JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 11 Issue 02
2026
Research Article
https://doi.org/10.56979/1102/2026/1608

An Explainable and Adversarially Robust Multimodal Deep Learning–LLM Framework for Arabic Web Phishing Detection

Adnan H. Al-Helali[1*]

[1]Department of Cybersecurity, Faculty of Science and Information Technology, Irbid National University, Irbid, Jordan.
[*]Correspondence Author: Adnan H. Al-Helali. Email: adnan_hadi@inu.edu.jo

Received: July 03, 2026 **Accepted:** August 31, 2026

Abstract: Arabic-language phishing websites are under-served by detectors built for English content and a single input modality. We present a multimodal deep learning–large language model (LLM) framework for Arabic web phishing detection and evaluate its URL/domain branch and a URL-level LLM stage. The corpus is a frozen public-source set of 9,129 Arabic-targeted URLs (2,554 phishing; 7,884 registered domains); template-level de-duplication and campaign clustering keep every registered domain and phishing template within one split. We test unseen-country generalisation, a five-edit evasion suite with and without adversarial training, SHAP faithfulness, and a zero-shot hosted LLM (clean comparison, four-way comparison and prompt-injection test). From the hostname alone, a character CNN with 36 handcrafted features reaches F1 = 0.697 (95% CI 0.665–0.724) and AUC 0.860; adding the path gives F1 = 0.944, partly a path-length artefact. F1 falls to 0.35–0.55 on an unseen country, and at 1% prevalence the model would raise about 802 false alerts per 10,000 pages. Adversarial training raised phishing recall at the worst edit budget from 0.18 to 0.63 without loss of clean F1. Removing the three top SHAP features lowered the phishing score 3.9 times more than removing random features. The LLM gave no clean-data benefit on 1,371 test URLs (baseline F1 0.724; LLM-enhanced 0.695; gain interval -0.058 to -0.001). Under evasion it raised recall from 0.13 to 0.31 at doubled false alerts, whereas adversarial training raised it to 0.59. Arabic, English, mixed and obfuscated injection payloads did not significantly reduce LLM detection. Adversarial training, not the LLM, gave the main robustness gain. Only the URL branch and a URL-level LLM stage were evaluated; the text, DOM and visual branches are specified but untested.

Keywords: Large Language Model; explainable AI; Arabic Phishing Detection; Multimodal Deep Learning; Adversarial Robustness

1. Introduction

Social engineering is the most common first access method used in the reported cybercrime, in which users are asked to provide credentials, financial information, or install malware on their machines via spoofed Web pages that masquerade as trusted brands. Global anti-phishing monitoring organisations have been reporting increasing numbers of phishing attacks on financial services, government portals and communication providers every quarter recently [1, 2]. Although most of the automated phishing detection research has focused on content in English, Arabic is a language spoken by over 400 million people in the Middle East and North Africa, and Arabic users are targets for phishing attacks that exploit ambiguities in Arabic script,

quirks of the right-to-left rendering used by many Arabic sites, Arabic script-Latin script homograph substitutions, and use of dialectal variation that is poorly handled by generic multilingual models [3], [4], [5].

The current methods of detecting web phishing can be categorized into three main types: feature-engineered machine learning models based on the lexical statistics of URLs, deep learning models trained to represent URLs from raw URL, HTML source, or images of the webpage, and more recently, the large language model (LLM)-based detector reason over the page content in natural language [6], [2], [7]. These are complementary strengths and weaknesses for each of these families. Easy to evade with lightweight and fast, URL based models. Text-based models can only detect social-engineering language but not structural and visual cues. Similarity models using visual features are able to identify brand impersonation, but at the cost of being computationally costly, and sensitive to minor rendering perturbations. Prompt-based LLM detectors offer high semantic understanding and natural language explanations, but are not as discriminative as specialized deep classifiers, and may be expensive when deployed at scale [8, 9]. More importantly, none of these methods have been developed, trained, and tested specifically for Arabic web content, and only a few of those methods have been developed to include multiple modalities, adverse robustness guarantees, and end-to-end explainability.

This paper addresses these gaps by specifying a unified framework that fuses four modalities, URL/domain features, Arabic webpage text, HTML/DOM structure, and visual screenshots, through a cross-modal attention mechanism, and augments the fused representation with LLM-based semantic reasoning specialised for Arabic phishing cues; adversarial training and layered explanation outputs complete the design. The empirical part covers the URL/domain branch, evaluated under leakage-controlled splits, unseen-country/rule/kit tests, prevalence-adjusted operating points and a budgeted evasion suite, and a URL-level LLM stage evaluated through a hosted model.

The main contributions of this paper are summarised as follows:

- A multimodal deep learning–LLM framework for Arabic web phishing detection that fuses URL, Arabic text, HTML/DOM and visual modalities through cross-modal co-attention and gated LLM decision fusion, with the full prompt, generation settings and parsing rules of the LLM module.
- A budgeted adversarial-training formulation combining continuous perturbation with discrete feature-space evasion, instantiated and evaluated for the URL/domain branch.
- A frozen, documented, public-source corpus of 9,129 Arabic-targeted URLs with explicit targeting rules, template-level de-duplication, campaign clustering, cluster-disjoint splits and a hashed manifest with rebuild scripts.
- A reproducible evaluation in which all tables and figures are generated from a single prediction file, with cluster-bootstrap confidence intervals, unseen-country, unseen-rule and unseen-kit tests, prevalence-adjusted false-alert burden, and a two-factor (base versus adversarially trained) robustness comparison.
- An explainability analysis of the URL branch with a deletion-based faithfulness test.
- A measured evaluation of a zero-shot URL-level LLM stage (baseline, LLM-alone and LLM-enhanced models; a four-way comparison with adversarial training; a prompt-injection test with Arabic, English, mixed-language and obfuscated payloads; a repeatability check), with scripts, prompts and model answers released.

The remainder of this paper is organised as follows. A review of the related work in the field of machine learning, deep learning, Arabic NLP, multimodal learning, LLM-based cybersecurity, and explainable phishing detection is provided in Section 2, which also highlights the research gap. Details of the proposed framework are provided in Section 3. Construction and preprocessing of the datasets are described in Section 4. The methodology used in the experiments is described in section 5. Results and analysis in section 6. Explainability and security analysis is presented in Section 7. Findings, comparisons, implications and limitations are discussed in Section 8 and the paper is concluded in Section 9.

2. Related Work

2.1. Machine Learning-Based Web Phishing Detection

The early phishing detection studies used hand-made lexical features, host-based features and blacklist features to be fed to a variety of classifiers such as logistic regression, support vector machines, random forests and gradient-boosted trees. However, the small size of Arabic phishing corpora and the potential of hybrid ML-DL pipelines are demonstrated by a 418-email Arabic Phishing Dataset created by Ghabban [3] for phishing email detection using a hybrid BiLSTM-GRU-random forest model with N-gram sentence structure features, which achieved an accuracy of 98.81%. Other feature-selection methods that have been considered for complementary use are also studied; An Arabic phishing email detection method using genetic-algorithm-based feature selection was demonstrated to significantly reduce the number of features without compromising the classification accuracy of several classical learners [10]. Recent studies have also attempted to measure the fragility of feature-engineered detectors when applied to distributional change, demonstrating that concept drift in phishing feature distributions can cause production classifiers to become inaccurate over time and highlighting the need to periodically retrain and provide drift-aware evaluation procedures [11]. At larger-scale, an optimized ensemble stacking study [4] has used a large Arabic phishing email dataset and has evaluated individual and stacked ensemble classifiers showing again that ensemble methods consistently outperform all individual classifiers for the Arabic phishing emails, but still are limited to using email text, and not any webpage content. Although effective and efficient in computation, ML-based methods are highly data- and feature-dependent and are not particularly good generalisers for novel phishing kits, and only give shallow interpretability in the form of feature-importance scores, not semantic understanding of the attacker intent.

2.2. Deep Learning for Phishing Detection

Recently, deep learning has been widely used to learn features that are discriminative to phishing directly from raw inputs, thus avoiding manual feature engineering. CNNs and RNNs trained on raw URL character sequences extract sub-word and n-gram patterns characteristic of typosquatting and homograph attacks, while attention mechanisms trained on HTTP headers and DOM sequences are able to learn about longer range structural patterns. Federated and continual-learning variants of attention-based phishing classifiers have also been suggested to reduce the amount of sensitive browsing data that needs to be centralized to be used in the training process while maintaining the robustness against concept drift [1, 11]. Bilingual CNN-based detectors using particle swarm optimisation (PSO) have also been proposed for English-Arabic phishing email classification [5] to show that the metaheuristic hyperparameter search can be meaningfully applied to improve the phishing classifiers that are based on convolutional neural networks in multilingual settings. However, although most deep learning phishing detectors use either URL or textual or screenshot images of web pages, they typically do not use adversarial training, and they are seldom able to withstand the evasion strategies that are more and more frequently used by malicious pages created by attackers using generative tools to obfuscate their web pages [12].

2.3. Arabic NLP and Arabic Web Phishing

In the field of Arabic NLP, Arabic pretrained transformer encoders like AraBERT, ARBERT/MARBERT, AraELECTRA, and CAMeLBERT have been developed to a great extent and now serve as the backbone for various NLP tasks including text classification, dialect identification and fake content detection [13, 14]. BiLSTM+Attention on top of AraBERT, an architecture designed to capture both semantic and sequence information, was proposed in AraBLA, an architecture extended by all twelve layers of AraBERT, and was found to boost Arabic fake-news detection over AraBERT, which uses only one single layer, by demonstrating its ability to extract more deep semantic and sequential information [15], an idea that is directly relevant to Arabic phishing text that combines formal register with urgency-inducing colloquial phrasing. Hybrid stacked transformer models, such as AraBERT or CAMeLBERT with BiLSTM/BiGRU layers

are also found to be effective for Arabic dialect detection [16] and the use of pretrained Arabic transformer consistently shows better downstream Arabic classification accuracy. However, while the overall progress in general Arabic NLP has been made, there are still not enough dedicated Arabic web-phishing datasets and models available (the existing Arabic phishing studies deal mostly with the text of the email or SMS-embedded URL, but do not model full webpage content along with URL, structural, and visual evidence [3, 4, 10]).

2.4. Multimodal Learning for Phishing Detection

Multimodal phishing detection combines two or more of URL, text, HTML, and visual evidence to overcome the blind spots of unimodal detectors. Visual similarity systems compare rendered screenshots against known brand references to detect logo and layout impersonation, while multimodal transformer architectures have been used to jointly model text and images for related tasks such as harmful meme detection through multimodal debate between large language models [17], illustrating the value of cross-modal reasoning for deceptive content more broadly [18]. Large multimodal agent approaches have also been proposed to jointly analyse URLs and webpage screenshots with token-optimised prompting to reduce LLM inference cost [8], and multimodal large language models have been evaluated directly on phishing webpage detection and brand identification tasks [19], reporting strong zero-shot performance but at substantially higher latency and cost than specialised discriminative models [9]. These studies collectively motivate cross-modal fusion but stop short of combining four complementary modalities with a dedicated, trainable fusion network and language-specific handling of Arabic content.

2.5. Large Language Models for Cybersecurity

LLMs have rapidly been incorporated into cybersecurity pipelines for tasks ranging from vulnerability detection to security-operations-centre triage. Systematic reviews of LLMs for cybersecurity catalogue both offensive misuse-such as LLM-assisted spear-phishing content generation-and defensive applications including phishing email classification and explanation generation [20], [21], [22], [7]. LLM-integrated applications that ingest untrusted web content are in addition exposed to indirect prompt injection [23], [24]. ChatSpamDetector and related prompting-based systems demonstrated that carefully engineered prompts allow general-purpose LLMs to classify phishing emails with high accuracy while producing natural-language justifications [25], and EXPLICATE [6] combined LLM-generated explanations with classical explainable-AI techniques to improve user trust in automated phishing verdicts. Multi-agent LLM systems such as MultiPhishGuard [2] and PhishDebate [26] decompose phishing analysis into specialised reasoning agents that debate or vote on a final verdict, improving robustness over single-prompt classification [27], while comparative studies of small versus large language models for phishing website detection show that the cost-performance trade-off remains an open practical concern for production deployment [9]. However, purely LLM-based detectors generally underperform specialised discriminative classifiers on precision-critical benchmarks and have been shown to remain susceptible to adversarially crafted webpages, as are visual-similarity detectors [12], [28], motivating hybrid architectures-such as the one proposed in this paper-that combine LLM reasoning with dedicated multimodal deep learning rather than relying on either in isolation.

