

Research Article

A Hybrid Framework Combining 1D-CNN Feature Learning, Handcrafted Features, and Bayesian Updating for Phishing URL Detection

Hamza Zidoum^{1, *}, Thuwaiba S. Al Busaidi^{1,}¹ Sultan Qaboos University, Department of Computer Science, Al Khod 123, Oman.**ARTICLEINFO**

Article History

Received 02 May 2026

Revised: 18 Jun 2026

Accepted 17 Jul 2026

Published 04 Aug 2026

Keywords

Phishing URL detection,

1D convolutional neural network,

Bayesian inference,

Hybrid deep learning,

Cybersecurity.

**ABSTRACT**

Phishing remains a widespread cyber threat, deceiving users into revealing sensitive information. Traditional detection draws on URL-based features and non-URL-based features obtained through page parsing or third-party queries, yet adapting to evolving tactics remains challenging. This study proposes a four-phase framework and investigates whether Bayesian inference can revise classification decisions using external evidence proxies. Classical classifiers are trained on 52 handcrafted URL-based features; a one-dimensional convolutional neural network (1D-CNN) is trained directly on raw URL strings; the two are fused; and Bayes' theorem is applied, treating the 1D-CNN output as the prior and updating it with four non-URL-based evidence items: Google indexing, DNS record, WHOIS registration and suspicious top-level domain. On a balanced benchmark of 11,430 URLs, the 1D-CNN with subword tokenisation and particle swarm optimised hyperparameters reaches 92.25% accuracy, exceeding the strongest classical baseline (XGBoost, 91.21%), and hybrid fusion attains 92.53%. The Bayesian layer raises recall from 90.57% to 92.01%, reducing false negatives from 105 to 89. Because false negatives dominate the security cost of phishing, this trade-off favours real-time deployment. Likelihoods were estimated from the benchmark distribution rather than live queries, so the Bayesian component is a proof of concept.

1. INTRODUCTION

Phishing is one of the most common and dangerous forms of cybercrime. Attackers trick users into revealing sensitive information such as usernames, passwords and financial details, usually through fake emails or websites that closely resemble legitimate ones. The Anti-Phishing Working Group (APWG) reported 932,923 phishing attacks in the third quarter of 2024, an increase from 877,536 in the second quarter [1]. Phishing therefore continues to pose a serious risk to individuals and businesses alike, causing financial losses, reputational damage and compromised security. A range of countermeasures has been proposed, including list-based methods such as whitelists and blacklists, visual content similarity approaches, heuristic techniques, and machine learning (ML) and deep learning (DL) methods.

Current anti-phishing techniques rely largely on manual feature engineering and static rules, which struggle to keep pace with the constantly evolving tactics of attackers. The diversity of phishing techniques makes it difficult for traditional methods to detect new threats, resulting in more false positives and slower detection, and cybercriminals frequently adjust their strategies specifically to bypass established detection systems [2]. Earlier approaches typically relied on hand-crafted features, which may not fully capture the dynamic nature of phishing attacks [3]. As phishing has become more sophisticated, relying solely on fixed rules and manual feature extraction is no longer sufficient.

Machine learning for phishing detection depends on feature engineering, in which manually selected characteristics of the raw data, such as URL length or domain structure, are supplied to classifiers such as decision trees or support vector machines. This requires considerable expertise, is time-consuming, and adapts poorly to evolving tactics because the feature set is predefined. Deep learning, and convolutional neural networks (CNNs) in particular, instead extract features automatically from raw data, allowing complex and subtle patterns in phishing URLs to be detected without manual intervention.

Combining the two paradigms is attractive: classical ML handles structured data with explicitly engineered features well, while CNNs learn directly from raw URL strings. Neither, however, addresses adaptability at the moment of access. By using real-time information about the status of a URL as it is accessed, a detector can obtain additional evidence about whether that URL is legitimate or malicious [4]. Bayes' rule provides a principled mechanism for revising a belief in the light of such evidence [5], and is therefore well suited to updating a detection decision as external signals arrive.

*Corresponding author email: zidoum@squ.edu.omDOI: <https://doi.org/10.70470/SHIFRA/2026/011>

Because the evidence used in the Bayesian layer was drawn from features already present in the benchmark dataset rather than retrieved by live queries, and because those features are treated as independent and static, the Bayesian component is reported here as a proof of concept rather than as a deployed system. This scoping is discussed in Section 6.

This work addresses two research questions. First, how can features extracted from the URL string, when combined with features learned by a one-dimensional convolutional neural network (1D-CNN), improve the prediction of phishing websites? Second, can Bayesian inference, applied as a post-hoc update using static external feature proxies, demonstrate potential for improving recall in a simulated detection environment?

