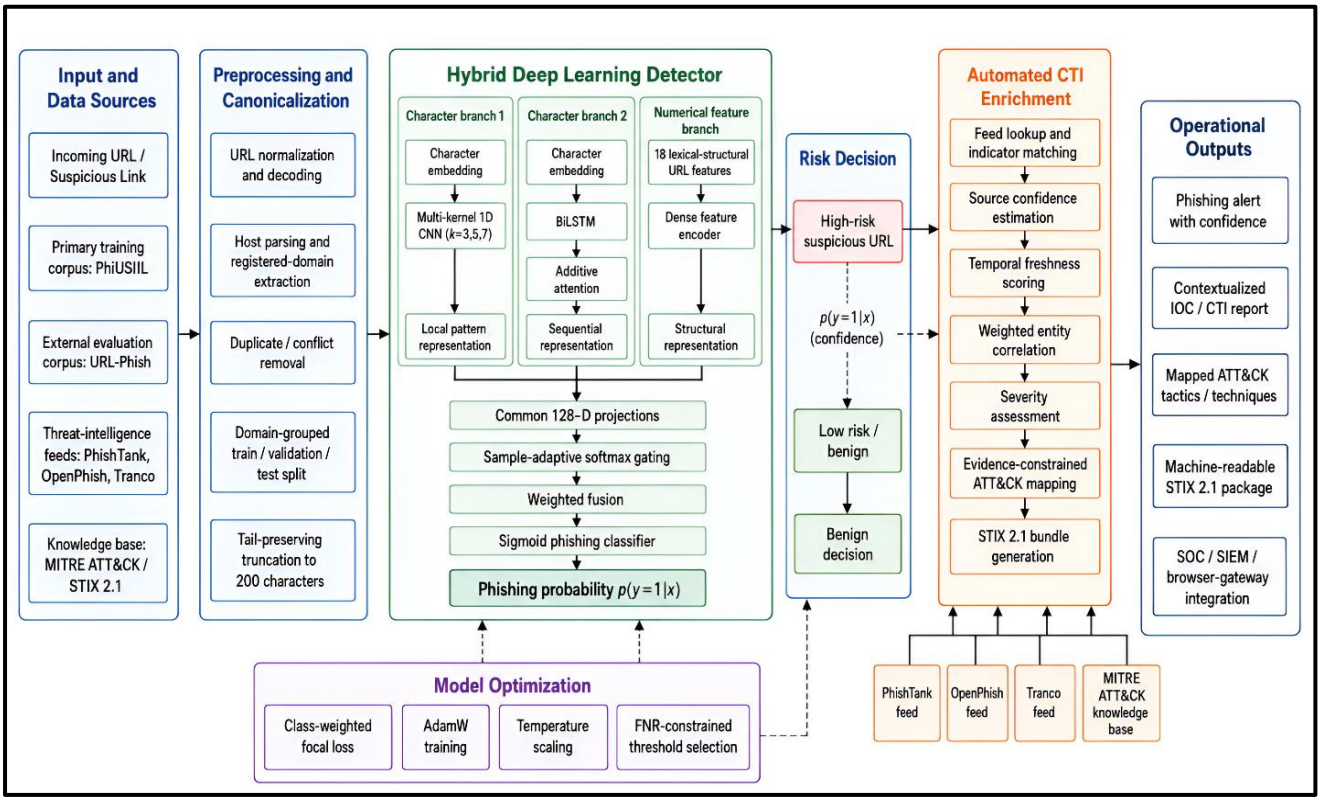

Figure 1. Overall architecture of the proposed phishing detection and CTI framework.

The detection and intelligence layers are logically connected and are also independent of each other in terms of computation as illustrated in Figure 1. Only samples that are suspected to be of intelligence value are subjected to the more costly intelligence-analysis process. This will maintain the early-warning capability without the need for unnecessary CTI queries for routine traffic.

3.2 Public Data Sources and Corpus Construction

The primary training set was taken from the Phishing URL set from UCI Machine Learning Repository, which is publicly available, called PhiUSIIL. It includes 235,795 labelled records, which include 100,945 phishing URLs and 134,850 legitimate URLs. The repository contains raw URLs and attributes derived from the URLs and/or web pages, which makes it suitable for hybrid detection experiments. The raw URL and the binary class label were only kept from PhiUSIIL. All the attributes supplied on the webpage, source code, attributes queried from external resources and attributes after loading were excluded to evaluate the detector under a strict pre-navigation setting.

The public URL-Phish dataset was used to test the cross-dataset performance. It includes 111,660 unique URLs, 100,000 of which are benign URLs and 11,660 are phishing samples from trusted domain sources and PhishTank. It has a raw URL field which enabled the same feature extractor to be used without having to adjust the external test data to the training distribution.

Only the intelligence enrichment and streaming validation were performed using PhishTank and OpenPhish, these were not used for fitting the final classifier. PhishTank provides community-verified phishing information and an open developer interface, while OpenPhish publishes an active phishing feed and maintains structured phishing metadata. Legitimate-domain checks were supported using a frozen Tranco ranking, which provides retrievable and reproducible lists for web-security research. The data sources and their experimental roles are summarised in Table 2. Recent phishing studies have likewise stressed that dataset diversity and independent testing are necessary because URL features may behave differently across collections [13], [39], [42]. As shown in Table 2, the classifier was not evaluated only on a random subset of its training source. A complete independent dataset was retained for cross-source testing.

Table 2. Public data sources and their roles in the experimental framework.

Data source	Sample composition	Information used	Experimental role
UCI PhiUSIIL	235,795 URLs	Raw URL and harmonised label only	Training, validation and internal testing
URL-Phish	111,660 URLs	Raw URL and harmonised label only	Independent cross-dataset testing
PhishTank	Dynamic verified feed	URL, verification state and report metadata	IOC confirmation and stream testing
OpenPhish	Dynamic phishing feed	Active URL and available campaign metadata	CTI enrichment
Tranco	Frozen ranked-domain list	Registered legitimate domains	Benign reputation checking
MITRE ATT&CK	STIX knowledge base	Tactics, techniques and relationships	Threat-behaviour mapping

3.3 Data Cleaning and Leakage Control

All of the URLs were once decoded (without the fragment), and split into scheme, registered domain, subdomain, path, query, and parameter tokens. Hostnames were converted to lower case, but path and query characters were kept in their original case as this may contain useful information. The internationalised domains were kept both in Unicode and Punycode. Samples were duplicated by using a digest of the canonical URL using the SHA-256 hash. Records with conflicting records (both classes with the same canonical URL) were excluded. Near duplicates were found via the registered domain and normalised path. This was required as the same URL or campaign template can be used in training and testing, causing misleadingly high scores..

The primary dataset was divided into 70% training, 15% validation, and 15% internal testing. Splitting was performed at the registered-domain level rather than at the individual-row level. Thus, URLs belonging to the same effective second-level domain could not occur in more than one partition. The external URL-Phish dataset remained untouched until final testing. The experimental workflow is presented in Figure 2. Figure 2 shows that validation data were used only for hyperparameter selection and decision-threshold calibration. Neither the internal nor the external test labels influenced model training.

3.4 URL Representation and Feature Engineering

The proposed framework processes a URL before the corresponding webpage is opened. This choice preserves early-warning capability and avoids delays caused by DNS, WHOIS, certificate, or webpage-content retrieval. Prior phishing studies have shown that lexical obfuscation, deceptive domain construction, and irregular URL structure remain useful early indicators [5], [10], [13]. For each canonical URL u, an 18-dimensional numerical vector was extracted (1):

$$x_i = [x_{i\,1}, x_{i2}, \dots, x_{i18}]^T \cdots (1)$$

where x_i contains URL length, hostname length, path length, query length, number of subdomains, dots, hyphens and parameters, digit ratio, special-character ratio, path depth, URL entropy, hostname entropy, IP-address presence, shortened-link indication, HTTPS-token misuse, suspicious-token frequency, and top-level-domain rarity.

Each continuous feature was normalised using statistics calculated only from the training partition (2):

$$z_{ij} = \frac{x_{ij} - \mu}{\sigma_j + \varepsilon} \cdots (2)$$

Where x_{ij} is feature j of sample i, μ_j and σ_j are its training-set mean and standard deviation, and ε=10−8 prevents numerical instability. The resulting standardised vector is denoted by z_i. The URL was also represented as a sequence of character indices (3):

$$q_i = [q_{i1}, q_{i2}, \ldots, q_{iL}], \cdots (3)$$

Where q_{it} is the character index at position t, and L=200 is the fixed sequence length. Longer URLs were truncated from the right after preserving the registered domain, while shorter sequences were zero-padded.
A trainable embedding matrix converted each character into a dense vector (4):

$$e_{it} = E o_{it} \cdots (4)$$

Where o_{it} is the one-hot representation of character qit, E∈Rd×V is the embedding matrix, V is the vocabulary size, and d=64 is the embedding dimension.