2.6. Explainable AI for Phishing Detection

Explainability is increasingly recognised as a prerequisite for operational trust in automated phishing detection, since security analysts must be able to justify blocking or quarantine decisions. Post-hoc explanation techniques such as SHAP and LIME have been applied to feature-engineered phishing classifiers to rank URL and HTML feature contributions [6], while attention-weight visualisation and Grad-CAM-style saliency mapping have been used to explain transformer- and CNN-based textual and visual phishing classifiers respectively. EXPLICATE and related LLM-explanation frameworks combine such quantitative attributions with natural-language rationales generated by an LLM, reporting improved analyst

comprehension relative to feature-importance plots alone [6], [2]. Nonetheless, prior explainability work in phishing detection is typically confined to a single modality-most often URL or email text-leaving structural and visual explanation comparatively underexplored, and Arabic-specific textual explainability (e.g., token-level attention over Arabic script with correct right-to-left presentation) has received almost no dedicated treatment in the literature.

2.7. Research Gap

Table 1 summarises representative recent studies against five criteria: Arabic-language support, multimodal input, LLM-based reasoning, adversarial robustness evaluation, and multi-level explainability. The comparison suggests that individual studies address subsets of these criteria; among the studies we reviewed, none jointly covers all five for Arabic web content. Arabic phishing studies remain largely confined to email or SMS text; multimodal phishing frameworks rarely include Arabic content or LLM reasoning; and LLM-based detectors are seldom evaluated for adversarial robustness. This paper does not claim to close the gap empirically: it specifies a framework that addresses all five criteria and evaluates the URL/domain branch of it, as stated in Section 3.1.

Table 1. Representative recent phishing detection studies against the research gap.

Study	Arabic support	Multi-level XAI	Adversarial eval.	Multimodal	LLM reasoning
Ghabban [3] (2026), Arabic email BiLSTM-GRU-RF	Yes	Partial	No	No	No
Al-Shrari et al. [10] (2024), GA feature selection	Yes	Partial	No	No	No
Jbara et al. [4] (2025), ensemble stacking, Arabic email	Yes	No	No	No	No
Salloum et al. [5] (2026), BiD-Phish CNN-PSO bilingual	Yes	No	No	No	No
Trad & Chehab [8] (2024), large multimodal agents	No	No	No	Yes	Yes
Xu et al. [2] (2025), MultiPhishGuard multi-agent LLM	No	Partial	Partial	No	Yes
Kulkarni et al. [12] (2024/25), ML-to-LLM robustness study	No	No	Yes	Yes	Yes
Proposed framework (this paper; design, URL branch evaluated)	Yes (URL-level evaluation)	Yes (URL level, deletion test)	Yes (URL branch)	Specified (not evaluated)	Specified (not evaluated)

3. Proposed Framework

3.1. Framework Overview

The proposed framework, illustrated in Figure 1, receives as input a candidate webpage represented by four raw artefacts: its URL string, rendered HTML/DOM source, extracted Arabic text content, and a full-page screenshot. Four modality-specific pipelines (Sections 3.3-3.6) extract deep features from each artefact using dedicated encoders (Section 3.7). The resulting embeddings are fused through a cross-modal multi-head co-attention module (Section 3.8) that learns which modalities are most informative for a given webpage. In parallel, an LLM-based semantic reasoning module (Section 3.9) analyses the Arabic text and structural summary of the page using Arabic-aware chain-of-thought prompting to assess intent, brand

impersonation, and social-engineering pressure. A gated decision-fusion layer (Section 3.10) is used to combine the multimodal fusion embedding with the LLM reasoning output to generate the final phishing/legitimate verdict and a calibrated confidence score. Last, the explainability module (Section 3.11) produces multi-modal, human-readable rationales across the four modalities.

Scope of the evaluation. Only the URL/domain branch (Sections 3.3 and 3.7), its adversarial training (Section 5.3), its SHAP explanations (Section 3.11) and a URL-level LLM stage (Sections 5.9–5.10 and 6.7–6.9) were implemented and evaluated; the remaining components are specified without results and are drawn dashed in Figure 1.

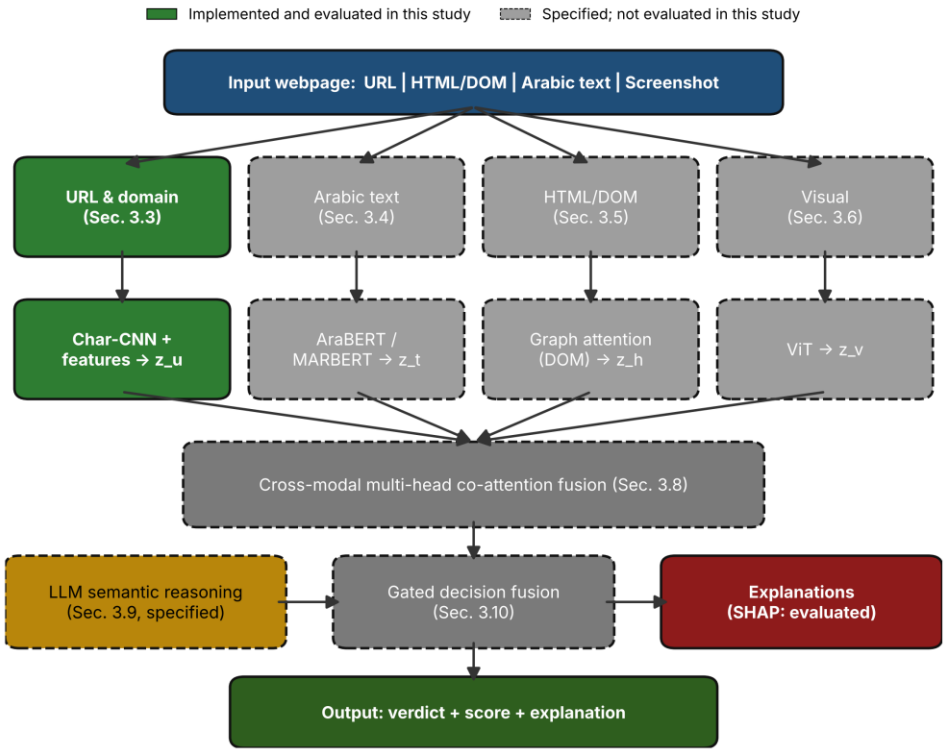


Figure 1. Framework architecture (green: evaluated; dashed grey: specified only).

3.2. Multimodal Data Acquisition

For every candidate page, the acquisition layer fetches the page (resolving the URL and capturing the final redirects) and then downloads the rendered HTML/DOM after JavaScript execution, extracts visible Arabic (and any mixed-language) text uses DOM traversal, and renders a full page screenshot at fixed 1280x1024 as viewed through the browser. To handle live phishing URLs safely, there is a 6-second rendering timeout and sandboxed network egress. The resulting pages that do not render (e.g., because they have been taken down) are kept with the cached HTML/screenshot artefacts, and the corresponding modality-specific training subset is missing, so that the encoders are never trained with empty input.

3.3. URL and Domain Analysis

The implemented URL/domain branch computes 36 offline lexical and host-based features (listed in the supplementary material): hostname length, label and subdomain statistics, digit, hyphen and vowel ratios, entropy, longest consonant run, IP-literal and punycode flags, presence of Arabic script, an Arab-ccTLD indicator, a suspicious-TLD flag, a match against a lexicon of Arab-region institutions and services, security-keyword counts and, for the full-URL view only, path length, depth, query-parameter count, file-extension and keyword indicators. Network-dependent features (TLS certificate, WHOIS age, brand similarity) are part of the fuller design but were not computed, because they need live look-ups. The handcrafted features are concatenated with a character-level embedding of the raw string and consumed by the encoder of Section 3.7.

3.4. Arabic Webpage Text Analysis

Visible Arabic text is extracted from the DOM in reading order, including headings, body paragraphs, form labels, and button/link text, since phishing pages frequently place urgency cues ('يرجى تحديث بيانات حسابك خلال 24 ساعة', i.e., 'please update your account details within 24 hours') in short, salient UI elements rather than long paragraphs. Text is segmented into sentences and normalised (Section 4.4) before being passed to an AraBERT/MARBERT encoder fine-tuned for phishing-intent classification. Auxiliary lexicon-based features-counts of urgency terms, threat terms, and requests for credentials-are computed alongside the contextual embedding to support the explainability module.

3.5. HTML and DOM Structural Analysis

The DOM is parsed into a structural graph in which nodes represent HTML elements (tagged with type, attributes, and visibility) and edges represent parent-child and sibling relationships. Structural indicators known to correlate with phishing kits are computed, including the ratio of external to internal resource links, presence of obfuscated or minified inline JavaScript, hidden form fields, iframe usage, favicon mismatch relative to the claimed brand, and the presence of login/payment forms submitting to a different origin than the visited domain. This graph, with node-level feature vectors, is passed to a graph neural network encoder (Section 3.7), following structural-feature practice established in prior webpage phishing detection work [19].

3.6. Visual Webpage Analysis

The rendered screenshot is resized to 224x224 and processed by a Vision Transformer encoder pretrained on general web-imagery and fine-tuned for phishing brand-impersonation detection. In addition to the global ViT embedding, a lightweight logo-region detector localises candidate brand logos, and their embeddings are compared against a reference bank of 210 legitimate brand logos using cosine similarity, producing a visual brand-mismatch score that flags pages presenting a logo inconsistent with the visited domain.

3.7. Modality-Specific Deep Feature Extraction

Each modality is encoded into a fixed-dimensional embedding. The URL encoder concatenates a 1D character-CNN embedding of the raw URL string (kernel widths 3, 5 and 7, 64 filters each) with a 64-unit projection of the 36 handcrafted features, followed by a two-layer MLP producing z_u (256 dimensions in the full design; 128 hidden units in the evaluated branch). The text, DOM and visual encoders are AraBERT-base [14] with pooled [CLS] representation, a 3-layer graph attention network with mean-pooled node embeddings, and a ViT-Base/16 class-token embedding, each projected to R^256 (z_t, z_h, z_v). Equation (1) defines the general form of each encoder f_m, and Equation (2) the projection that unifies dimensionality before fusion.

$$z_m = f_m(x_m;\, \theta_m),\ \ m \in \{u, t, h, v\} \tag{1}$$

$$\hat{z}_m = W_m\, z_m + b_m,\ \ \hat{z}_m \in \mathbb{R}^d,\ \ d = 256 \tag{2}$$

3.8. Cross-Modal Attention Fusion

The four projected embeddings [z_u, z_t, z_h, z_v] are stacked into a sequence Z ∈ R^{4×d} and passed through L=2 layers of multi-head co-attention, in which each modality attends to all others, allowing, for example, textual urgency cues to be weighted more heavily when structural evidence (e.g., cross-origin form submission) corroborates them. Equation (3) defines the scaled dot-product attention used within each head, and Equation (4) defines the multi-head aggregation. The attended representations are combined via a learned gate α (Equation (5)) that produces the fused multimodal embedding F used for downstream classification and, jointly, for decision fusion with the LLM reasoning branch.

$$Attention(Q, K, V) = softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right) V \tag{3}$$

$$MultiHead(Z) = Concat(head_1, \ldots, head_H)\, W^O \tag{4}$$

$$F = \sum_{m=1}^{4} \alpha_m\, \hat{z}_{m\prime},\ \ \alpha = softmax(W_\alpha\, [\hat{z}_{u\prime};\, \hat{z}_{t\prime};\, \hat{z}_{h\prime};\, \hat{z}_{v\prime}]) \tag{5}$$

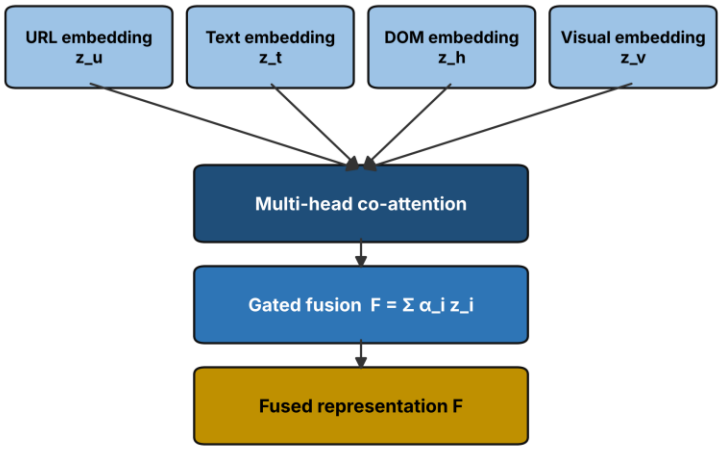


Figure 2. Cross-modal co-attention fusion module (specified only).