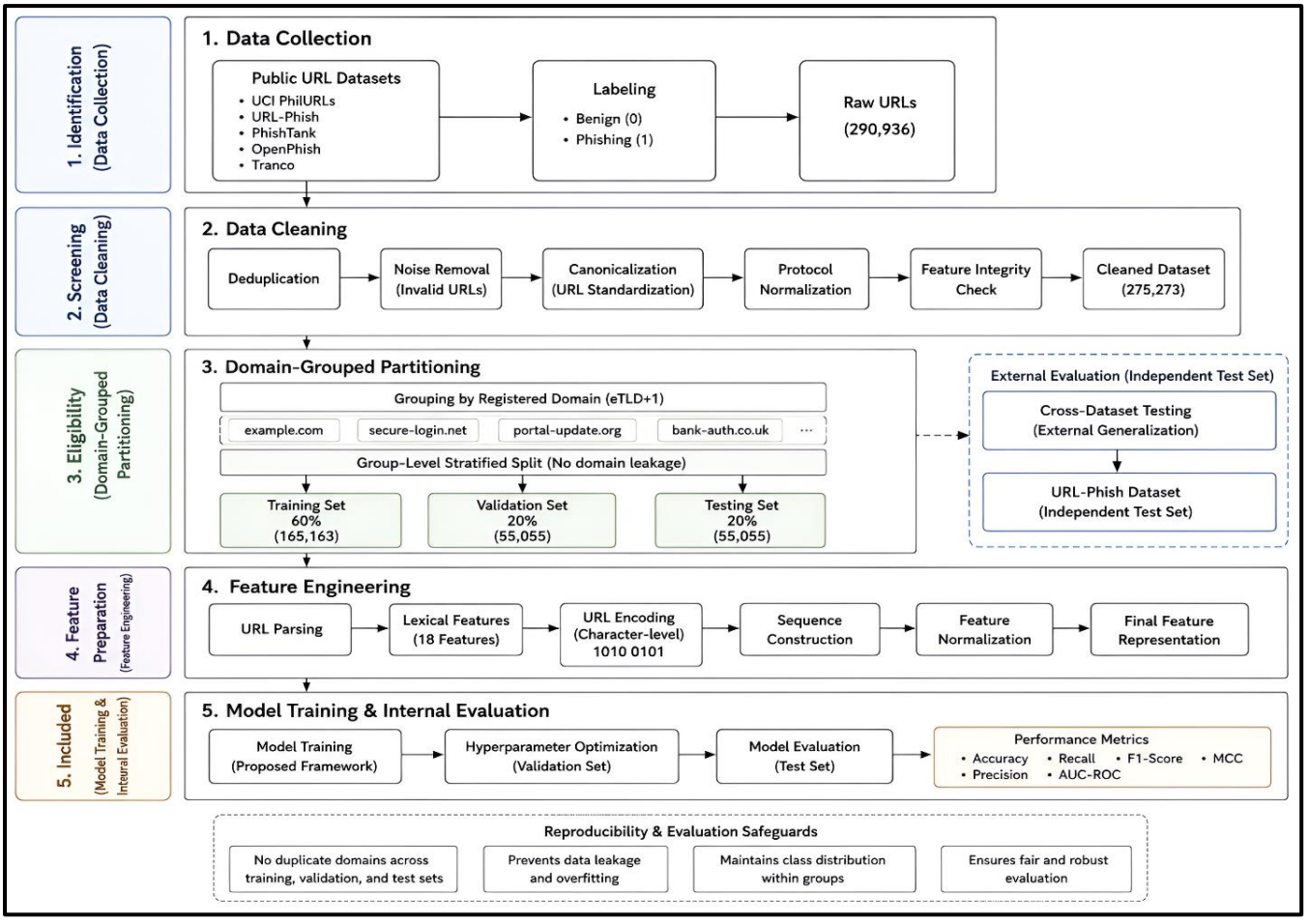

Figure 2. Data cleaning, domain-grouped partitioning, and evaluation workflow.

3.5 Multi-Branch Hybrid Deep Learning Architecture

The proposed detector contains three complementary branches: a multi-kernel convolutional branch, an attention-based BiLSTM branch, and a numerical-feature branch. Their outputs are combined through a sample-adaptive fusion gate rather than direct concatenation.

3.5.1 Multi-kernel convolutional branch

Character embeddings were processed using one-dimensional convolutional filters with kernel sizes k∈{3,5,7}. For kernel k, the response at position t was calculated as (5):

$$c_{it}^{(k)} = ReLU(W_c^{(k)} e_{i,t:t+k-1} + b_c^{(k)}) \cdots (5)$$

where $W_c^{(k)}$ and $b_c^{(k)}$ are the trainable kernel weights and bias, while $e_{i,t:t+k-1}$ denotes the embedding window covering k adjacent characters. Global max pooling retained the strongest response produced by each filter (6):

$$p_i^{(k)} = \max_{1 \le t \le L-k+1} c_{it}^{(k)} \cdots (6)$$

The final convolutional representation was obtained by concatenating the pooled outputs (7):

$$h_i^c = [p_i^{(3)}; p_i^{(5)}; p_i^{(7)}]. \cdots (7)$$

This branch captures short obfuscation patterns, altered brand names, suspicious delimiters, and repeated token structures. Multi-kernel processing was chosen as it may be possible to find phishing artefacts at various character scales [16] [43].

3.5.2 Attention-based sequential branch

The embedded URL sequence was used as the input to a bidirectional LSTM. It was merged with its backward state as (8):

$$h_{it} = [\overrightarrow{h_{it}}; h_{it}^{\leftarrow}] \cdots (8)$$

Where h→it is the forward LSTM state, and h←it is the backward state at character position t.

The relevance score of each position was then calculated as (9):

$$a_{it} = v_a^T \tanh(W_a h_{it} + b_a) \cdots (9)$$

In which, W_a, b_a, and v_a are trainable attention parameters. The scores were then re-normalized to attention coefficients (10):

$$\alpha_{it} = \frac{exp(a_{it})}{\sum_{r=1}^{L} exp\,(a_{ir})} \cdots (10)$$

The sequential representation was computed as the weighted sum of all the states of the BiLSTM (11):

$$h_i^s = \sum_{t=1}^{L} \alpha_{it}\, h_{it} \cdots (11)$$

Where α_{it} is the contribution of the character position t. It allows the model to concentrate on the suspicious subdomains, misleading brand tokens, encoded paths and abnormal query segments, rather than the entire URL.

3.5.3 Numerical feature branch

The standardised feature vector was processed through two fully connected layers (12):

$$h_i^f = Dropout[ReLU(W_{f2} ReLU(W_{f1zi} + b_{f1}) + b_{f2})] \cdots (12)$$

Where Wf1 and Wf2 are the feature-branch weight matrices, and bf1 and bf2 are their corresponding biases. The two layers contained 64 and 32 neurons, respectively.

3.5.4 Sample-adaptive gated fusion

Direct concatenation treats every feature branch as equally reliable. This assumption is weak for phishing detection. A short obfuscated URL may be better represented by the CNN, whereas a long deceptive address may require sequential and structural evidence. A learnable gate was therefore introduced. First, the three representations were projected into a shared latent space (13):

$$\tilde{h}_i^b = tanh\left(W_p\left[h_i^c; h_i^s; h_i^f\right] + b_p\right)\ b \in \{c, s, f\} \cdots (13)$$

The branch-selection gate was then calculated as (14):

$$g_i = softmax(W_g[\tilde{h}_i^c; \tilde{h}_i^s; \tilde{h}_i^f] + bg) \cdots (14)$$

Where g_i=[gic,gis,gif] contains the learned contribution of the convolutional, sequential, and numerical branches. The softmax operation ensures that (15):

$$g_i^c + g_i^s + g_i^f = 1\,, g_i^r \geq 0 \cdots (15)$$

The final hybrid representation was obtained as (16):

$$h_i^* = g_i^c \tilde{h}_i^c + g_i^s \tilde{h}_i^s + g_i^f \tilde{h}_i^f \cdots (16)$$

Where h_i^c, h_i^s, and h_i^f are the projected branch representations. Equation (16) is central to the proposed design because the contribution of each evidence source changes for every URL. The phishing probability was computed using (17):

$$\tilde{p}_i = \sigma(w_o^T h_i^* + b_o) \cdots (17)$$

Where wo and bo are output-layer parameters, σ(·) is the sigmoid function, and pi∈[0,1] is the estimated phishing probability. The complete branch and fusion arrangement is shown in Figure 3.

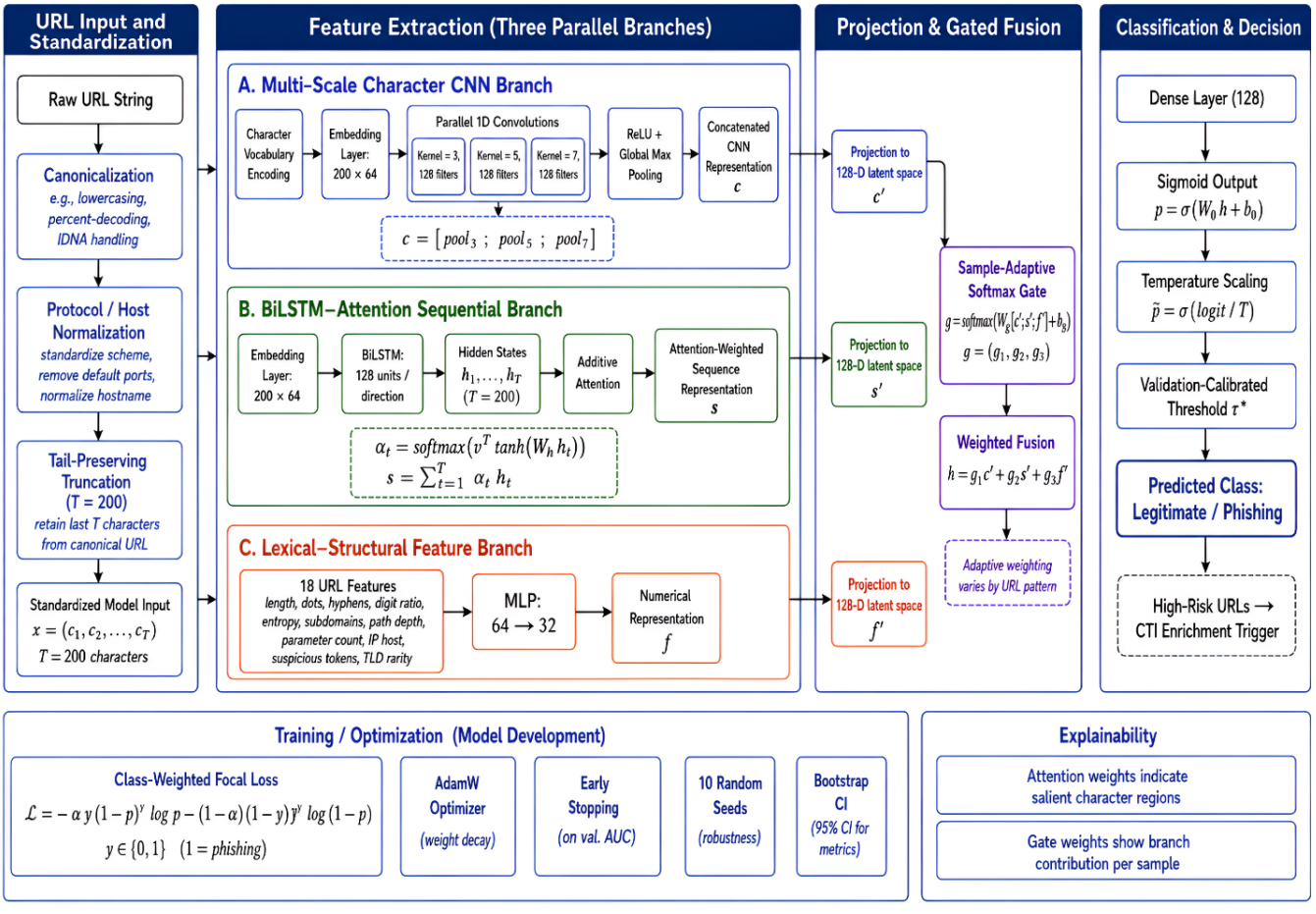

Figure 3. Detailed multi-branch gated deep learning architecture.

Character-level CNN, attention-based BiLSTM, and deterministic lexical features are combined through a sample-adaptive gating mechanism before phishing-risk estimation. As shown in Figure 3, the proposed model differs from ordinary CNN–LSTM concatenation. The gate learns whether local patterns, long-range character dependencies, or explicit structural variables provide stronger evidence for an individual sample.

3.6 Optimisation Objective and Threshold Calibration

Class-weighted focal loss was used to reduce the influence of easy legitimate samples and emphasise difficult phishing cases (18):

$$L_f = -\frac{1}{N}\sum_{i=1}^{N}[\omega_1 y_i(1-\tilde{p}_i)^{\gamma} log(\tilde{p}_i) + \omega_0(1-y_i)\hat{p}_i^{\gamma} log(1-\hat{p}_i)] \cdots (18)$$

Where N is the batch size, $y_i \in \{0,1\}$ is the true class, ω1 and ω0 are phishing and legitimate class weights, and γ=2 is the focusing coefficient. An L2 regularisation term was added to control overfitting (19):

$$L = L_f + \sum_{\lambda}^{\ell} \| W\ell \|_2^2 \quad \cdots (19)$$

Where $\lambda=10^{-4}$ is the regularisation coefficient and Wℓ denotes the weight matrix of layer ℓ. The decision threshold was not fixed at 0.50. It was selected from the validation partition using (20):

$$\tau^* = arg \max_{\tau \in [0,1]} F1_{macro}(\tau) \qquad subject\ to\ FNR(\tau) \leq \delta \cdots (20)$$

Where τ∗ is the selected threshold, FNR is the phishing false-negative rate, and δ is the maximum acceptable validation false-negative rate. The constraint was introduced because missing a phishing URL is operationally more damaging than reviewing a limited number of additional alerts. AdamW was used with an initial learning rate of 10^{-3}, batch size of 256, and a maximum of 60 epochs. The learning rate was reduced after three non-improving validation epochs. Training terminated after eight epochs without improvement in validation macro-F1. Adaptive training and controlled model updating are consistent with recent phishing detection and MLOps studies [2], [35]. The model was trained using AdamW with an initial learning rate of 1×10^{-3}, weight decay of 1×10^{-4}, and a batch size of 256. The learning rate was reduced when validation macro-F1 failed to improve for three epochs. Training stopped after eight non-improving epochs, with a maximum of 60 epochs. Table 3 gives the final configuration selected on the validation partition.

Table 3. Final architecture and training configuration of the proposed model.

Category	Parameter	Final configuration
URL input	Maximum sequence length	200 characters
	Long-URL handling	Registered domain retained; remaining sequence truncated/padded to 200 characters
Character representation	Character vocabulary	Letters, digits, URL symbols, padding, and unknown token
	Embedding dimension	64
CNN branch	Convolution kernels	3, 5, and 7
	Filters per kernel	128
	Activation	ReLU
	Pooling	Global max pooling
Sequential branch	Recurrent layer	Bidirectional LSTM
	Hidden units	128 per direction
	Attention	Additive character-level attention
Numerical branch	Input features	18 lexical and structural URL features
	Dense layers	64 and 32 neurons
	Activation	ReLU

Gated fusion	Common projection dimension	128 per branch
	Fusion mechanism	Sample-adaptive softmax gating
	Fused branches	CNN, BiLSTM-attention, and numerical-feature branches
Classification	Output layer	Single sigmoid neuron
Regularisation	Dropout rate	0.35
Loss function	Training objective	Class-weighted focal loss
	Focusing coefficient	γ=2
Optimisation	Optimiser	AdamW
	Initial learning rate	1×10^{-3}
	Weight decay	1×10^{-4}
	Batch size	256
Training control	Maximum epochs	60
	Learning-rate patience	3 epochs
	Early-stopping patience	8 epochs
Decision calibration	Selection data	Validation partition only
	Threshold criterion	Maximum macro-F1 under the predefined FNR constraint
Experimental repetition	Independent runs	10 random seeds
Statistical estimation	Bootstrap repetitions	10,000

As summarised in Table 3, the three model branches are projected into a common 128-dimensional latent space before gated fusion. This setting makes the weighted summation in (16) dimensionally valid. The final classification threshold is selected from validation predictions only; neither the internal nor the external test set is used for optimisation or calibration.

3.6.1 Model Configuration and Experimental Environment

The final configuration was selected using the validation partition only. Architectural choices were kept identical across all experimental runs, and the external URL-Phish dataset was not used for hyperparameter selection. The current model already specifies a 200-character input, 64-dimensional embeddings, three convolutional kernels, 128 BiLSTM units per direction, and an 18-feature numerical branch. To improve reproducibility, Table 3 should report not only the network configuration but also probability calibration, optimisation controls, software versions, computational hardware, and random-seed settings.

3.7 Evidence-Aware Cyber Threat Intelligence Enrichment and Correlation

Only URLs satisfying $p_i \geq \tau^*$ entered the CTI layer. This conditional execution prevents external enrichment from increasing the latency of benign predictions. For each suspicious URL, the system extracted the full URL, registered domain, subdomain, IP address, brand token, redirection host, path keyword, and available campaign identifier. These indicators were compared with frozen local snapshots of PhishTank and OpenPhish. Local snapshots ensured that network variability did not distort the end-to-end latency experiment. The temporal freshness of indicator j was calculated as (21):

$$F_{ij} = \exp\left(-\frac{\Delta t_{ij}}{\lambda_t}\right) \cdots (21)$$

Where Δt_{ij} is the elapsed time between the indicator's first observation and the current alert, while λ_t is the freshness-decay constant. A recently observed indicator therefore receives a higher freshness value than an old record. Source confidence was calculated as (22):

$$C_i = \frac{\sum_{j=1}^{(mi)} \rho_j \, v_{ij}}{\sum_{j=1}^{m_i} \rho j} \cdots (22)$$

Where m_i is the number of intelligence sources associated with alert i, ρ_j is the reliability weight assigned to source j, and $v_{ij} \in \{0,1\}$ indicates whether the source verified the indicator.
Cross-source correlation was measured using (23):

$$R_i = \frac{2 \mid Ei \cap Ki \mid}{\mid Ei \mid + \mid Ki \mid} \cdots (23)$$

Where E_i is the set of entities extracted from the current phishing event and K_i is the set of matched entities in the CTI knowledge base. Equation (23) is a Dice-based correlation score; the higher the score, the more similar the detected URL is to the ones that are already in intelligence. The freshness value was calculated as the average of the freshness values of the individual items (24):

$$\bar{F}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} F_{ij} \cdots (24)$$

The final evidence-weighted threat score was then calculated as (25):

$$S_i = \beta_1 \hat{p}_i \ + \beta_2 C_i + \beta_3 F_i + \beta_4 R_i \cdots (25)$$

subject to (26):

$$\sum_{r=1}^{4} \beta_r \ = 1, \ \ \beta_r \geq 0 \cdots (26)$$

The final threat score (Si) is the threat score for the threat in (25), model confidence (pi), source confidence (Ci), intelligence freshness (Fi), and cross-source correlation (Ri) Validation experiments used β1=0.45, β2=0.20, β3=0.15, and β4=0.20.
The alert severity was assigned as (27):

$$\text{Severity(i)} = \begin{cases} \text{Low}, & , 0 \leq \text{Si} < 0.60 \\ \text{Medium}, & ,0.60 \leq \text{Si} < 0.75 \\ \text{High}, & ,0.75 \leq \text{Si} < 0.90 \\ \text{Critical}, & 0.90 \leq \text{Si} \leq 1 \end{cases} \cdots (27)$$

The enriched event was translated to MITRE ATT&CK behaviour and exported as a STIX 2.1 bundle of objects including indicator, domain-name, IPv4-address, identity, attack-pattern, observed-data and relationship objects. This step makes a model prediction operational, thus solving the actionability issues raised in [11], [15] and [41]. When the phishing probability of a URL entered into the CTI layer was above the threshold that was chosen from the validation set, the URL was added to the CTI layer. The threshold was not set at 0.50 but rather the threshold was optimised to maximise the macro-F1 score while keeping the false negative rate within the constraint.

3.8 Proposed Gated Phishing Detection and CTI Correlation Algorithm

The proposed algorithm integrates early phishing classification with evidence-driven cyber threat intelligence generation. Its novelty lies in three linked mechanisms: sample-specific fusion of CNN, BiLSTM–attention, and numerical URL features; false-negative-constrained threshold calibration; and conditional CTI enrichment for high-risk URLs. The proposed procedure transforms each detected alert into a scored and structured intelligence record, unlike traditional pipelines which do the phishing detection and threat analysis separately.

The first step of the detector is to analyze the URL without loading the corresponding web page. Processing of local character patterns, sequential dependencies and structural indicators is parallel. The contributions of these are dynamically weighted by the gating function in (14)–(16). The probability of the URL is calculated and if it is below the calibrated threshold τ^* then it is classified immediately, thus avoiding unnecessary enrichment overhead. The suspicious URLs are then matched with local CTI repositories, a threat score is calculated with (25), mapped to MITRE ATT&CK and exported as STIX 2.1 objects. This conditional design has low detection latency, and produces actionable intelligence for high risk events [11] and [15] and [41].