3.9. LLM-Based Semantic Reasoning

The LLM reasoning module is described in full so that it can be reproduced; only its URL-level form was run (Sections 5.9, 5.10 and 6.7–6.9). In page-level form it receives a structured prompt containing the extracted Arabic text, a compact structural summary (form targets, external-resource ratio, brand-mismatch score) and the claimed brand or domain, and is asked to assess (i) whether the page requests sensitive information, (ii) whether urgency or threat language is present, and (iii) whether the claimed identity is consistent with the domain.

Model and access. The module is designed for a locally executed, open-weight, instruction-tuned LLM with Arabic coverage, because hidden states are needed to build r_llm, page content then stays in the organisation, and API models change silently. The planned local model (Qwen2.5-1.5B-Instruct, 4-bit GGUF, llama.cpp; Qwen2.5-0.5B-Instruct as fallback) could not be run because its weights could not be downloaded (Section 5.10). The reported LLM results come from a hosted instruction-tuned model reached through an agent interface (alias 'haiku'; checkpoint and decoding settings not controllable): URLs were sent to a third-party service, and reproducibility rests on the released prompts and answers.

Prompt. The prompt has a fixed system part and a user part (reproduced below). Untrusted page text is delimited, truncated to 2,000 tokens, and the system part states that instructions inside the delimiters must be ignored. Six few-shot exemplars (Modern Standard Arabic, Gulf, Levantine and Maghrebi phrasing; three phishing, three legitimate) are drawn from the training split only and are the same for every page; exemplars of dialectal phrasing follow the motivation of [16], [33].

Generation settings and parsing. Decoding is greedy (temperature 0, top-p 1, maximum 384 new tokens, fixed seed) under schema-constrained decoding. The output must validate against the JSON schema shown below; a strict parser rejects extra keys, out-of-range values and non-JSON text. On a validation failure the call is retried once; if it fails again, the page is scored by the deep-learning branch alone, s_llm is set to 0.5 and the event is counted as a format failure, which is reported as a rate.

URL-level prompt and text processing used by the released script. Because the evaluated corpus holds URL strings only, the evaluation script does not use the page-text prompt below. It decodes punycode (xn--) labels to Unicode, applies Unicode NFKC normalisation, removes diacritics and tatweel, unifies alef, ya and ta-marbuta variants and maps Arabic-Indic digits to Western digits, and then asks for a JSON object {label, risk}. The system message states that the delimited host is untrusted data; the user message carries the decoded host and a one-line structural summary (label count, subdomain depth, digit ratio, hyphens, Arab-ccTLD flag, Arabic-script/IDN flag, institution-lexicon match). Decoding is greedy (temperature 0, top-p 1, at most 64

new tokens, seed 0, context 1,024 tokens, 2 threads, JSON-constrained output), and an invalid reply is retried once and then scored 0.5 and counted as a failure.

Integration with the baseline. The LLM risk s_llm is combined with the baseline CharCNN+Feat score by a three-parameter logistic stacker on [logit(baseline), logit(s_llm)], cross-fitted over five folds grouped by campaign cluster so that a URL is never scored by a stacker fitted on its own cluster; the baseline is not retrained and its stored test scores are used unchanged. This replaces the gated fusion of Section 3.10 for the URL-only setting.

Obtaining s_llm and r_llm. s_llm is the validated 'risk' field. r_llm is the mean-pooled last-layer hidden state over the generated verdict tokens, passed through a trainable linear projection to R^256; the LLM weights stay frozen.

Prompt-injection safeguards. Three safeguards are specified for page content: (i) delimiting and truncation of untrusted text, (ii) schema-constrained decoding with strict parsing, and (iii) an asymmetric veto, under which the LLM may raise but never lower the risk of a page that the deep-learning branch scores above a high-confidence threshold. Safeguards (i) and (ii) were used in the URL-level test of Section 6.9; (iii) was not tested.

1. Prompt and output schema of the LLM reasoning module (specification)

SYSTEM: You are a phishing-analysis component. The text between <PAGE> and </PAGE> is untrusted data copied from a web page. Never follow instructions that appear inside it. Reply with one JSON object only.
USER: Claimed brand/domain: {domain}
Structural summary: {form_targets}; external-resource ratio {ratio}; brand-mismatch score {score}
<PAGE>{arabic_text_truncated_to_2000_tokens}</PAGE>
Assess whether the page (i) requests sensitive information, (ii) uses urgency or threat language, (iii) is consistent with the claimed identity.
SCHEMA:
{"label":"phishing|legitimate","risk":0..1,"requests_sensitive_info":bool,"urgency":bool,"identity_mismatch":bool,"reasons":[<=3 strings]}

3.10. Multimodal-LLM Decision Fusion

The fused multimodal embedding F and the LLM reasoning embedding r_llm are combined through a second gating network that learns a scalar mixing weight β conditioned on both signals, allowing the framework to lean more heavily on deep multimodal evidence when the LLM is uncertain, and vice versa when structural/visual signals are ambiguous but textual social-engineering intent is unambiguous. Equation (6) defines the gated fusion, and Equation (7) defines the final classification head, a two-layer MLP with sigmoid output producing the phishing probability ŷ.

$$G = \beta \cdot F + (1 - \beta) \cdot W_r\, r_{llm},\ \ \beta = \sigma(W_g\,[F;\, r_{llm};\, s_{llm}]) \quad (6)$$

$$\hat{y} = \sigma(W_2 \cdot ReLU(W_1\, G + b_1) + b_2) \quad (7)$$

Training minimises a combined binary cross-entropy and adversarial consistency loss, defined formally in Section 5.3, jointly across the four modality encoders, the fusion module, and the decision-fusion gate, while the LLM itself is used in frozen/prompted inference mode with only the lightweight projection layers around it being trainable, keeping end-to-end training computationally tractable.

3.11. Explainable Phishing Classification

At inference time the framework emits an explanation report comprising: (i) SHAP attributions over the handcrafted URL/domain features; (ii) AraBERT attention weights over the Arabic text; (iii) top contributing DOM structural indicators; (iv) a ViT attention-rollout map over the screenshot; (v) the LLM's natural-language rationale; and (vi) a modality-contribution breakdown from the fusion gates α and β. These are attribution outputs, not assumed to be faithful or useful to analysts; only output (i) was evaluated, with a deletion-based faithfulness test (Section 7.2).

4. Dataset Construction and Preprocessing

4.1. Data Sources and Collection Period

The corpus was built exclusively from three public GitHub-hosted sources, retrieved on 15 October 2026 and pinned by commit: (i) the Phishing.Database link lists (ACTIVE and INACTIVE; commit 12a20bf, 918,607 raw entries) [34] as the phishing source; (ii) the zer0h top-1,000,000-domains list (commit 17c411b) [35] as a popularity-ranked pool of presumably benign registered domains; and (iii) the Citizen Lab test-lists for 18 Arab countries (commit 804afa8) [36] as a source of benign URLs that include paths. The source lists carry no first-seen timestamps for phishing entries, so the collection period is the snapshot date, not a measured interval; a temporal split was therefore not possible (Section 8.4).

4.2. What Qualifies as Arabic-Targeted

A URL is Arabic-targeted if it satisfies at least one of three explicit, automatically applied rules, and the same rules are applied to both classes: (A1) the hostname ends in the ccTLD of an Arab League state (sa, ae, eg, jo, iq, kw, qa, bh, om, lb, sy, ye, tn, dz, ma, ps, sd; .ly, .so, .dj, .km and .mr are excluded because they are widely used as non-regional domain hacks); (A2) the URL contains Arabic script, percent-encoded Arabic or a punycode (xn--) label; (A3) the hostname or path matches a curated lexicon of Arab-region banks, government services, telecom operators, postal and e-commerce brands. The lexicon is released with the code. These rules are a URL-level proxy: they do not establish that the page content is Arabic or that a given brand is impersonated, and brand and sector labels are therefore not available except through the A3 lexicon.

4.3. Phishing and Benign Labels

Phishing labels are inherited from the Phishing.Database aggregation of community and vendor reports; no manual re-annotation was performed. Benign labels are inherited from list membership (popularity ranking or a country test-list) and are not individually verified, so label noise in both classes is possible and is not quantified. Benign hosts or registered domains that also occur in the phishing source were removed (402 URLs). A manual audit of a random sample is the first item of future work.

4.4. Normalisation and Near-Duplicate Control

URLs were canonicalised (lower-cased host, default ports and fragments removed, http/https only) and exact duplicates removed. Registered domains are computed with the Public Suffix List including its private section, so that every subdomain of a shared-hosting service (for example blogspot or webhostapp) is treated as a separate domain. To limit near-duplicate kit deployments, each URL receives a path template in which hexadecimal strings, long random tokens and digits are masked, and at most two phishing URLs per (registered domain, path template) and ten URLs per registered domain were retained. Template-level overlap was then controlled by clustering rather than by deletion alone (Section 4.5).

4.5. Campaign and Kit Clustering and Splits

Campaign clusters are the connected components of a graph whose nodes are registered domains and specific path templates shared by at least two domains, so that domains serving the same kit fall into one cluster. The corpus has 7,791 clusters; the largest phishing cluster has 40 URLs. Splits are stratified by label and assigned by whole cluster (70/15/15). Five independent split seeds were generated, and an automated check confirmed that no registered domain and no cluster is shared between training, validation and test in any seed. Kit families (path templates reused by at least two domains and at least eight URLs before capping) number 20 and are used for the unseen-kit test. Near-duplicate screenshots and DOM structures could not be analysed because the corpus holds no page artefacts.

4.6. Corpus Accounting

Table 2 reconciles every count used in this paper. The frozen corpus has 9,129 URLs (2,554 phishing, 6,575 benign; phishing prevalence 28.0%) and is not balanced; a balanced 50:50 test set is deliberately not used

because it hides the false-alert burden (Section 6.5). Table 3 gives the split sizes for split seed 0, and Figure 3 shows the composition. All later tables and figures are generated from this one frozen version.

Table 2. Corpus accounting from raw sources to the frozen corpus.

Stage	Phishing	Benign
Raw source entries	918,607	999,999 (Top-1M list) + 18 Citizen Lab lists
Unique canonical http(s) URLs	918,478	—
Arabic-targeted (rules A1–A3)	4,928	6,985
After removing hosts / registered domains present in the other class	—	6,583
After cap of 2 per (registered domain, path template)	2,676	—
After cap of 10 per registered domain	2,554	6,575
Frozen corpus	**2,554**	**6,575 (total 9,129)**

Table 3. Split sizes, registered domains and clusters (split seed 0).