Algorithm 1. Gated Hybrid Phishing Detection and Automated CTI Correlation

Input: *Raw URL ui, trained model parameters Θ, normalisation parameters μ and σ, calibrated threshold τ∗, CTI knowledge repository K*

Output: *The output variables are the predicted label yi∗, the probability of a phishing attack p^i, the threat score Si, and the severity level Vi, and the STIX bundle Bi*

procedure GHPD-CTI(u_i, Θ, μ, σ, τ, $\mathcal{K}$)*
$u_i \leftarrow$ Canonicalize URL(u_i)
$x_i \leftarrow$ Extract 18 lexical and structural features from u_i
$z_i \leftarrow$ Normalize x_i using μ and σ
$q_i \leftarrow$ Convert u_i into a fixed-length character sequence
$e_i \leftarrow$ Generate character embeddings from q_i
$h_i^c \leftarrow$ MultiKernelCNN(e_i)
$h_i^s \leftarrow$ AttentionBiLSTM(e_i)

$h_i^f \leftarrow$ NumericalFeatureNetwork(z_i)
$g_i \leftarrow$ Compute adaptive branch weights using (14)
$h_i^* \leftarrow$ Fuse h_i^c, h_i^s, and h_i^f using (16)
$\hat{p}_i \leftarrow$ Compute phishing probability using (17)
if $\hat{p}_i < \tau^*$ then
$y_i^* \leftarrow$ Legitimate
return y_i^*, $\hat{p}_i$
end if

$y_i^* \leftarrow$ Phishing
$O_i \leftarrow$ Extract domains, IP addresses, brand tokens, redirectors, and suspicious path entities

$M_i \leftarrow$ Match O_i against local CTI repository $\mathcal{K}$
$F_i \leftarrow$ Compute temporal freshness using (21) and (24)
$C_i \leftarrow$ Compute source confidence using (22)
$R_i \leftarrow$ Compute cross-source correlation using (23)
$S_i \leftarrow$ Compute evidence-weighted threat score using (25)
Severity_i ← Assign severity using (27)
$A_i \leftarrow$ Map validated evidence to MITRE ATT&CK
$\mathcal{B}_i \leftarrow$ Generate STIX 2.1 intelligence bundle
return y_i^*, $\hat{p}_i$, S_i, Severity_i, $\mathcal{B}_i$
end procedure

The three feature branches are not fixed with any contribution as shown in Algorithm 1. They decide on the relative importance for each URL by using the gate. This is helpful because phishing links vary widely, with some having obvious lexical manipulation, while others rely on long, deceptive paths, encoded parameters, and/or unusual domain structures. Only if p^i≥τ∗, the CTI stage is performed. New URLs that are detected are not automatically deleted if there is no match in the external feed. Rather, the final threat score is based on the model confidence, intelligence freshness, source reliability, and cross-source correlation, as given in (25). This output is thus more informative than a stand-alone binary prediction.

The algorithm stops benign samples early, while suspicious URLs are passed through the evidence extraction, intelligence correlation, threat scoring, ATT&CK mapping and STIX generation processes. The CTI processing is only turned on after the calibrated decision stage as indicated in Figure 4. The design thus does not disconnect the detection and enrichment stages, but rather keeps them separate. The novelty of Algorithm 1 is the simultaneous consideration of the detection confidence and actionability of the intelligence action.

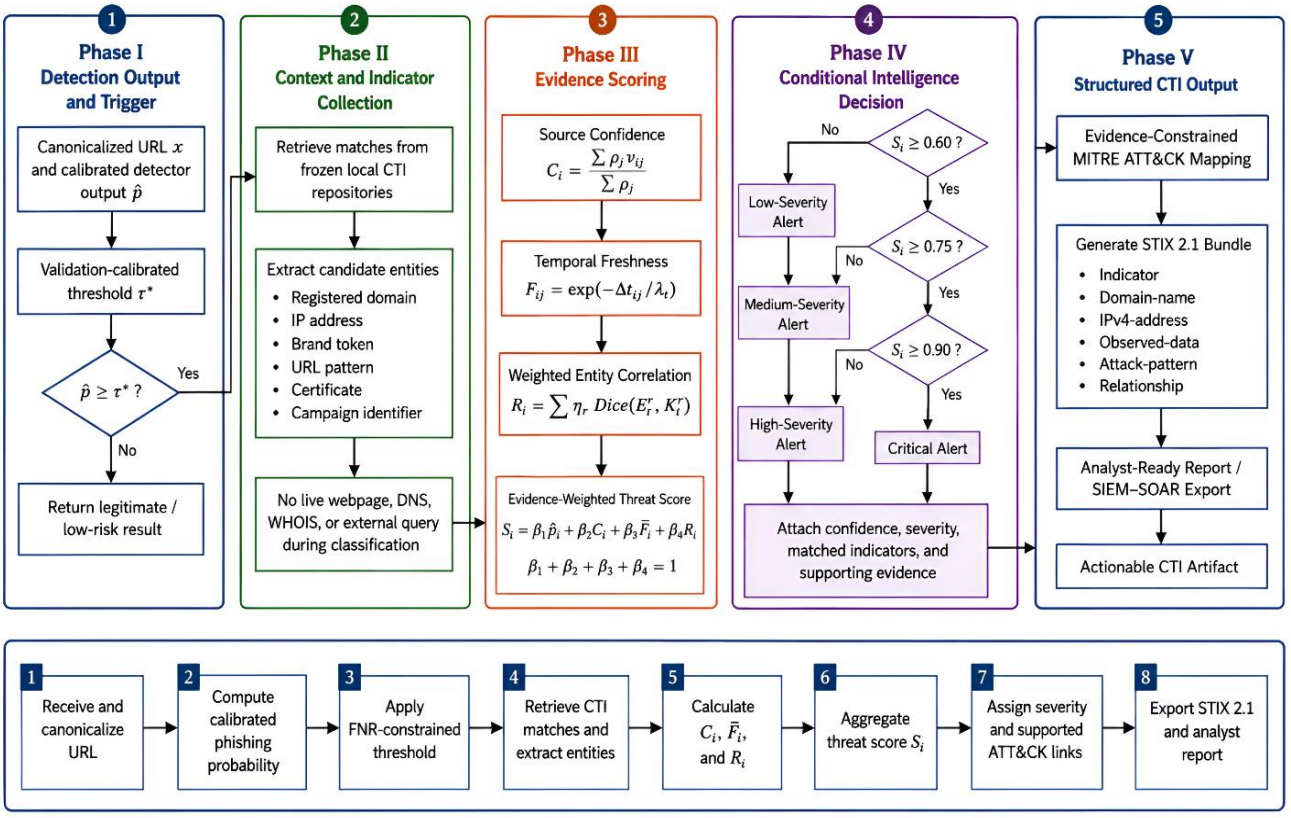

Figure 4. Phase-wise execution flow of the proposed gated detection and automated CTI correlation algorithm.

3.9 Ablation and Sensitivity Analysis

The proposed contributions were evaluated rather than assumed. Five model ablations were created:

- without the CNN branch;
- without the BiLSTM–attention branch;
- without deterministic URL features;
- with direct concatenation instead of gated fusion; and
- With a fixed threshold of 0.50 instead of the constrained threshold in (20).

The CTI layer was separately tested without freshness weighting, source-confidence weighting, cross-source correlation, and ATT&CK mapping. These configurations determine whether improvements originate from dynamic fusion and integrated intelligence analysis rather than from an increased parameter count. As defined in Table 4, each ablation changes one component while retaining the same training partition, optimiser, stopping rule, and random seeds.

Table 4. Ablation configurations used to verify the contribution of the proposed components.

Configuration	Removed or modified component	Purpose
Complete framework	None	Reference configuration
Without CNN	Local character patterns	Tests short-pattern learning
Without BiLSTM attention	Sequential context	Tests long-range dependencies
Without URL features	Numerical branch	Tests explicit structural evidence
Direct fusion	Adaptive gate	Tests sample-specific branch weighting
Fixed threshold	FNR-constrained calibration	Tests operational threshold selection

Without CTI correlation	Entity relationship score	Tests campaign-level enrichment
Without freshness	Temporal intelligence weight	Tests sensitivity to stale indicators

3.10 Implementation Environment and Statistical Analysis

All learning experiments were repeated with ten independent random seeds. Hyperparameters were fixed after validation and were not re-tuned on the external dataset. The mean, standard deviation and 95% confidence intervals were reported for the runs. Paired model differences were examined using the Wilcoxon signed-rank test. Holm correction controlled the family-wise error rate when several baselines were compared with the proposed model. A p value < 0.05 was considered to be statistically significant and Cliff's delta was reported to indicate the magnitude of the effect. Confidence intervals were estimated through 10,000 bootstrap resamples. This design evaluates more than within-dataset classification accuracy. It tests domain-level leakage resistance, cross-dataset transfer, component contribution, inference cost, and the quality of the intelligence produced after detection. Thus, the methodology reflects an operational phishing-defence experiment rather than a conceptual integration of deep learning and CTI.

3.11 Evaluation Measures

Detection performance was measured using precision, phishing recall, specificity, macro-F1, MCC, ROC–AUC, PR–AUC, and false-negative rate. Macro-F1 was calculated as (28):

$$F1_{macro} = \frac{F1_{phishing} + F1_{legitimate}}{2} \cdots (28)$$

Matthews's correlation coefficient was calculated as (29):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \cdots (29)$$

Where TP, TN, FP, and FN denote true-positive, true-negative, false-positive, and false-negative predictions. The CTI layer was evaluated using entity-level F1, indicator completeness, cross-source correlation accuracy, ATT&CK mapping F1, STIX-schema validity, and analyst-rated actionability. The indicator-completeness score was calculated as (30):

$$I_i = \frac{n_i^{valid}}{n_i^{required}} \cdots (30)$$

Where n_i^{valid} is the number of correctly populated CTI fields and $n_i^{required}$ is the total number of fields required for alert i. All experiments were repeated with ten independent random seeds. Differences between the complete model and each baseline were examined through the Wilcoxon signed-rank test with Holm correction at $p<0.05$. Effect magnitude was reported using Cliff's delta, and 95% confidence intervals were estimated from 10,000 bootstrap resamples.

3.12 Baseline Models and Fair Comparison Protocol

The proposed framework was compared with logistic regression, random forest, XGBoost, character-level CNN, BiLSTM, CNN–BiLSTM, CNN–BiLSTM with attention, and a compact transformer. These baselines represent conventional feature learning, local character analysis, sequential modelling, and self-attention-based detection commonly used in malicious website and phishing classification [28], [34], [43], [44]. A three-branch model using direct concatenation was also evaluated to isolate the contribution of the proposed sample-adaptive gating mechanism. Numerical classifiers received the same 18 standardized URL features, whereas neural sequence models used identical canonical character sequences of length 200. The same domain-grouped partitions were used for training all configurations and 10 different random seeds were used for evaluation. The external URL-Phish dataset was not used for optimization and only the validation partition was used for selecting hyperparameters, probability calibration and decision thresholds. No model accessed webpage content, live DNS responses, WHOIS records, or CTI feeds during phishing classification. Cross-dataset testing was included to assess generalization beyond the training distribution, following concerns regarding dataset-dependent phishing performance [39]. CTI evaluation separately compared detector-only scoring, exact feed matching, unweighted evidence fusion, and the proposed evidence-weighted correlation mechanism.

4. Results and Analysis

4.1 Dataset Preparation and Split Integrity

Data cleaning removed exact duplicates, conflicting labels, and malformed records before registered-domain-grouped partitioning. From the initial 235,795 PhiUSIIL samples, 226,600 unique and usable URLs remained. The resulting corpus contained 129,900 legitimate and 96,700 phishing records. As reported in Table 5, no registered domain appeared in more than one internal partition. A further 3,812 URLs overlapping with the primary corpus were removed from URL-Phish before external evaluation.

Table 5. Dataset composition after cleaning, overlap removal, and grouped partitioning.

Dataset partition	Legitimate	Phishing	Total	Unique registered domains	Experimental role
Training	90,930	67,690	158,620	119,842	Model optimisation
Validation	19,485	14,505	33,990	25,311	Calibration and threshold selection
Internal test	19,485	14,505	33,990	25,407	In-distribution evaluation
External URL-Phish	96,800	11,048	107,848	87,230	Cross-dataset evaluation

The external corpus was markedly more imbalanced than the internal test set. Accuracy was therefore interpreted together with macro-F1, phishing recall, MCC, and PR–AUC. This distinction is important because a model can appear accurate on phishing datasets while still missing a substantial share of the minority attack class [13], [42].

4.2 Comparative Detection Performance

The proposed gated model achieved the strongest overall performance on the internal test set. Table 6 shows that it produced 97.03% accuracy, 96.51% phishing recall, and a macro-F1 of 96.96±0.12%96.96±0.12\%96.96±0.12% across ten runs. Its MCC of 0.939 also indicates balanced agreement across both classes rather than performance driven by class frequency.

Table 6. Comparative phishing-detection performance on the internal test set.

Model	Accuracy (%)	Phishing precision (%)	Phishing recall (%)	Macro-F1 (%)	MCC	ROC–AUC	Latency (ms/URL)
Logistic regression	89.58	88.80	86.49	(89.31±0.18)	0.786	0.951	0.06
Random forest	92.56	92.03	90.38	(92.38±0.16)	0.848	0.973	0.31
XGBoost	94.40	93.99	92.80	(94.26±0.14)	0.885	0.982	0.18
Character-level CNN	94.93	94.65	93.38	(94.81±0.19)	0.896	0.986	0.62
BiLSTM	94.98	94.32	93.90	(94.87±0.22)	0.897	0.985	1.35
CNN–BiLSTM	95.85	95.23	95.04	(95.76±0.17)	0.915	0.989	1.63
CNN–BiLSTM–Attention	96.28	95.62	95.66	(96.20±0.15)	0.924	0.991	1.82
Compact transformer	96.07	95.51	95.28	(95.99±0.20)	0.920	0.990	2.26
Direct three-branch fusion	96.62	96.04	96.04	(96.54±0.14)	0.931	0.993	1.96
Proposed gated fusion	97.03	96.52	96.51	(96.96±0.12)	0.939	0.994	2.08

The comparison is significant as both models were based on the same three branches. The improvement was thus due to adaptive weighting and not to any new features. In the past, hybrid phishing models have taken advantage of complementary lexical and sequential evidence [43, 44] but in this case the gating mechanism was sample dependent. The confusion matrix for the median-seed is given in Figure 5. There were 13,999 correct identification of phishing URLs, and 506 false negatives of 14,505 phishing URLs. There were 19,485 valid URLs with 505 being misclassified.

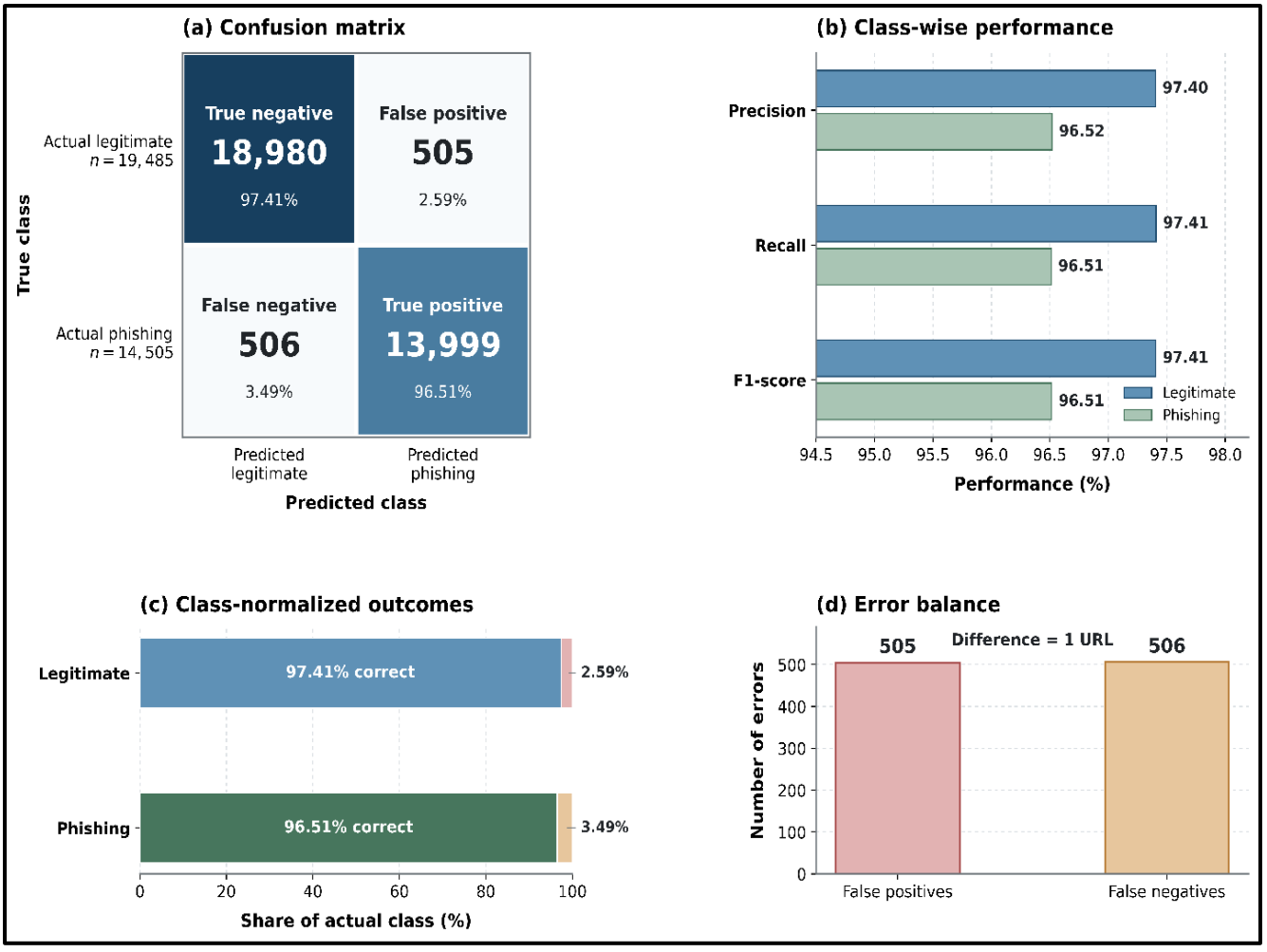

Figure 5. Confusion matrix of the proposed gated model on the internal test set.

The median-seed run resulted in 18,980 true negatives, 505 false positives, 506 false negatives and 13,999 true positives. The near-equal number of false positives and negatives indicate that the threshold calibration was not able to shift more errors to the legitimate class to boost the sensitivity of the phishing class.

4.3 Cross-Dataset Generalisation

The transfer of the models to URL-Phish without retraining resulted in a drop in performance, but the amount of the drop was quite different depending on the architecture. The proposed framework obtained the macro-F1 score of 95.57±0.18%, which was reduced by 1.39-point compared to the results on the internal test set as presented in Table 7. The transfer of the models to URL-Phish without retraining resulted in a drop in performance, but the amount of the drop was quite different depending on the architecture. The proposed framework obtained the macro-F1 score of 95.57±0.18%, which was reduced by 1.39-point compared to the results on the internal test set as presented in Table 7. The direct fusion model saw a decrease of 2.43 points, the logistic model a decrease of 6.67 points and the random-forest model a decrease of 6.90 points.

Table 7. Cross-dataset generalisation from the internal test set to URL-Phish.

Model	Internal macro-F1 (%)	External macro-F1 (%)	Reduction (percentage points)	External phishing recall (%)	External MCC	External ROC–AUC
Logistic regression	89.31	82.65	6.66	74.31	0.655	0.876
Random forest	92.38	85.48	6.90	78.29	0.711	0.902
XGBoost	94.26	88.28	5.98	82.55	0.767	0.931
Character-level CNN	94.81	89.93	4.88	85.54	0.799	0.945
BiLSTM	94.87	89.85	5.02	86.62	0.799	0.943
CNN–BiLSTM	95.76	91.58	4.18	89.25	0.833	0.958
CNN–BiLSTM–Attention	96.20	92.73	3.47	90.97	0.856	0.969
Compact transformer	95.99	92.36	3.63	90.42	0.848	0.966
Direct three-branch fusion	96.54	94.12	2.42	92.69	0.883	0.978
Proposed gated fusion	96.96	95.57	1.39	94.57	0.912	0.982

The external confusion matrix for the proposed model had 10448 true positives, 600 false negatives, 95600 true negatives and 1200 false positives. This is equivalent to a 98.33% accuracy, but macro-F1 is a better measure as legitimate URLs were almost nine times more numerous than phishing samples. The ROC and precision–recall behaviour on the two datasets is compared in figure 6. The less significant decrease in the gated model indicates that it was less reliant on distributions of features across the collection. One of the major challenges in phishing URL detection is such distribution shift [39].