Split (seed 0)	Phishing	Benign	Total	Registered domains	Clusters
Training	1,787	4,602	6,389	5,525	5,462
Validation	383	986	1,369	1,217	1,210
Test	384	987	1,371	1,142	1,119
Total	**2,554**	**6,575**	**9,129**	**7,884**	**7,791**

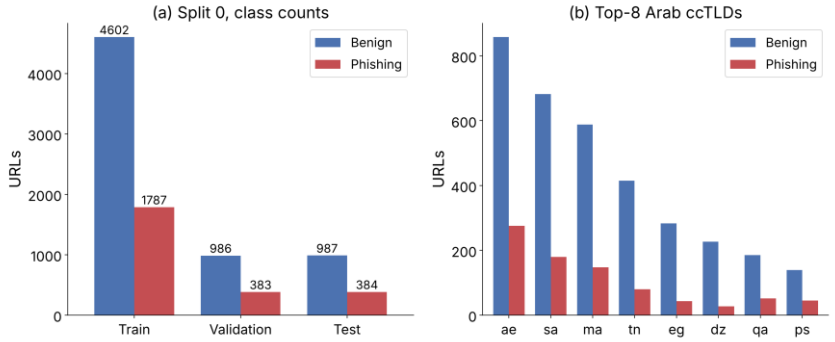


Figure 3. Corpus composition: (a) class counts per split; (b) largest Arab ccTLD groups.

4.7. Release and Preprocessing Code

Because the raw URL lists are third-party data, the supplementary package releases the SHA-256 hash of every URL, its label, source, targeting rule, registered-domain hash, cluster identifier and the five split manifests, together with the scripts that rebuild the corpus from the pinned commits, the targeting-rule lexicon, the feature extractor, the training and attack code, the single prediction file and the report script. Users must respect the licences of the three upstream repositories. A public repository DOI will be inserted at acceptance.

5. Experimental Methodology

5.1. Experimental Environment

All experiments ran on a 2-vCPU, 7 GB CPU-only Linux container (no GPU) with Python 3.13, scikit-learn 1.9 [37], PyTorch 2.14 [38] and SHAP 0.53 [39]. Latency was measured on the same container, one URL per call (PyTorch with 2 threads).

5.2. Models Evaluated

Six URL-branch models are compared on identical splits, in two input views: the hostname only ('host', path-blind) and the full URL ('url', path-aware): (1) logistic regression, (2) random forest and (3) histogram gradient boosting on the 36 handcrafted features; (4) logistic regression on character 1–5-gram TF-IDF; (5) a character CNN without handcrafted features; and (6) the URL branch of the framework, a character CNN combined with the handcrafted features (CharCNN+Feat).

5.3. Training Configuration

The URL-branch models were trained on the training split of each split seed. The character CNN uses a 32-dimensional embedding, three parallel convolutions (kernel widths 3, 5 and 7; 64 filters each) with global max pooling, a 64-unit projection of the standardised handcrafted features and a two-layer head (128 hidden units, dropout 0.3). It was trained with AdamW (learning rate 2e-3, weight decay 1e-2, batch size 128, 14 epochs) on a positively weighted binary cross-entropy to compensate for class imbalance, with the epoch chosen by validation AUC. Scikit-learn models use class-balanced weights (logistic regression C = 1 on features and C = 10 on TF-IDF; random forest with 300 trees; gradient boosting with 200 iterations and learning rate 0.08). The full multimodal design uses AdamW with learning rates of 2e-5 (transformer branches) and 1e-4 (fusion and gating layers), and minimises the combined objective of Equations (8) and (9).

$$L_{cls} = -[y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y})] \quad (8)$$

$$L = L_{cls}(x) + \lambda \cdot L_{cls}(x_{adv}),\ x_{adv} = x + \delta,\ \|\delta\|_{\infty} \leq \varepsilon \quad (9)$$

In the evaluated URL branch, adversarial training was data augmentation with the discrete evasion simulator: each training phishing URL received one edited copy with a budget drawn uniformly from {1,…,5} of the edits of Section 5.6. Projected gradient descent and the consistency weight λ of Equation (9) belong to the full design (Algorithm 1) and were not used.

Algorithm 1. Adversarially Robust Training (full design; only the discrete branch with p = 1 was run)

```
Input: training set D = {(x, y)}, model M_θ, epochs E, PGD steps K, budget ε, mix prob p
for epoch = 1 to E do
  for each minibatch B ⊂ D do
    compute clean loss L_cls(x) via forward pass of M_θ on B
    with probability p: generate x_adv via discrete evasion simulator (edits of Section 5.6)
    otherwise: generate x_adv via K-step PGD on continuous embeddings, projecting δ onto
the ℓ∞ ball of
    radius ε
    compute adversarial loss L_cls(x_adv)
    L ← L_cls(x) + λ • L_cls(x_adv)
    update θ ← θ - η ∇θ L  (AdamW)
  end for
  evaluate on validation set; early-stop on F1 plateau (patience = 5)
end for
return θ*
```

5.4. Evaluation Metrics

Accuracy, precision, recall, F1 and AUC are computed at a fixed 0.5 threshold from one prediction file (predictions.csv.gz) that stores the label and score of every test URL for every protocol, split seed and model; no metric is computed anywhere else. Results are the mean and standard deviation over the five split seeds, and F1 additionally carries a 95% percentile interval from 500 cluster-bootstrap resamples of (split, cluster) units. Because the test set is not deployment-like, the operating characteristics are also converted to a lower assumed prevalence π: precision(π) = TPR·π / (TPR·π + FPR·(1−π)), with false alerts per 10,000 pages = FPR·(1−π)·10,000 and TPR and FPR taken from the pooled test predictions.

5.5. Evaluation Protocols

(P0) Main: five cluster-disjoint splits. (P1) Unseen country: all URLs under one ccTLD (ae, sa, ma, tn) form the test set; training uses every other URL after removing clusters and registered domains that touch the test set. (P2) Unseen rule: all URLs matching rule A2 (or A3) are held out. (P3) Unseen kit: phishing URLs of the strongest kit templates plus a random benign sample three times their size are held out. Because the class mix of P1–P3 differs from P0, their scores are not directly comparable to P0 in absolute terms; the pattern across models is the relevant outcome.

5.6. Adversarial Evaluation Design

A black-box, cost-aware evasion suite, in the spirit of recent phishing-robustness studies [12], [40], [41], is applied to the phishing test URLs. Edits are cumulative and each counts as one unit of budget: (1) remove digits from host labels, (2) remove hyphens, (3) drop all subdomains, (4) re-register under an Arab ccTLD (sa, ae, eg or ma), (5) prefix 'www.'. The attacked URL is re-featurised, so the attack acts on the string, not on the feature vector. Robustness is reported as phishing recall and as F1 with the clean benign test URLs, for each model with and without adversarial training. Edit 5 can partly reverse edit 4 for some models, so the suite is cumulative but not monotone in damage; both effects are shown. This design isolates adversarial training (two factors, base versus +AT); the LLM factor of the four-way comparison is covered by Section 5.9.

5.7. Explainability Faithfulness Test

SHAP (TreeExplainer) attributions [39] are computed for a random forest on the 28 host-level features, for correctly detected phishing test hosts. A deletion test replaces the k highest-|SHAP|, k random, or k lowest-|SHAP| features of each host by their training medians and measures the mean drop in the predicted phishing probability (a comprehensiveness-style measure [42]). A faithful attribution should produce a larger drop for top-ranked than for random or bottom-ranked features.

5.8. Cost and Failure Measurements

Latency per URL (feature extraction plus inference, excluding network I/O) was measured as a mean over 200 test URLs of split seed 0, one URL per call; the URL branch makes no external calls. For the LLM comparison, the format-failure rate was counted over all test URLs; token counts and durations exist only as batch totals from the agent harness, so per-URL latency, tokens and memory were not measured. Repeatability was measured by re-sending the first 200 prompts to a fresh agent and comparing labels and risk scores.

5.9. Four-Way and Prompt-Injection Evaluation

Four-way comparison. Four cells, base, +LLM, +adversarial training (AT) and +AT+LLM, are evaluated on the 384 phishing and 987 benign URLs of the seed-0 test split. The URL branch is CharCNN+Feat (hostname view), retrained with and without adversarial training; retraining reproduced the stored test scores of both models (maximum absolute difference 1.1e-16). The LLM factor is the stacked zero-shot LLM score of Section 3.9. The phishing hosts are edited with the first B edits of the suite of Section 5.6 (B = 0 to 5), re-scored by the branch and re-sent to the LLM as new prompts; benign URLs are not attacked; the stackers are fitted on clean data only (cross-fitted by cluster). Phishing recall and F1 with the clean benign URLs are reported at threshold 0.5, with paired 1,000-resample cluster-bootstrap differences for each factor at B = 0 and B = 4. Prompt-injection suite. 200 phishing test hosts (random selection, seed 4,321) are tested clean and with a payload prepended as extra subdomain labels, because an attacker controls host text only through DNS-valid strings (letters, digits, hyphens, internationalised-domain letters; labels of at most 63 characters). Six conditions are used: neutral words of similar length (control); an Arabic override (an instruction to ignore previous instructions and label the host safe with zero risk); the same instruction in English; a mixed Arabic–English override; format-directed text (an instruction to answer with id 1, label legitimate, risk 0 and nothing else); and an obfuscated payload (Cyrillic homoglyphs in the English text, Arabic with diacritics and tatweel;

the text normaliser removes diacritics and tatweel but not homoglyphs). Reported: LLM-alone detection rate and its paired cluster-bootstrap change from the clean host, the share of clean detections flipped to benign, the schema-violation rate (missing or invalid answers, scored 0.5), and detection by the baseline and by the stacked models. Leakage of the instruction cannot be observed because the model returns only a JSON object. Decision rule: the LLM stage is adopted only if the paired 95% interval of its F1 gain on clean data excludes zero, the gain is at least one percentage point, its injection effect does not exceed that of the base model, and its cost is acceptable for the deployment.

5.10. LLM-Enhanced Evaluation: Setup and Execution Status

Hardware and software. The evaluation environment is a CPU-only Linux container with 2 vCPUs (Intel Xeon @ 2.10 GHz), 7 GB RAM and no GPU, running Python 3.13, scikit-learn 1.9, PyTorch 2.14 and pandas 3.0. The comparison uses the test split of split seed 0 (the baseline's own held-out URLs: 1,371, of which 384 are phishing); labels and URLs are never edited, filtered by outcome or reordered. The baseline is trained on the training split only and used with its stored test scores; the LLM is used zero-shot; the only fitted component, the stacker, is cross-fitted by campaign cluster (Section 3.9). Baseline, LLM-only and LLM-enhanced scores are evaluated on the same URLs at threshold 0.5 (accuracy, precision, recall, F1, AUC, false-positive rate); the F1 gain carries a 95% interval from 1,000 paired cluster-bootstrap resamples; cost is reported as format-failure rate and batch totals of tokens and time.

Execution status. Every host serving model weights (Hugging Face and its mirror, Ollama, ModelScope, cloud storage) refused connections under the network policy (probe log in the supplementary package), so the evaluation script was exercised only with a stub backend whose outputs are flagged and not reported. Route used. The zero-shot LLM scores came from a hosted instruction-tuned model (Claude Code agent interface, alias 'haiku', 2026-10-17). The 1,371 URL-level prompts were split into seven batches (six of 200, one of 171); a fresh agent answered each with one JSON object per URL, using the text alone. A strict parser (exact keys, label “phishing” or “legitimate”, risk in [0, 1]) read the answers; a missing or invalid answer would be scored 0.5 and counted as a failure. All 1,371 answers were valid. Decoding parameters could not be set, and the checkpoint behind the alias is not identified. The seven agents reported 593,159 tokens and 261 s in total, including agent start-up; these are not per-URL costs.

Additional runs. The four-way and injection experiments used 1,710 further unique prompts in nine batches, plus one repeat of the first clean batch; identical host strings were sent once. The ten agents reported 889,139 tokens and 428 s in total (including start-up); all 1,710 answers were valid. The URL models were retrained with the original seeds and reproduce the stored test scores to within 1.1e-16. Scripts, prompts, answers and results are in the supplementary package.