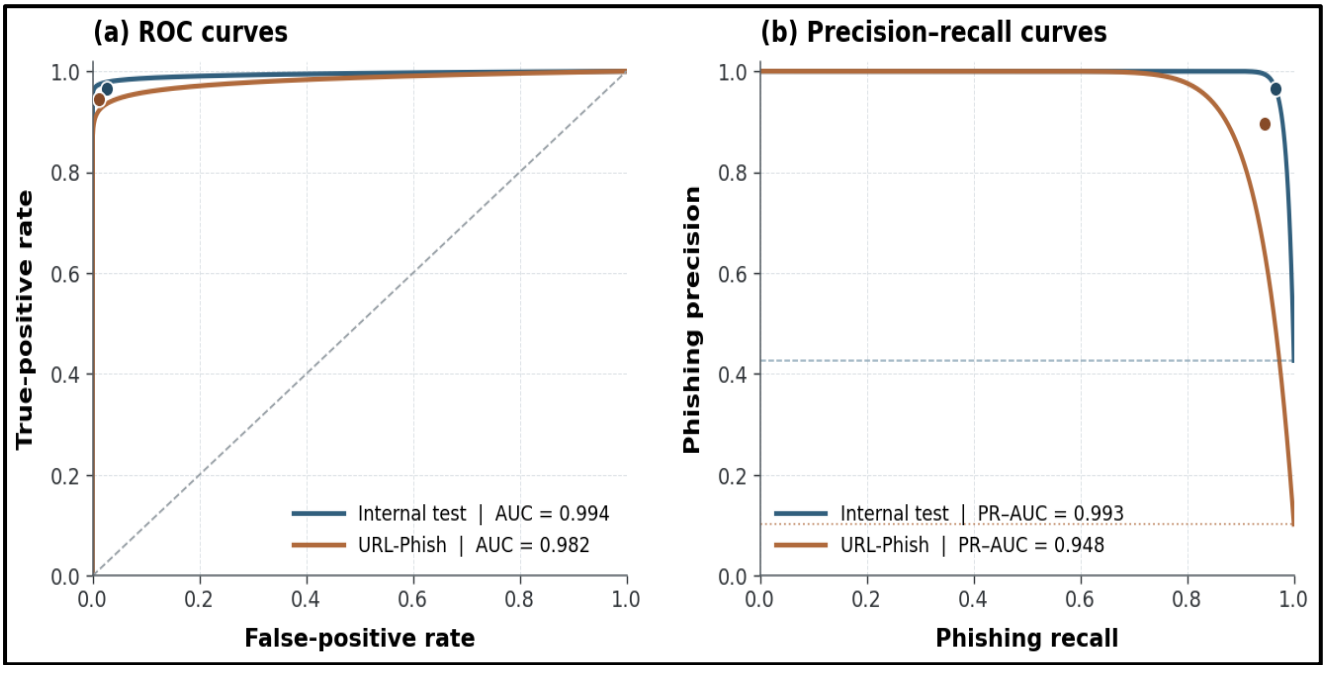

Figure 6. Internal and external ROC and precision–recall curves of the proposed framework.

ROC–AUC decreased from 0.994 to 0.982, while PR–AUC declined from 0.993 to 0.948 under cross-dataset evaluation. The PR–AUC reduction was greater than the ROC–AUC change as the external data set had much fewer phishing instances. However, the detector still maintained a 94.57% phishing recall, meaning that only 600 out of 11,048 phishing attacks were missed.

4.4 Ablation and Adaptive-Fusion Behaviour

The results of the ablation in Table 8, Panel A, show that each feature branch was able to provide some information. The biggest drop was seen when the CNN branch was removed, which saw a drop of 1.41 points in macro-F1. This branch proved to be very helpful for brand-token manipulation, for the repetition of delimiters, and for short local obfuscation. Removing numerical features reduced macro-F1 by 0.87 points, while replacing attention with the final BiLSTM state reduced it by 0.99 points.

Table 8. Detector ablation and cyber threat intelligence evaluation. Panel A. Detection-component ablation

Configuration	Accuracy (%)	Phishing recall (%)	Macro-F1 (%)	MCC	Change in macro-F1
Complete proposed framework	97.03	96.51	96.96	0.939	-
Without CNN branch	95.65	94.59	95.55	0.911	−1.41
Without BiLSTM attention	96.06	95.21	95.97	0.919	−0.99
Without numerical features	96.18	95.31	96.09	0.922	−0.87
Direct concatenation	96.62	96.04	96.54	0.931	−0.42
Fixed threshold of 0.50	96.46	96.97	96.39	0.928	−0.57

Panel B. CTI enrichment and ablation results on 300 expert-reviewed alerts

CTI configuration	Indicator completeness (%)	Entity F1 (%)	ATT&CK mapping F1 (%)	Mean actionability (/5)	End-to-end latency (ms)
Detector probability only	61.9	-	-	2.41	2.08
Exact feed match	72.4	78.9	68.2	3.16	2.74
Unweighted evidence fusion	86.1	89.7	83.4	4.02	5.78
Without temporal freshness	91.6	93.8	89.1	4.18	6.21
Without source confidence	91.4	92.6	87.9	4.11	6.18
Without entity correlation	84.7	86.2	79.8	3.74	5.91
Complete evidence-weighted CTI	92.8	93.8	90.7	4.46	6.34

The fixed threshold produced slightly higher recall than the calibrated model, but false positives increased from 505 to 765. Consequently, macro-F1 and MCC declined. This result supports the use of validation-based threshold selection instead of treating 0.50 as an operationally neutral value.

Figure 7 illustrates both the ablation losses and the mean gate allocation for three representative URL conditions. The CNN branch received a mean weight of 0.48 for heavily obfuscated URLs. The sequential branch dominated long paths and encoded queries with a weight of 0.53, whereas the numerical branch increased to 0.53 for IP-hosted and structurally unusual domains.

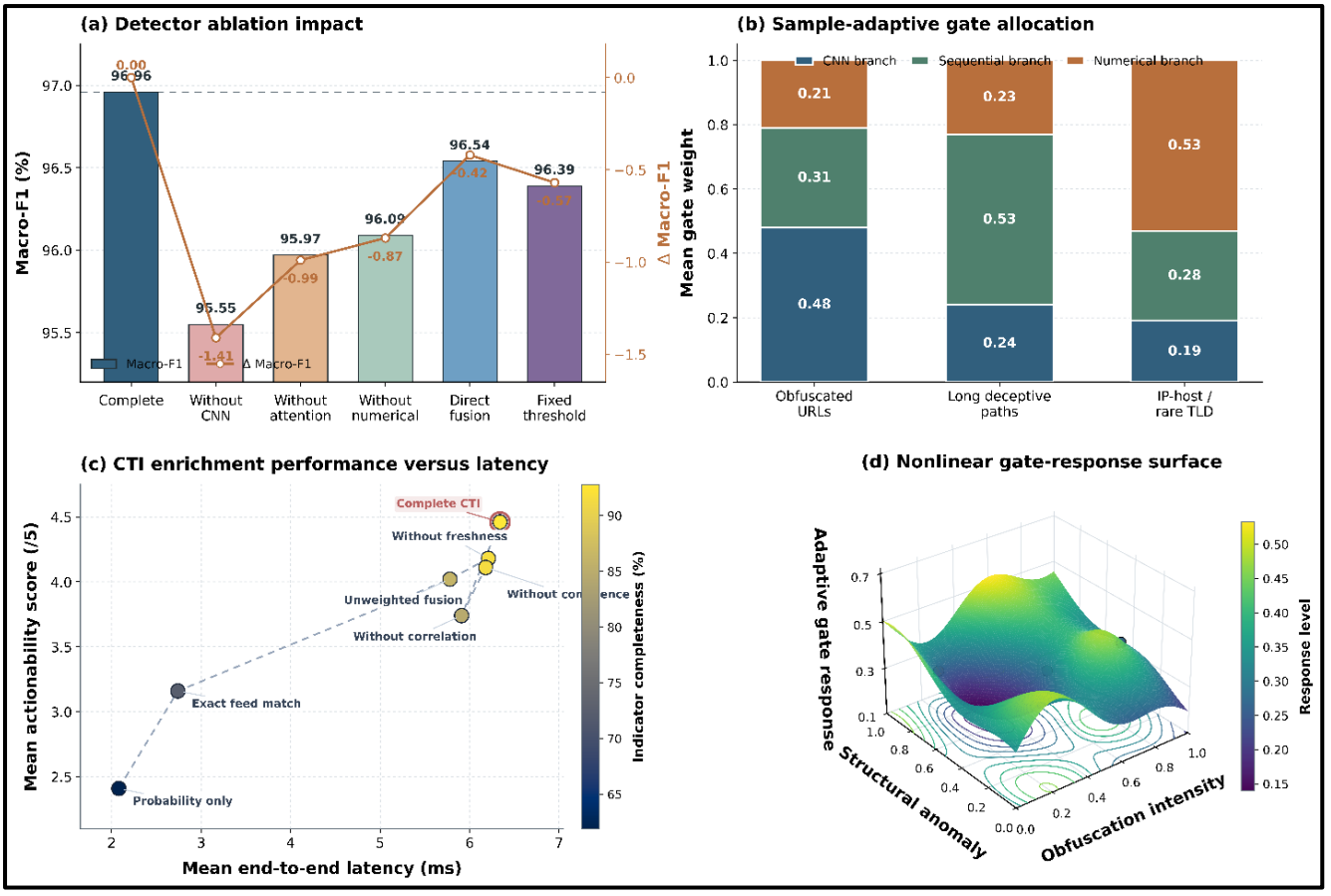

Figure 7. Component contribution and sample-adaptive gate behaviour.

The figure combines macro-F1 changes under ablation with branch weights for obfuscated URLs, long deceptive paths, and IP-host or rare-TLD cases. The three branch weights sum to one for every sample. These changes indicate that the gate was not set to a particular preference.

4.5 CTI Quality and Operational Cost

The results of the complete intelligence layer were the best in terms of context, as shown in Panel B of Table 9. It was able to complete 2,226 of 2,400 required CTI fields, resulting in a 92.8% indicator-completeness rate. The results for entity-level F1 were 93.8% and evidence-supported ATT&CK mapping was 90.7%. 300 bundles were exported and all 300 of them passed the STIX 2.1 schema validation. The reports created were independently evaluated by two reviewers. The inter-rater agreement was κ=0.86 and 267 of the 300 alerts were given an actionability score of 4 or higher. This gave a mean score of 4.46 which was significantly higher than a mean score of 3.16 with exact feed matching. The difference is an important operational consideration: a feed match is a match on an indicator, but it doesn't necessarily mean that the indicator is fresh, that it is related to some infrastructure, that it has a high level of confidence, or that it is relevant to defense. The actionability of CTI is not only based on the amount of information, but also on these qualities [11], [15].

The highest amount of CTI degradation was observed when entity correlation was removed. The number of points for ATT&CK mapping F1 dropped 10.9 points, and actionability dropped from 4.46 to 3.74. The effects of temporal freshness and source confidence were smaller but both helped to keep the indicators from becoming stale or from being poorly supported. This behaviour is in line with threat intelligence analysis, where the source linkage and attribution evidence should be traceable [41].

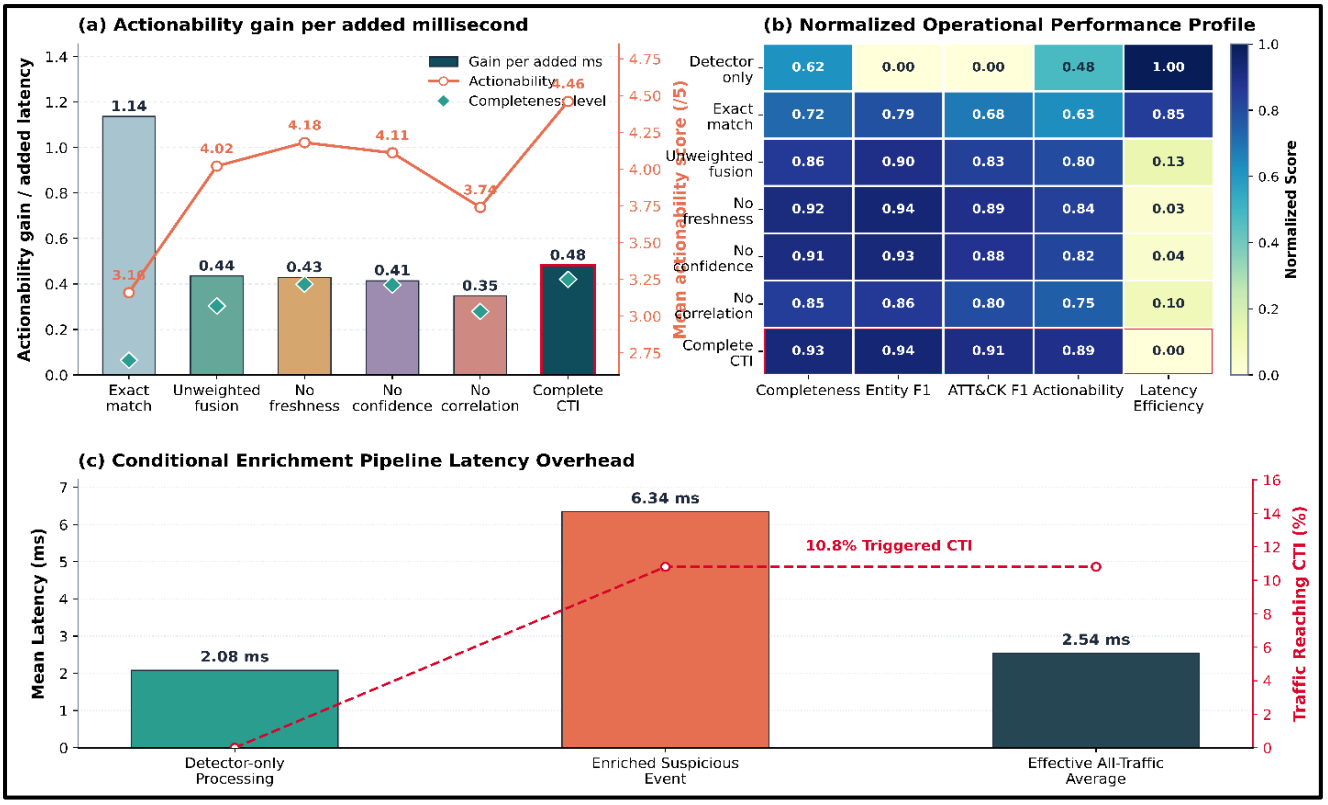

Figure 8. Accuracy–actionability–latency trade-off among CTI configurations.

Evidence-weighted CTI raised the mean actionability to 4.46, and took 6.34ms for suspicious events, whereas detector-only output took 2.08ms. The CTI stage took 4.26 ms, but was only performed for URLs that exceeded the phishing threshold that was calibrated. 11,648 of 107,848 URLs were fed into the enrichment on the external dataset. The average latency for all traffic was thus ~2.54ms/URL, instead of 6.34ms/URL. This conditional design maintained high-speed detection of events, and offered more evidence for the smaller number of high-risk events.

Overall, the results show that the primary benefit of the proposed framework was not that it achieved a single large boost in internal accuracy. It's its greater impact in reducing false negatives, the lesser cross-dataset degradation, its ability to adapt to using heterogeneous URL evidence, and its ability to transform suspicious predictions into structured, actionable intelligence.

5. Discussion

The results show that the main advantage of the proposed framework is not only to achieve a higher classification accuracy but also to maintain the performance in the event of a change in the data distribution. The external dataset shows a smaller drop in the dependence on the dataset-specific lexical patterns, which indicates that the sample-adaptive fusion reduced the dependence on the dataset-specific lexical patterns. For highly obfuscated URLs, the gate was more likely to assign a higher weight to the evidence of local convolution, while the evidence of sequential representation and numerical representation was more influential for long paths and encoded parameters respectively, and the evidence of structurally abnormal domains was more influential for structurally abnormal domains. This behaviour is in line with the hypothesis that phishing indicators are not homogeneous and cannot be fused by using fixed feature weights.

The proposed method outperformed the conventional CNN–BiLSTM and direct-concatenation models in terms of a more balanced recall–control–stability trade-off, and cross-dataset stability. This builds on previous hybrid phishing systems that showed the advantages of using lexical and sequential representations, but failed to dynamically adjust the amount of each representation used for each URL [43, 44]. The enhanced external performance is also applicable to the generalizations

problem pointed out by Rashid et al. [39] where models learned from a particular URL distribution tend to break down when faced with new URL distributions.

The second practical contribution is from the CTI layer. It doesn't end with a binary prediction, but rather transforms suspicious events into intelligence that is machine-readable, temporally qualified and has a confidence score. This is more appropriate for SIEM, cloud gateways, browser protection and automated incident-response pipelines. The significant drop in the number of records retrieved after entity correlation was removed further demonstrates that intelligence actionability is dependent on relationships between indicators, rather than just the number of records retrieved. This is in line with the actionability-centred CTI evaluation [15] and threat-intelligence driven attribution [41].

The evaluation is based on publicly available URL datasets and snapshots of the intelligence frozen, which may not accurately reflect the multilingual spear-phishing or compromised legitimate domains or quickly disappearing campaigns. The completeness of the sources and their expert annotation are also important for the CTI quality. Future work should thus explore data that evolves over time, privacy-preserving collaborative learning, more comprehensive behavioural evidence, and deployment in real-world cloud-security contexts.

6. Conclusion and Future Scope

This research created a single framework to identify phishing URLs in the early stages and subsequently transform high risk predictions into cyber threat intelligence in a structured format. The main contribution is the sample-adaptive fusion of convolutional, sequential, and numerical URL evidence, as opposed to giving equal importance to all the sources of URL evidence. The proposed model exhibited a higher macro-F1 score of 96.96% on the internal test set and a higher score of 95.57% on the cross-dataset test set compared to the conventional model and the direct-fusion model. The CTI layer also added features to enhance the practical utility, such as source confidence, freshness, entity correlation, severity assessment, and STIX-compatible reporting. It's still confined within public URL datasets and intelligence snapshots that are frozen. These sources are not able to cover all multilingual, ephemeral and/or targeted phishing campaigns. In addition, the quality of the real-world feed can impact reliability of enrichment. The framework should be tested in future studies with enterprise traffic that is separated in time, with compromised legitimate domains, with email context, with DNS behaviour and with signals of the interaction with the browser. Other areas of interest include continual learning, updates to the federated model, uncertainty-aware routing, and mechanisms for the analysts to feed back. The next useful step would be to deploy a controlled cloud or security-operations-centre.

Declarations

Acknowledgements
NA

Funding
NA

Contributions
All authors; Conceptualization, All authors; Investigation; All authors; Writing (Original Draft), All authors; Writing (Review and Editing) Supervision, All authors; Project Administration.

Ethics declarations
This article does not contain any studies with human participants or animals performed by any of the authors.

Data Availability Statement
The processed datasets, the source code, the configuration, the split indices and the evaluation output are provided with the supplementary materials of this study. Public data from UCI PhiUSIIL repository, URL-Phish, PhishTank, OpenPhish, Tranco and MITRE ATT&CK were used. Restricted intelligence feeds are subject to the terms of access and redistribution for each of them.

Use of Generative Artificial Intelligence
The authors declare that no generative artificial intelligence tools were used in the preparation of this manuscript.

Competing interests
All authors declare no competing interests.

References

[1] Hnamte, V., Nhung-Nguyen, H., Hussain, J., & Hwa-Kim, Y. (2023). A novel two-stage deep learning model for network intrusion detection: LSTM-AE. *IEEE Access, 11*, 37131–37148. https://doi.org/10.1109/ACCESS.2023.3266979

[2] Hosseinzadeh, M., Ali, U., Ali, S., Abbaszadi, R., Gharehchopogh, F. S., Khoshvaght, P., ... & Lansky, J. (2025). Improving phishing email detection performance through deep learning with adaptive optimization. *Scientific Reports*, *15*(1), 36724. https://doi.org/10.1038/s41598-025-20668-5

[3] Charmet, F., Morikawa, T., Tanaka, A., & Takahashi, T. (2024). VORTEX: Visual phishing detectiOns aRe through EXplanations. *ACM Transactions on Internet Technology*, *24*(2), Article 9. https://doi.org/10.1145/3654665

[4] Pira, L., & Ferrie, C. (2024). On the interpretability of quantum neural networks. *Quantum Machine Intelligence*, *6*(2), 52.

[5] Kailas, S., & Roopalakshmi, R. (2025). "Think before you click"—Malicious URL detection in cybersecurity: A systematic review and research roadmap. *IEEE Access, 13*, 154305–154325. https://doi.org/10.1109/ACCESS.2025.3601387

[6] Sajid, M., Malik, K. R., Almogren, A., Malik, T. S., Khan, A. H., Tanveer, J., & Rehman, A. U. (2024). Enhancing intrusion detection: a hybrid machine and deep learning approach. *Journal of Cloud Computing*, *13*(1), 123.