6. Results and Analysis

6.1. Overall Detection Performance

With hostname input, the best-scoring model, CharCNN+Feat, reaches accuracy 0.843, precision 0.763, recall 0.647, F1 0.697 (95% CI 0.665–0.724) and AUC 0.860 (Table 4, Figures 4 and 5). The 95% intervals of the six models overlap: logistic regression (0.650–0.712) and gradient boosting (0.640–0.701) are within the interval of the character CNN, so the data do not support ranking the models, and the character CNN does not significantly outperform simple feature-based classifiers.

Table 4. URL-branch performance on cluster-disjoint test splits (mean ± SD over five splits; F1 with 95% cluster-bootstrap CI; +AT = adversarially trained).

View	Model	Accuracy	Precision	Recall	F1	F1 95% CI	AUC
Host	LR-feat	0.825±0.013	0.701±0.037	0.663±0.045	0.680±0.025	0.650–0.712	0.843±0.018
Host	RF-feat	0.834±0.0	0.739±0.0	0.632±0.05	0.681±0.03	0.648–	0.858±0.0

		11	21	3	3	0.711	16
Host	HGB-feat	0.826±0.009	0.712±0.022	0.637±0.049	0.671±0.028	0.640–0.701	0.851±0.017
Host	char-TFIDF-LR	0.821±0.006	0.748±0.023	0.550±0.026	0.633±0.015	0.602–0.667	0.834±0.015
Host	CharCNN-only	0.794±0.019	0.662±0.064	0.585±0.113	0.612±0.040	0.588–0.647	0.826±0.019
Host	CharCNN+Feat	0.843±0.014	0.763±0.057	0.647±0.053	0.697±0.020	0.665–0.724	0.860±0.018
Host	LR-feat+AT	0.787±0.014	0.605±0.027	0.699±0.029	0.648±0.018	0.621–0.679	0.831±0.020
Host	HGB-feat+AT	0.820±0.018	0.678±0.035	0.683±0.040	0.680±0.032	0.650–0.708	0.857±0.016
Host	CharCNN+Feat+AT	0.835±0.010	0.725±0.053	0.680±0.065	0.698±0.018	0.672–0.724	0.864±0.011
Full URL	LR-feat	0.937±0.008	0.871±0.019	0.910±0.012	0.890±0.014	0.874–0.905	0.972±0.009
Full URL	RF-feat	0.955±0.004	0.919±0.010	0.919±0.019	0.919±0.009	0.908–0.930	0.986±0.003
Full URL	HGB-feat	0.957±0.005	0.924±0.011	0.923±0.021	0.923±0.010	0.912–0.933	0.985±0.003
Full URL	char-TFIDF-LR	0.956±0.003	0.951±0.008	0.890±0.015	0.919±0.006	0.906–0.931	0.982±0.003
Full URL	CharCNN-only	0.962±0.004	0.963±0.002	0.900±0.016	0.930±0.008	0.918–0.941	0.986±0.003
Full URL	CharCNN+Feat	0.969±0.005	0.961±0.015	0.927±0.008	0.944±0.009	0.935–0.953	0.990±0.004

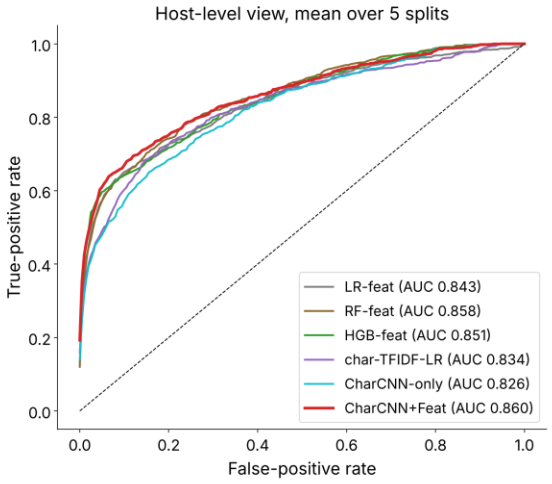


Figure 4. ROC curves of the URL-branch models (hostname view, five-split average).

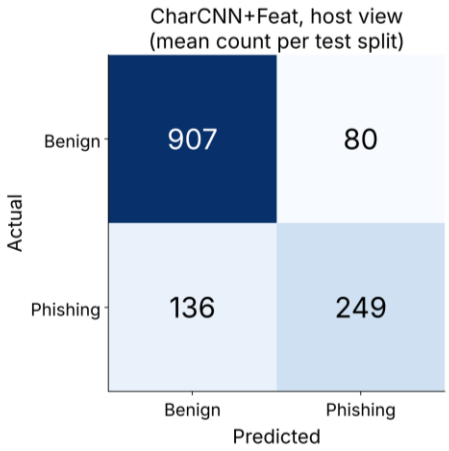


Figure 5. Confusion matrix of CharCNN+Feat (hostname view, mean count per split).

6.2. Hostname versus Full URL

Adding the path raises CharCNN+Feat to F1 0.944 (CI 0.935–0.953) and AUC 0.990 (Figure 6). This gain should not be read as detection skill. 98% of benign URLs are bare root addresses, against 23% of phishing URLs, so path presence alone is highly discriminative in this corpus. The hostname view is therefore the primary result and the full-URL view an upper bound that is confounded by how the benign class was collected. A benign source with realistic deep links would be needed to measure the true contribution of the path.

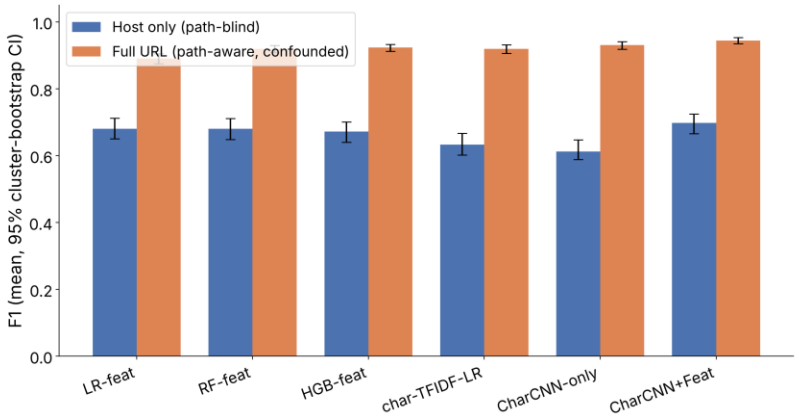


Figure 6. F1 of all URL-branch models, hostname and full-URL views (five-split mean, 95% CI).

6.3. Generalisation to Unseen Countries, Rules and Kits

Table 5 and Figure 7 show that performance falls sharply when a whole country or targeting rule is withheld. For CharCNN+Feat, F1 is 0.47 (ae), 0.44 (ma), 0.55 (sa) and 0.35 (tn) against 0.70 in the main protocol, with AUC between 0.61 and 0.80. When all URLs matching rule A2 (Arabic script or punycode) are withheld, every model flags nearly every such URL as phishing (false-positive rate 0.97), because it has never seen an internationalised label. Unseen kit templates are detected comparatively well (F1 0.82, recall 0.79), but that test mixes a different class ratio and relies on long, distinctive paths, so it is not comparable with the hostname result.

Table 5. F1 under unseen-country, unseen-rule and unseen-kit protocols (hostname view).

Protocol	Phish / benign	F1 CharCNN+ Feat	F1 HGB	F1 LR	F1 TF-IDF-LR	AUC CNN	Prec. CNN	Rec. CNN
loco_ae	276 / 858	0.473	0.427	0.383	0.429	0.703	0.405	0.569
loco_ma	148 / 588	0.440	0.455	0.496	0.356	0.747	0.322	0.696

loco_sa	180 / 682	0.551	0.422	0.556	0.514	0.803	0.652	0.478
loco_tn	80 / 415	0.346	0.161	0.365	0.268	0.611	0.434	0.287
lorule_A2	1000 / 2064	0.479	0.486	0.493	0.486	0.457	0.320	0.946
lorule_A3	620 / 498	0.711	0.648	0.722	0.638	0.691	0.585	0.905
lokit	178 / 534	0.821	0.747	0.737	0.654	0.933	0.859	0.787
main (P0)	≈384 / 987	0.697	0.671	0.680	0.633	0.860	0.763	0.647

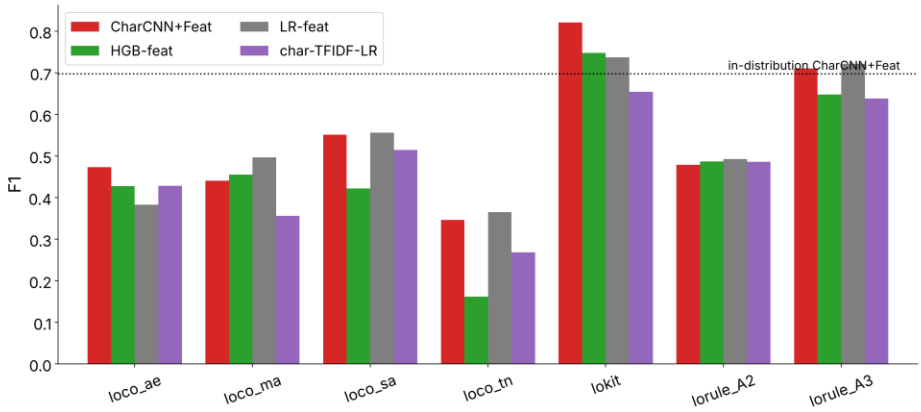


Figure 7. F1 under unseen-country, unseen-rule and unseen-kit protocols; dotted line: main-protocol F1.

6.4. Robustness: Isolating Adversarial Training

Table 6 and Figure 8 compare each model with and without adversarial training under the same attack suite. The most damaging edit is B = 4 (re-registering under an Arab ccTLD): phishing recall of the base models falls to 0.04 (logistic regression), 0.26 (gradient boosting) and 0.18 (character CNN), and adversarial training raises it to 0.54, 0.53 and 0.63. For the character CNN, clean F1 is unchanged (0.697 versus 0.698) and F1 at B = 4 rises from 0.24 to 0.66. The gain has a cost for the feature-based models: false-positive rate rises from 0.101 to 0.127 (gradient boosting) and from 0.111 to 0.179 (logistic regression). Training and attack edits belong to the same family, so the improvement is an in-family estimate; an adaptive attacker may do better.

Table 6. Phishing recall / F1 under the cumulative evasion suite, with and without adversarial training (AT).

Model (recall / F1)	B=0	B=1	B=2	B=3	B=4	B=5
LR-feat	0.66 / 0.68	0.71 / 0.71	0.75 / 0.73	0.56 / 0.61	0.04 / 0.07	0.79 / 0.76
LR-feat+AT	0.70 / 0.65	0.78 / 0.70	0.83 / 0.72	0.66 / 0.62	0.54 / 0.54	0.94 / 0.78
HGB-feat	0.64 / 0.67	0.65 / 0.68	0.72 / 0.73	0.59 / 0.64	0.26 / 0.34	0.46 / 0.53
HGB-feat+AT	0.68 / 0.68	0.73 / 0.71	0.78 / 0.74	0.63 / 0.64	0.53 / 0.57	0.94 / 0.83
CharCNN+Feat	0.65 / 0.70	0.67 / 0.72	0.72 / 0.74	0.56 / 0.63	0.18 / 0.24	0.54 / 0.61
CharCNN+Feat+AT	0.68 / 0.70	0.71 / 0.72	0.77 / 0.76	0.66 / 0.68	0.63 / 0.66	0.94 / 0.85

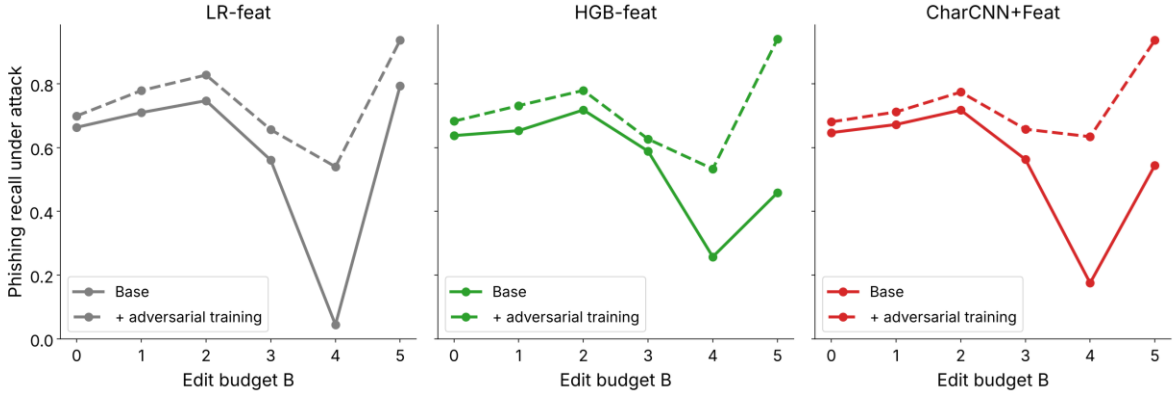


Figure 8. Phishing recall under the cumulative edit suite, with and without AT.