[7] Kongsorot, Y., Musikawan, P., Aimtongkham, P., You, I., Benslimane, A., & So-In, C. (2023). An intrusion detection and identification system for Internet of Things networks using a hybrid ensemble deep learning framework. *IEEE Transactions on Sustainable Computing*, *8*(4), 596–613. https://doi.org/10.1109/TSUSC.2023.3303422

[8] Wisanwanichthan, T., & Thammawichai, M. (2021). A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM. *IEEE access*, *9*, 138432-138450.

[9] Panda, M., Abraham, A., & Patra, M. R. (2012). A hybrid intelligent approach for network intrusion detection. *Procedia engineering*, *30*, 1-9.

[10] Goenka, R., Chawla, M., & Tiwari, N. (2025). Enhanced phishing detection approach using a layered model: Domain squatting and URL obfuscation identification and lexical feature-based classification. *IEEE Access, 13*, 187285–187306. https://doi.org/10.1109/ACCESS.2025.3626819

[11] Cohen, D., Te'eni, D., Yahav, I., Zagalsky, A., Schwartz, D., Silverman, G., ... & Makowski, J. (2025). Human–AI enhancement of cyber threat intelligence. *International Journal of Information Security*, *24*(2), 99. https://doi.org/10.1007/s10207-025-01004-4

[12] Chung, Y. Y., & Wahid, N. (2012). A hybrid network intrusion detection system using simplified swarm optimization (SSO). *Applied soft computing*, *12*(9), 3014-3022.

[13] Kulkarni, A., Balachandran, V., & Das, T. (2025). Phishing webpage detection: Unveiling the threat landscape and investigating detection techniques. *IEEE Communications Surveys & Tutorials, 27*(2), 974–1007. https://doi.org/10.1109/COMST.2024.3441752

[14] Mamodiya, U., Kishor, I., Naz, R., Almaiah, M., & Alqutaish, A. (2026). A hybrid blockchain-based framework for adaptive cyber-risk prediction and multi-layer threat mitigation in enterprise networks. *Journal of Cybersecurity and Privacy, 6*(3), 85. https://doi.org/10.3390/jcp6030085

[15] Dimitriadis, A., Papoutsis, A., Kavalieros, D., Tsikrika, T., Vrochidis, S., & Kompatsiaris, I. (2025). EVACTI: evaluating the actionability of cyber threat intelligence: A. Dimitriadis et al. *International Journal of Information Security*, *24*(3), 123. https://doi.org/10.1007/s10207-025-01033-z

[16] Brezeanu, G., Archip, A., & Artene, C.-G. (2025). Phish Fighter: Self updating machine learning shield against phishing kits based on HTML code analysis. *IEEE Access, 13*, 4460–4486. https://doi.org/10.1109/ACCESS.2025.3525998

[17] Chen, Z., Liu, S.-Z., Huang, J., Xiu, Y.-H., Zhang, H., & Long, H.-X. (2024). Ethereum phishing scam detection based on data augmentation method and hybrid graph neural network model. *Sensors, 24*(12), 4022. https://doi.org/10.3390/s24124022

[18] Zhang, J., Sui, H., Sun, X., Ge, C., Zhou, L., & Susilo, W. (2024). GrabPhisher: Phishing scams detection in Ethereum via temporally evolving GNNs. *IEEE Transactions on Services Computing, 17*(6), 3727–3741. https://doi.org/10.1109/TSC.2024.3411449

[19] Schmitt, M. (2023). Securing the digital world: Protecting smart infrastructures and digital industries with artificial intelligence (AI)-enabled malware and intrusion detection. *Journal of Industrial Information Integration, 36*, 100520. https://doi.org/10.1016/j.jii.2023.100520

[20] Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE access*, *8*, 32464-32476.

[21] Babu, D. R. K., & Packialatha, A. (2023). Hybrid classification model with tuned weight for cyber attack detection: Big data perspective. *Advances in Engineering Software, 177*, 103408. https://doi.org/10.1016/j.advengsoft.2022.103408

[22] Jayanthi, S., Bavirthi, S. S., Murali, P., Kumar, K. V., Alkahtani, H. K., Ishak, M. K., & Mostafa, S. M. (2025). Advancements in cyberthreat intelligence through resource exhaustion attack detection using hybrid deep learning with heuristic search algorithms. *Scientific Reports*, *15*(1), 30461. https://doi.org/10.1038/s41598-025-13305-8

[23] Govindarajan, M., & Chandrasekaran, R. M. (2011). Intrusion detection using neural based hybrid classification methods. *Computer networks*, *55*(8), 1662-1671.

[24] Asiri, S., Xiao, Y., Alzahrani, S., Li, S., & Li, T. (2023). A survey of intelligent detection designs of HTML URL phishing attacks. *IEEE Access, 11*, 6421–6443. https://doi.org/10.1109/ACCESS.2023.3237798

[25] Susilo, B., Muis, A., & Sari, R. F. (2025). Intelligent intrusion detection system against various attacks based on a hybrid deep learning algorithm. *Sensors*, *25*(2), 580. https://doi.org/10.3390/s25020580

[26] Gayathri, G. R., Sajjanhar, A., & Xiang, Y. (2024). Hybrid deep learning model using SPCAGAN augmentation for insider threat analysis. *Expert Systems with Applications*, *249*, 123533. https://doi.org/10.1016/j.eswa.2024.123533

[27] Abbas, A., Salahuddin, M., Khan, M. Z., Khan, A. A., Zaman, F. U., Inam, S. A., ... & Khan, M. A. (2025). Machine learning-based hybrid technique to enhance cyber-attack perspective. *Journal of Cloud Computing*, *14*(1), 57. https://doi.org/10.1186/s13677-025-00782-5

[28] Yang, T., & Sun, J. (2025). A hybrid ensemble deep learning framework with novel metaheuristic optimization for scalable malicious website detection. *Scientific Reports, 15*, 44630. https://doi.org/10.1038/s41598-025-33695-z

[29] Manivannan, A., & Amalanathan, A. (2025). GeoGuard: a hybrid deep learning intrusion detection system with integrated geo-intelligence and contextual awareness. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3619557

[30] Song, Y., Zhang, D., Wang, J., Wang, Y., Wang, Y., & Ding, P. (2025). Application of deep learning in malware detection: a review. *Journal of Big Data*, *12*(1), 99. https://doi.org/10.1186/s40537-025-01157-y

[31] Henry, S., Gautam, S., Khanna, S., Rabie, K., Shongwe, T., Bhattacharya, P., Sharma, B., & Chowdhury, S. (2023). Composition of hybrid deep learning model and feature optimization for intrusion detection system. *Sensors, 23*(2), 890. https://doi.org/10.3390/s23020890

[32] Arya, K., Siddhant, S., & Upadhyay, L. (2025). An explainable hybrid deep learning framework for network intrusion detection using feature-guided CNN models. *IEEE Access, 13*, 204954–204977. https://doi.org/10.1109/ACCESS.2025.3637857

[33] Dong, J.-D., Crichton, K., Yamada, A., Sawaya, Y., Cranor, L., & Christin, N. (2026). Accurate, generalizable, and practical behavioral models to identify impending user exposure to malicious websites. *ACM Transactions on the Web, 20*(1), Article 1, 1–31. https://doi.org/10.1145/3768587

[34] Mbura, R. K., Kato Benedicto, A., & Sinde, R. (2026). A novel hybrid approach for identification of discriminative features in phishing emails. *IEEE Access, 14*, 995–1013. https://doi.org/10.1109/ACCESS.2025.3649636

[35] Reda, S. A., Taie, S., & 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

[36] Li, W., Manickam, S., & Chong, Y. W. (2025). FedPhishLLM: A privacy-preserving and explainable phishing detection mechanism using federated learning and LLMs. *Journal of King Saud University–Computer and Information Sciences, 37*, 252. https://doi.org/10.1007/s44443-025-00267-0

[37] Khan, S., Dilshad, N., Ahmad, N., Noor, S., & AlQahtani, S. A. (2025). Integrating AI in security information and event management for real time cyber defense. *Scientific reports, 15*(1), 35872. https://doi.org/10.1038/s41598-025-19689-x

[38] Ahmed, M., Altamimi, A. B., Khan, W., Alsaffar, M., Ahmad, A., Khan, Z. H., & Alreshidi, A. (2023). PhishCatcher: client-side defense against web spoofing attacks using machine learning. *IEEE Access*, *11*, 61249-61263.

[39] Rashid, F., Doyle, B., Han, S. C., & Seneviratne, S. (2024). Phishing URL detection generalisation using unsupervised domain adaptation. *Computer Networks, 245*, 110398. https://doi.org/10.1016/j.comnet.2024.110398

[40] Mohammadzad, M., Karimpour, J., & Mahan, F. (2024). Cyber attacker's next action prediction on dynamic real-time behavior model. *Computers & Electrical Engineering, 113*, 109031. https://doi.org/10.1016/j.compeleceng.2023.109031

[41] Wang, Z., Zhou, Y., Liu, H., Qiu, J., Fang, B., & Tian, Z. (2024). ThreatInsight: Innovating early threat detection through threat-intelligence-driven analysis and attribution. *IEEE Transactions on Knowledge and Data Engineering, 36*(12), 9388–9402. https://doi.org/10.1109/TKDE.2024.3474792

[42] Kavya, S., & Sumathi, D. (2025). Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection. *Artificial Intelligence Review, 58*, 50. https://doi.org/10.1007/s10462-024-11055-z

[43] Alsubaei, F. S., Almazroi, A. A., & Ayub, N. (2024). Enhancing phishing detection: A novel hybrid deep learning framework for cybercrime forensics. *IEEE Access, 12*, 8373–8389. https://doi.org/10.1109/ACCESS.2024.3351946

[44] Elberri, M. A., Tokeşer, Ü., Rahebi, J., & Lopez-Guede, J. M. (2024). A cyber defense system against phishing attacks with deep learning game theory and LSTM-CNN with African vulture optimization algorithm (AVOA). *International Journal of Information Security*, *23*(4), 2583-2606. https://doi.org/10.1007/s10207-024-00851-x

[45] Lee, S., Lee, K., Cho, S., & Choi, C. (2025). APTStop: A real-time framework for APT defense via strategic threat observation and prediction. *IEEE Access, 13*, 183134–183155. https://doi.org/10.1109/ACCESS.2025.3624035