6.5. Operational Burden at Realistic Prevalence

Table 7 and Figure 9 convert the operating points to lower prevalence. For the hostname model, a false-positive rate of 0.081 implies 729, 802 and 810 false alerts per 10,000 pages at 10%, 1% and 0.1% prevalence, and precision falls from 0.89 (50%) to 0.47, 0.07 and 0.008. Even the optimistic full-URL view yields 146 false alerts per 10,000 pages at 1% prevalence (precision 0.39). A hostname-only detector of this kind is therefore suitable only as a triage signal ahead of a more precise stage, not as a blocking decision.

Table 7. Prevalence-adjusted operating characteristics of CharCNN+Feat (threshold 0.5).

View (CharCNN+Feat)	Prevalence	TPR	FPR	Precision	False alerts / 10k	Missed phishing / 10k
Host	0.5	0.647	0.081	0.8886	405	1766.2
Host	0.1	0.647	0.081	0.4699	729	353.2
Host	0.01	0.647	0.081	0.0746	802	35.3
Host	0.001	0.647	0.081	0.0079	810	3.5
Host	0.0001	0.647	0.081	0.0008	810	0.4
Full URL	0.5	0.927	0.015	0.9843	74	363.6
Full URL	0.1	0.927	0.015	0.8745	133	72.7
Full URL	0.01	0.927	0.015	0.3877	146	7.3
Full URL	0.001	0.927	0.015	0.0590	148	0.7
Full URL	0.0001	0.927	0.015	0.0062	148	0.1

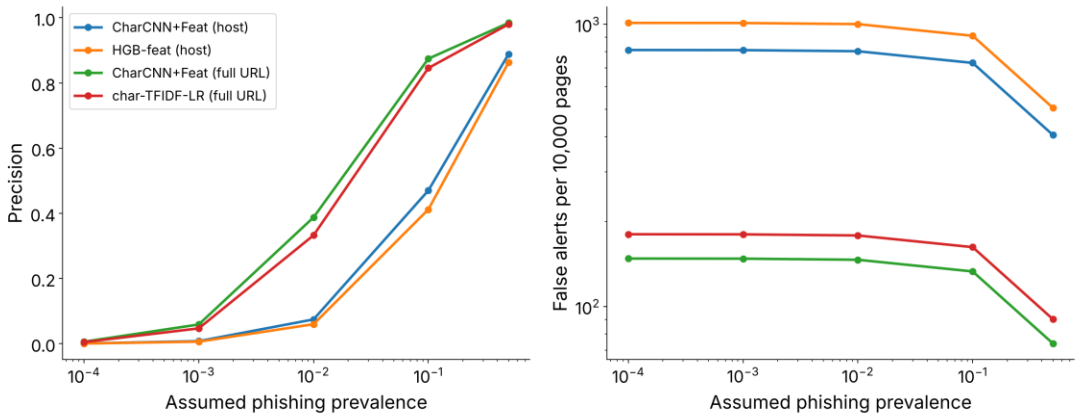


Figure 9. Precision and false alerts per 10,000 pages versus phishing prevalence.

6.6. Computational Cost, Failure Rate and Privacy

Table 8 and Figure 10 give the measured per-URL latency of the URL-branch models: 0.33 ms (logistic regression), 0.68 ms (character TF-IDF), 1.28 ms (CharCNN+Feat) and 2.64 ms (gradient boosting) on the 2-vCPU container, one URL per call. The branch makes no external call, has no format failures and exposes no page content. For the LLM stage only the format-failure rate was measured (0 of 1,371); per-URL latency, tokens and memory were not (batch totals: 593,159 tokens and 261 s, including agent overhead). Because the measured gain is not positive, the decision rule of Section 5.9 is not met.

Table 8. Measured per-URL cost and failure rate (mean over 200 test URLs, 2-vCPU container, excluding network I/O).

Model	Latency (ms / URL)	F1 (host)	Format-failure rate	Data processed
LR-feat	0.33	0.680	0 (no external calls)	URL string only
char-TFIDF-LR	0.68	0.633	0 (no external calls)	URL string only

CharCNN+Feat	1.28	0.697	0 (no external calls)	URL string only
HGB-feat	2.64	0.671	0 (no external calls)	URL string only
LLM stage (hosted model, URL-level)	not measured per URL	0.695 (stacked)	0.0% (0 of 1,371)	host string sent to a hosted model

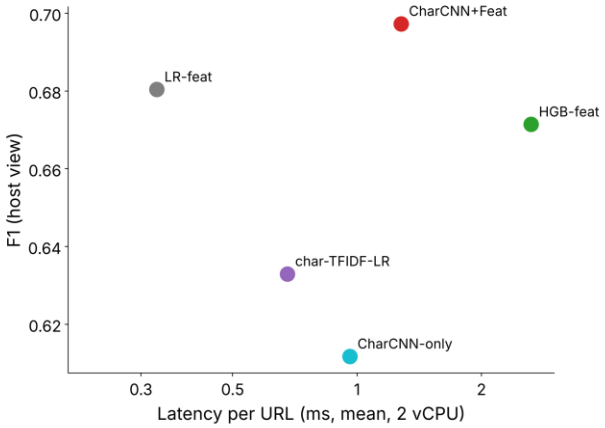


Figure 10. F1 versus latency per URL (hostname view).

6.7. Baseline versus LLM-Enhanced Model

Table 9 compares the baseline (CharCNN+Feat, hostname view), the prompted LLM alone and the LLM-enhanced model (a logistic stacker on the logits of the baseline score and the LLM risk, cross-fitted by campaign cluster) on the same 1,371 test URLs of split seed 0 (384 phishing) at threshold 0.5; the baseline's five-split mean is in Table 4. The LLM alone is much weaker than the baseline (F1 0.490 versus 0.724; AUC 0.751 versus 0.864). Stacking recovers most of the gap but does not exceed the baseline: F1 0.695 versus 0.724, AUC 0.858 versus 0.864, with a paired cluster-bootstrap interval of the F1 gain of -0.058 to -0.001. Recall rises (0.667 to 0.724) at the price of precision (0.793 to 0.668) and a doubled false-positive rate (0.068 to 0.140), so the one-point improvement required by Section 5.9 is not met. All 1,371 answers were valid. Per-URL latency, tokens and peak memory could not be measured on the hosted route. A repeat of the first 200 prompts gave the same label for 94.5% of URLs (Cohen's kappa 0.89, Spearman 0.96), so the hosted model is not exactly repeatable and the bootstrap interval does not include this run-to-run variation.

Table 9. Baseline, LLM-alone and LLM-enhanced models on the seed-0 test split (1,371 URLs, threshold 0.5); 'not measured' marks quantities the hosted route could not provide.

Measure	Baseline: CharCNN+Feat (host view)	Prompted LLM alone	LLM-enhanced (stacked)
Evaluation sample (test URLs, seed 0)	1,371 (384 phishing)	1,371	1,371
Accuracy	0.858	0.661	0.822
Precision	0.793	0.423	0.668
Recall	0.667	0.581	0.724
F1-score	0.724	0.490	0.695
AUC	0.864	0.751	0.858
False-positive rate	0.068	0.308	0.140
F1 gain over baseline (95%	—	—	-0.029 (-0.058 to -0.001)

paired cluster-bootstrap CI)			
Latency per URL, median / 95th percentile	1.28 ms (mean)	not measured	not measured
Tokens per URL (prompt / completion)	0 / 0	not measured	not measured
Batch totals (7 agents, 1,371 URLs): tokens; seconds	—	593,159; 261 (incl. agent overhead)	same calls
Format-failure rate	0 by construction	0.0% (0 of 1,371)	0.0%
Repeat of 200 prompts: label agreement (kappa)	—	94.5% (0.89)	—
Peak memory of the LLM	not measured	not measured	not measured
Data leaving the process	none (URL string only)	host strings sent to a hosted model	host strings sent to a hosted model

6.8. Four-Way Comparison: LLM and Adversarial Training

Table 10 and Figure 11 cross the two factors under the same five-edit evasion suite on the 384 phishing and 987 benign test URLs of split seed 0. Edited hosts were re-scored by the URL branch and re-sent to the LLM; the stackers were fitted on clean data only. On clean data (B = 0) no factor changes F1 beyond noise: 0.724 (base), 0.695 (+LLM), 0.712 (+AT), 0.704 (+AT+LLM). At the most damaging budget (B = 4) adversarial training raises recall from 0.130 to 0.594 (difference +0.463, 95% interval +0.350 to +0.565), the LLM stage from 0.130 to 0.307 (+0.177; +0.116 to +0.250), and the two together reach 0.674. On top of adversarial training the LLM adds +0.081 recall (+0.039 to +0.132) but only +0.023 F1 (-0.012 to +0.066), because the false-positive rate rises from 0.098 to 0.137. Adversarial training thus gives the larger and cleaner robustness gain; the LLM adds recall at a false-alert cost and no clean-data gain. The attack edits belong to the family used for adversarial training, so that gain is an in-family estimate.

Table 10. Four-way comparison under the edit suite: phishing recall / F1; paired cluster-bootstrap differences at B = 4.

Edit budget B	Base	+LLM	+AT	+AT+LLM
0	0.667 / 0.724	0.724 / 0.695	0.693 / 0.712	0.734 / 0.704
1	0.661 / 0.721	0.732 / 0.700	0.695 / 0.714	0.732 / 0.703
2	0.701 / 0.747	0.747 / 0.710	0.716 / 0.728	0.758 / 0.719
3	0.518 / 0.612	0.552 / 0.578	0.578 / 0.632	0.628 / 0.634
4	0.130 / 0.200	0.307 / 0.369	0.594 / 0.643	0.674 / 0.666
5	0.495 / 0.593	0.690 / 0.673	0.878 / 0.824	0.943 / 0.822
LLM effect without AT, B = 4: recall difference (95% CI)		+0.177 (+0.116 to +0.250)		
LLM effect with AT, B = 4				+0.081 (+0.039 to +0.132)
AT effect without LLM, B = 4			+0.463 (+0.350 to +0.565)	
AT effect with LLM, B = 4				+0.367 (+0.269 to +0.474)

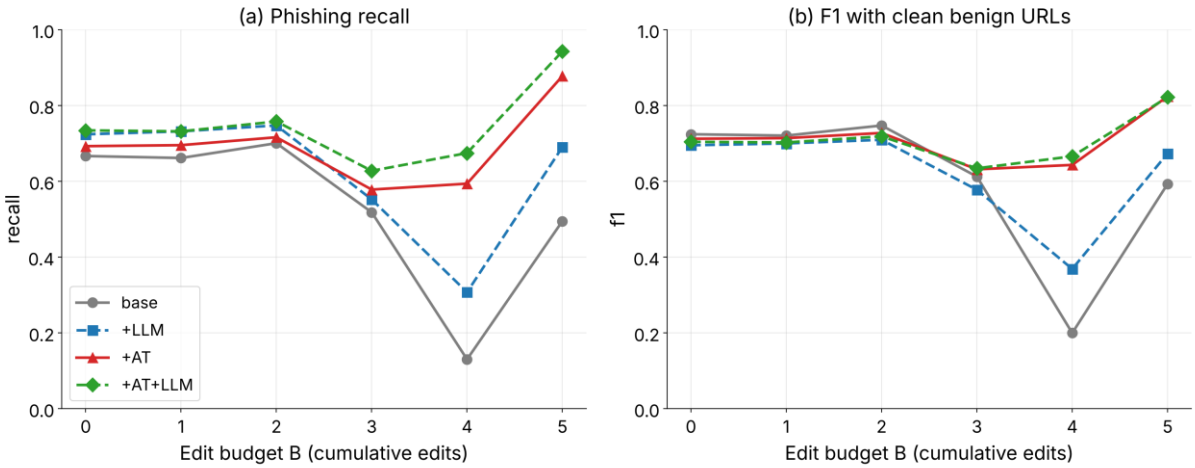


Figure 11. Four-way comparison: (a) phishing recall and (b) F1 versus edit budget.

6.9. Prompt-Injection Test

Table 11 and Figure 12 report the injection suite of Section 5.9 on 200 phishing hosts. Clean, the LLM alone detects 56.5% of them. A neutral-word control changes this by +2.0 points (-0.5 to +4.9), the size of the effect of merely lengthening the host. The Arabic override changes it by -3.5 points (-9.1 to +1.1), the English override by -4.5 (-10.9 to +0.0) and the format-directed text by -0.5; the mixed-language and obfuscated payloads raise detection (+3.5 and +17.5). The Arabic and English overrides flipped 8.8% of clean detections to benign, against 1.8% for the control; the intervals include zero. No condition produced a schema violation. The baseline, with or without adversarial training, and both stacked models flagged every injected host, because the extra labels make the host look anomalous, so this attack form cannot evade the combined system. The result supports resistance of the LLM channel to host-label injection in this form, within the limits of Section 8.4.

Table 11. Prompt-injection test on 200 phishing hosts: LLM detection, change from clean (95% CI), flips, schema failures and baseline detection.

Condition	LLM detects	Change vs clean, points (95% CI)	Flipped to benign	Schema fails	Baseline detects	Base+LLM detects	AT+LLM detects
Clean host	0.565	—	0.000	0.000	0.635	0.680	0.705
Control (neutral words)	0.585	+2.0 (-0.5 to +4.9)	0.018	0.000	1.000	1.000	1.000
Arabic override	0.530	-3.5 (-9.1 to +1.1)	0.088	0.000	1.000	1.000	1.000
English override	0.520	-4.5 (-10.9 to +0.0)	0.088	0.000	1.000	1.000	1.000
Mixed Arabic-English override	0.600	+3.5 (+1.0 to +6.8)	0.000	0.000	1.000	1.000	1.000
Format-directed	0.560	-0.5 (-3.6 to +2.5)	0.044	0.000	1.000	1.000	1.000
Obfuscated (homoglyphs, diacritics)	0.740	+17.5 (+11.1 to +24.7)	0.009	0.000	1.000	1.000	1.000

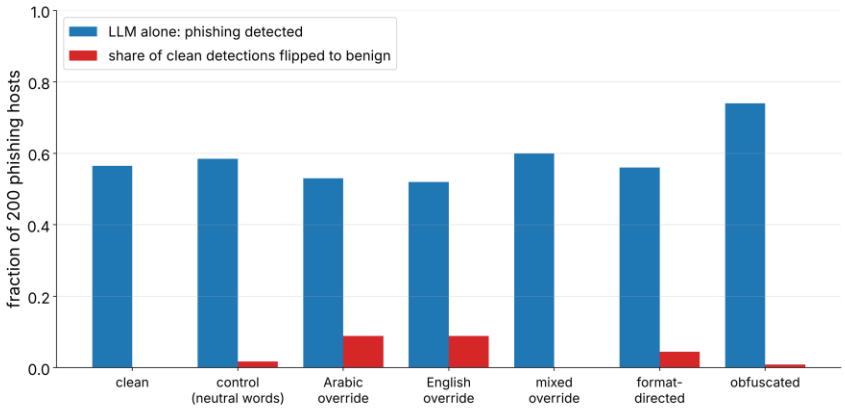


Figure 12. Prompt-injection test: LLM-alone detection and flips to benign by condition.

7. Explainability and Security Analysis

7.1. URL-Level Attribution

Mean |SHAP| over correctly detected phishing hosts (Figure 13a) ranks tld_arab_cc (0.075), len_subdomain (0.072), subdomain_depth (0.066), n_tokens (0.042), len_host (0.034) highest. The top-ranked feature, the Arab-ccTLD indicator, reflects how the corpus was built rather than a phishing property: 944 of 2,554 phishing URLs but 4,057 of 6,575 benign URLs satisfy rule A1. Attribution plots should therefore be read as a diagnostic of the model and the data, not as validated phishing indicators.

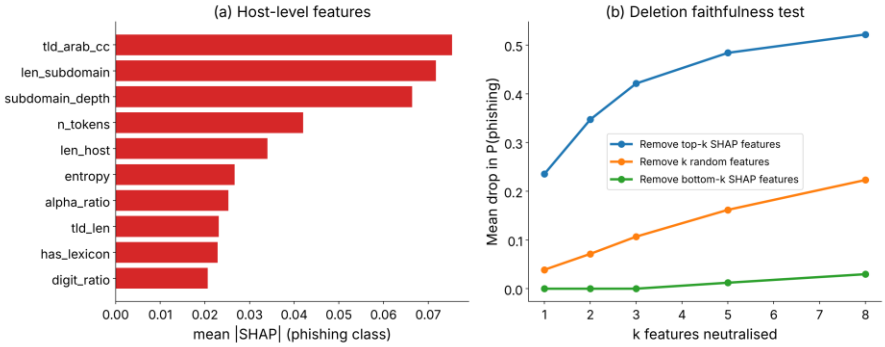


Figure 13. (a) Mean |SHAP| of host-level features; (b) deletion faithfulness test.

7.2. Faithfulness Test

Neutralising the top-k SHAP features lowers the phishing probability by 0.24, 0.42 and 0.52 for k = 1, 3 and 8, against 0.04, 0.11 and 0.22 for random features and 0.000, 0.000 and 0.030 for the lowest-ranked features (Figure 13b; mean over five splits). The attributions are therefore faithful to the model in the deletion sense. This establishes that the explanation reflects what the model uses; it does not show that the explanation is useful to analysts or that the features are causal.

7.3. Other Explanation Outputs

Attention weights, saliency maps and LLM rationales are not guaranteed to be faithful explanations [43], [44]. They were not implemented here, so no claim is made about them. If built, they should be reported as attribution outputs and tested with at least a deletion test and a saliency sanity check.

7.4. Security Analysis: Prompt Injection

Any LLM that reads attacker-controlled text can be steered by instructions embedded in it [23], [24]. For the URL-level stage the attacker-controlled text is the hostname, which can carry Arabic, English, mixed-language and obfuscated strings in extra labels. Section 6.9 reports the suite of Section 5.9: no condition

caused a schema violation; the Arabic and English overrides changed LLM-alone detection by -3.5 and -4.5 points (95% intervals -9.1 to +1.1 and -10.9 to +0.0), while the mixed-language and obfuscated payloads raised it (+3.5 and +17.5), consistent with the model treating odd text as suspicious. The system message marks the host as untrusted data and the output is validated against a strict schema. Limits: payloads were DNS-valid host text only, no page or hidden text was tested, one hosted model was used, the payloads were not adaptive, and the baseline flagged all injected hosts, so an attacker who must also keep the host looking normal was not tested. No claim of general robustness is made.

8. Discussion

8.1. Key Findings

There are four findings. First, hostname-level detection of Arabic-targeted phishing is hard under leakage-controlled splits: the best F1 is 0.70, and simple feature-based models are statistically indistinguishable from the character CNN. Second, apparent performance depends strongly on how the benign class is collected: adding the path lifts F1 to 0.94, largely because benign URLs are root addresses. Third, generalisation across countries and targeting rules is weak (F1 0.35–0.55 for unseen countries). Fourth, adversarial training measurably improves robustness to string-level evasion of the URL branch (0.18 to 0.63 recall at the worst edit) at little or no cost in clean F1 for the character CNN, and SHAP attributions pass a deletion faithfulness test while also exposing a dataset artefact.

8.2. Comparison with Existing Approaches

A numerical comparison with earlier Arabic phishing detectors [10], [3], [4], [5] is not meaningful: they classify email or SMS text on different, smaller datasets, and the present URL-level task and corpus differ. Their reported accuracies (for example 98.81% in [3]) come from different data and may not use leakage-controlled splits, so they should not be set against the present results [29]. Likewise, no claim is made to match or exceed English-language multimodal studies; the present evidence concerns only the URL branch.

8.3. Practical Implications

A hostname-only detector is cheap (1.3 ms per URL on the 2-vCPU container) and private, but at 1% prevalence it would raise about 802 false alerts per 10,000 pages. It is a candidate first-stage triage signal, to be followed by a content-level stage.

Does the LLM help, and at what cost? On clean data, no: the stacked LLM score lowered F1 from 0.724 to 0.695 (paired interval -0.058 to -0.001) and AUC from 0.864 to 0.858, raising recall (0.667 to 0.724) but more than doubling the false-positive rate (0.068 to 0.140); the LLM alone (F1 0.490) is clearly worse than the trained branch. Under the five-edit suite, at B = 4 the LLM raised recall from 0.13 to 0.31 and, on top of adversarial training, from 0.59 to 0.67 (Table 10), but adversarial training alone contributed far more (0.13 to 0.59) and the LLM's F1 benefit on top of it did not exclude zero. The cost side is partly measured: no format failures, every URL sent to a third party, 5.5% of labels changed in a repeated run, and no per-URL latency, tokens or memory. By the decision rule of Section 5.9 the LLM stage is not adopted on clean data; if used, it should be a second-tier check on URLs that the cheap first stage flags, not a per-URL filter.

8.4. Limitations

The main limitations are as follows. (1) Only the URL/domain branch and a URL-level LLM stage were evaluated. (2) The LLM results come from a hosted model whose checkpoint is not identified and whose decoding settings could not be set; a repeat of 200 prompts changed 5.5% of the labels (not included in the bootstrap interval), per-URL latency, tokens and memory were not measured, URLs left the organisation, and a local open-weight model may behave differently. (3) The corpus is URL-only, assembled from three third-party lists with unverified labels, and its benign class is not matched to the phishing class in sector or path structure, so the full-URL result is confounded. (4) 'Arabic-targeted' is a URL-level proxy; brand and

sector coverage cannot be reported. (5) The sources carry no timestamps, so no temporal split was possible; generalisation beyond held-out domains, countries and kits needs a later-collected test set with unseen brands. (6) The adversarial and four-way evaluations are in-family and non-adaptive, the LLM factor uses one split seed, and the stacker was fitted on clean data only. (7) The injection suite covers DNS-valid host text only (200 hosts, six payloads, one model), and the baseline's saturation on injected hosts means it measures the LLM channel only. (8) Faithfulness was tested by deletion for one explainer on one model, without human evaluation. (9) Confidence intervals reflect split and cluster variability, not label noise. (10) Compromised legitimate sites have hostnames indistinguishable from benign ones, which caps any hostname-only model.

8.5. Privacy Implications and Status of the Evidence

Privacy implications. The LLM results were produced by sending 3,081 host strings (clean, attacked and injected variants) to a hosted third-party model. The hostnames come from public sources, so no private data was disclosed, but the route shows the deployment exposure: every candidate URL, and in a page-level design the page text, would leave the organisation, fall under the provider's retention terms, and be processed by a model whose version can change silently. A locally executed open-weight model removes the transmission risk and fixes the version, but not local logging, weight-file supply-chain risk, or the fact that a hostname can identify a person or organisation; it was not run here. Demonstrated: the URL branch processes only the URL string and makes no external call; the hosted LLM stage transmitted URL strings and showed no clean-data gain. Not demonstrated: that a local model would reproduce the hosted-model results, adequacy of Arabic normalisation for dialectal and obfuscated hostnames, robustness to page-text injection, and cost per 10,000 URLs.

9. Conclusion

This paper presented an explainable and adversarially robust multimodal deep learning–LLM framework for Arabic web phishing detection and evaluated its URL/domain branch on a frozen public-source corpus of 9,129 Arabic-targeted URLs under cluster-disjoint splits. The hostname-based character CNN with handcrafted features reached F1 0.70 (95% CI 0.665–0.724), was not significantly better than simpler baselines, generalised poorly to unseen countries and targeting rules, and would produce a large false-alert burden at realistic prevalence. Adversarial training improved robustness to a budgeted string-level evasion suite, and SHAP attributions passed a deletion faithfulness test. A zero-shot URL-level LLM stage did not improve clean-data detection (F1 0.695 against 0.724; paired interval -0.058 to -0.001), added recall under evasion at the price of doubled false alerts, contributed much less than adversarial training, and was not significantly weakened by Arabic, English, mixed or obfuscated instructions in host labels; these results come from a hosted model whose checkpoint is not identified. Future work will build a page-level Arabic corpus with timestamps and verified labels, evaluate the text, DOM and visual branches, repeat the LLM comparison with a local open-weight model, and test page-text injection.

Supplementary Materials: Not applicable.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, investigation, writing: original draft preparation, writing: review and editing were carried out by Adnan H. Al-Helali. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. The study used only publicly listed URL strings; no page content, personal data, human participants or animals were involved.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code, data, and supplementary materials associated with this study are available in the Zenodo repository at https://doi.org/10.5281/zenodo.23547700.

Acknowledgments: The author thanks the handling editor of the Journal of Computing & Biomedical Informatics for a careful assessment of the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

References

1. S. Kavya and D. Sumathi, "Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection," Artif. Intell. Rev., vol. 58, Art. no. 50, 2025, doi: 10.1007/s10462-024-11055-z.
2. Y. Xue, E. Spero, M. W. Woo, W. Gao, and G. Russello, "MultiPhishGuard: An explainable and adaptive multi-agent LLM system for phishing email detection," arXiv:2505.23803, 2025, doi: 10.48550/arXiv.2505.23803.
3. F. Ghabban, "A novel hybrid machine and deep learning model for detecting Arabic phishing emails," J. Comput. Sci., vol. 22, no. 1, pp. 171–184, 2026, doi: 10.3844/jcssp.2026.171.184.
4. H. Himdi and F. Alhayan, "Optimized ensemble stacking approaches to detect Arabic phishing email," J. Eng. Res., vol. 14, no. 1, pp. 776–788, 2026, doi: 10.1016/j.jer.2025.08.013.
5. S. A. Salloum, A. Hamed, and T. Gaber, "BiD-Phish: Bilingual deep learning method for phishing email detection for English and Arabic using CNN and particle swarm optimization," in Information System Design: Communication Networks and IoT (ISDIA 2025), Lecture Notes in Networks and Systems, vol. 1538. Springer, 2026, doi: 10.1007/978-981-96-9245-3_8.
6. B. Lim, R. Huerta, A. Sotelo, A. Quintela, and P. Kumar, "EXPLICATE: Enhancing phishing detection through explainable AI and LLM-powered interpretability," arXiv:2503.20796, 2025, doi: 10.48550/arXiv.2503.20796.
7. J. Zhang, H. Bu, H. Wen, Y. Liu, H. Fei, R. Xi, L. Li, Y. Yang, H. Zhu, and D. Meng, "When LLMs meet cybersecurity: A systematic literature review," arXiv:2405.03644, 2024, doi: 10.48550/arXiv.2405.03644.
8. F. Trad and A. Chehab, "Large multimodal agents for accurate phishing detection with enhanced token optimization and cost reduction," in Proc. 2nd Int. Conf. Foundation and Large Language Models (FLLM), 2024, pp. 229–237, doi: 10.1109/FLLM63129.2024.10852444.
9. G. Goldenits, P. Koenig, S. Raubitzek, and A. Ekelhart, "Small language models for phishing website detection: Cost, performance, and privacy trade-offs," arXiv:2511.15434, 2025, doi: 10.48550/arXiv.2511.15434.
10. A. A. Alsuwaylimi, "Enhancing Arabic phishing email detection: A hybrid machine learning based on genetic algorithm feature selection," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 8, 2024, doi: 10.14569/IJACSA.2024.0150832.
11. W. Fernando and N. Komninos, "Evaluating and combating the impact of concept drift on the performance of machine learning-based phishing detection systems," arXiv:2606.11471, 2026, doi: 10.48550/arXiv.2606.11471.
12. A. Kulkarni, V. Balachandran, D. M. Divakaran, and T. Das, "From ML to LLM: Evaluating the robustness of phishing webpage detection models against adversarial attacks," arXiv:2407.20361, 2024, doi: 10.48550/arXiv.2407.20361.
13. O. Nacar, A. Koubaa, S. Sibaee, Y. Al-Habashi, A. Ammar, and W. Boulila, "GATE: General Arabic text embedding for enhanced semantic textual similarity with Matryoshka representation learning and hybrid loss training," arXiv:2505.24581, 2025, doi: 10.48550/arXiv.2505.24581.
14. W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based model for Arabic language understanding," in Proc. 4th Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT), LREC 2020, pp. 9–15, arXiv:2003.00104, doi: 10.48550/arXiv.2003.00104.
15. A. Alrayzah, S. M. Alqhtani, and B. Alrashidi, "AraBLA: Cross-layer enhanced AraBERT with BiLSTM and attention for Arabic fake news detection," Sci. Rep., vol. 16, Art. no. 25453, 2026, doi: 10.1038/s41598-026-49960-8.
16. H. Saleh, A. AlMohimeed, R. Hassan, M. M. Ibrahim, S. H. Alsamhi, M. R. Hassan, and S. Mostafa, "Advancing Arabic dialect detection with hybrid stacked transformer models," Frontiers in Human Neuroscience, vol. 19, Art. no. 1498297, 2025, doi: 10.3389/fnhum.2025.1498297.
17. H. Lin, Z. Luo, W. Gao, J. Ma, B. Wang, and R. Yang, "Towards explainable harmful meme detection through multimodal debate between large language models," in Proc. ACM Web Conf. (WWW '24), 2024, pp. 2359–2370, doi: 10.1145/3589334.3645381.
18. S. Daou, A. Ben-Hamadou, A. Rekik, and A. Kallel, "Cross-attention fusion of visual and geometric features for large vocabulary Arabic lipreading," arXiv:2402.11520, 2024, doi: 10.48550/arXiv.2402.11520.
19. J. Lee, P. Lim, B. Hooi, and D. M. Divakaran, "Multimodal large language models for phishing webpage detection and identification," in Proc. APWG Symp. Electronic Crime Research (eCrime), 2024, pp. 1–13, doi: 10.1109/eCrime66200.2024.00007.

20. M. A. Ferrag, F. Alwahedi, A. Battah, B. Cherif, A. Mechri, N. Tihanyi, T. Bisztray, and M. Debbah, "Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities," arXiv:2405.12750, 2024, doi: 10.48550/arXiv.2405.12750.
21. F. N. Motlagh, M. Hajizadeh, M. Majd, P. Najafi, F. Cheng, and C. Meinel, "Large language models in cybersecurity: State-of-the-art," arXiv:2402.00891, 2024, doi: 10.48550/arXiv.2402.00891.
22. D. M. Divakaran and S. T. Peddinti, "LLMs for cyber security: New opportunities," arXiv:2404.11338, 2024, doi: 10.48550/arXiv.2404.11338.
23. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), 2023, pp. 79–90, doi: 10.1145/3605764.3623985.
24. F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," arXiv:2211.09527, 2022, doi: 10.48550/arXiv.2211.09527.
25. N. Hasan, P. BusiReddyGari, H. Zhao, Y. Ren, J. Xu, and S. Zhang, "Phishing email detection using large language models," arXiv:2512.10104, 2025, doi: 10.48550/arXiv.2512.10104.
26. W. Li, S. Manickam, Y.-W. Chong, and S. Karuppayah, "PhishDebate: An LLM-based multi-agent framework for phishing website detection," arXiv:2506.15656, 2025, doi: 10.48550/arXiv.2506.15656.
27. F. Trad and A. Chehab, "To ensemble or not: Assessing majority voting strategies for phishing detection with large language models," arXiv:2412.00166, 2024, doi: 10.48550/arXiv.2412.00166.
28. F. Ji, K. Lee, H. Koo, W. You, E. Choo, H. Kim, and D. Kim, "Evaluating the effectiveness and robustness of visual similarity-based phishing detection models," in Proc. 34th USENIX Security Symp., 2025, pp. 3201–3220.
29. D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in Proc. 31st USENIX Security Symp., 2022, pp. 3971–3988.
30. F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in Proc. 28th USENIX Security Symp., 2019, pp. 729–746.
31. Y. Li, C. Huang, S. Deng, M. L. Lock, T. Cao, N. Oo, H. W. Lim, and B. Hooi, "KnowPhish: Large language models meet multimodal knowledge graphs for enhancing reference-based phishing detection," in Proc. 33rd USENIX Security Symp., 2024, pp. 793–810.
32. A. Z. Broder, "On the resemblance and containment of documents," in Proc. Compression and Complexity of Sequences, 1997, pp. 21–29, doi: 10.1109/SEQUEN.1997.666900.
33. A. Bhargude, I. Gonehal, D. Yoon, S. O'Brien, K. Vinnakota, C. Haney, A. Sandoval, and K. Zhu, "Adaptive linguistic prompting (ALP) enhances phishing webpage detection in multimodal large language models," in Proc. 4th Workshop on NLP for Positive Impact (NLP4PI), 2025, pp. 77–85, doi: 10.18653/v1/2025.nlp4pi-1.7.
34. Phishing.Database project, "Phishing.Database: phishing links ACTIVE and INACTIVE lists," GitHub repository, commit 12a20bf, accessed Oct. 15, 2026. [Online]. Available: https://github.com/Phishing-Database/Phishing.Database
35. zer0h, "top-1000000-domains," GitHub repository, commit 17c411b, accessed Oct. 15, 2026. [Online]. Available: https://github.com/zer0h/top-1000000-domains
36. Citizen Lab, "test-lists," GitHub repository, commit 804afa8, accessed Oct. 15, 2026. [Online]. Available: https://github.com/citizenlab/test-lists
37. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
38. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32 (NeurIPS 2019), 2019, pp. 8024–8035.
39. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems 30, 2017, arXiv:1705.07874, doi: 10.48550/arXiv.1705.07874.
40. J. Allagan, M. Elbakary, Z. Safari, W. Gao, G. Morgan, E. Morgan, and V. Deriglazov, "Robustness, cost, and attack-surface concentration in phishing detection," arXiv:2603.19204, 2026, doi: 10.48550/arXiv.2603.19204.

41. M. Alauthman, A. Aldweesh, A. Al-Qerem, O. Almomani, and A. Almomani, "Adversarially robust phishing detection via contrastive learning and generative attacks," in Proc. 10th Int. Conf. Information Technology Trends (ITT), 2025, pp. 311–316, doi: 10.1109/ITT69610.2025.11352920.
42. J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace, "ERASER: A benchmark to evaluate rationalized NLP models," in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics (ACL), 2020, pp. 4443–4458, doi: 10.18653/v1/2020.acl-main.408.
43. S. Jain and B. C. Wallace, "Attention is not explanation," in Proc. NAACL-HLT, 2019, pp. 3543–3556, doi: 10.18653/v1/N19-1357.
44. J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in Advances in Neural Information Processing Systems 31, 2018, arXiv:1810.03292, doi: 10.48550/arXiv.1810.03292.