The contributions of this study are as follows.

A hybrid phishing detection model that fuses deep features learned by a 1D-CNN from raw URL strings with 52 handcrafted URL-based features, evaluated against six classical classifiers on a balanced benchmark of 11,430 URLs.

A systematic comparison of character-level and subword tokenisation for 1D-CNN phishing URL classification, with hyperparameters tuned by particle swarm optimisation.

A Bayesian inference layer that treats the 1D-CNN output probability as the prior and updates it with four non-URL-based items of evidence, evaluated with recall as the primary metric because false negatives dominate the security cost.

The remainder of the paper is organised as follows. Section 2 introduces the background concepts. Section 3 reviews related work. Section 4 presents the proposed four-phase methodology. Section 5 reports the experimental results. Section 6 discusses the findings and limitations, and Section 7 concludes.

2. BACKGROUND

2.1 Phishing

The first appearance of the word "phishing" in printed media is attributed to an article by Ed Stansel in the Florida Times Union, published on 16 March 1997 [6]. Phishing techniques changed little in their earliest years, but in 2001 attackers shifted their focus to online payment systems; by late 2003 they were registering large numbers of domains closely resembling legitimate sites such as eBay and PayPal, and using email worms to direct customers to fraudulent pages that requested credit card and other personal details [7].

The APWG, established in 2003, defines phishing as a criminal method that uses both social engineering and technological tricks to gain credentials from victims [7]. Phishing websites have two primary characteristics: they closely resemble a legitimate website, and they contain at least one input form through which users can enter their credentials [8]. A phishing campaign can therefore be decomposed into three steps: creating the phishing website, distributing its link through a communication channel, and harvesting the victim's credentials.

2.2 Uniform Resource Locators

Uniform Resource Locators (URLs) were introduced in RFC 1738 in 1994. A URL is a particular kind of Uniform Resource Identifier that not only identifies a resource but also describes how to locate and retrieve it [9]. Because a URL must be processed before any page content is available, it is the natural object of analysis for detection systems that need to act before a user interacts with a page.

A URL is composed of several parts, each of which carries potential signals of illegitimacy, as illustrated in Fig. 1: the protocol, subdomain, main domain, top-level domain (TLD), port, path, query string and fragment [10]. Suspicious domain names that are slightly misspelled or built from random character strings are common in phishing attempts; so are less common domain extensions such as .xyz or .club, long or confusing paths, random-looking query parameters, excessive redirection, and the absence of HTTPS.

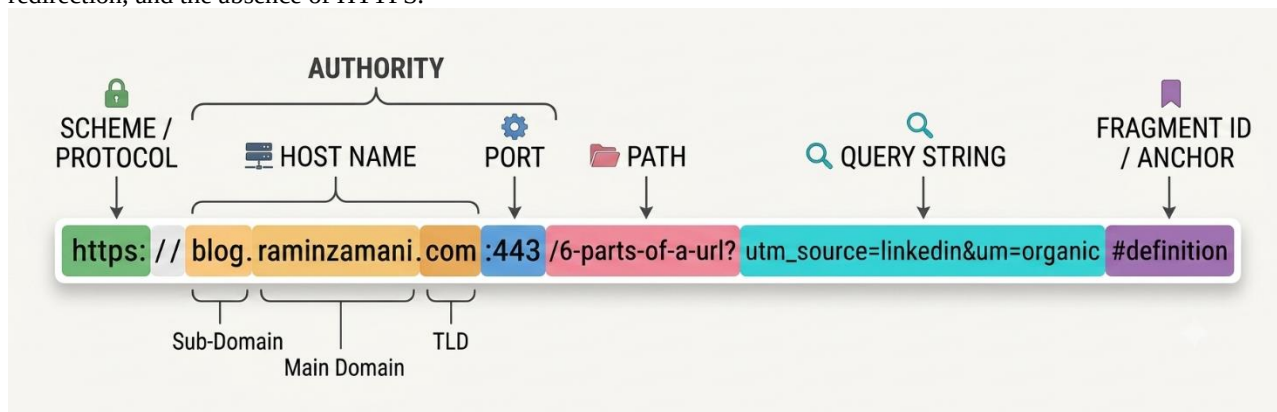


Fig. 1. Structural components of a URL: protocol, subdomain, main domain, TLD, port, path, query string and fragment (adapted from [10]).

2.3 Machine learning for phishing URL detection

Machine learning models classify URLs as legitimate or malicious on the basis of features extracted from the URL, from page activity, or from other artefacts; some approaches are vision-based and analyse an image of the rendered page, while others use site content [11]. The classifiers most commonly applied to this task are decision trees (DT), random forests (RF), logistic regression (LR), support vector machines (SVM), k-nearest neighbours (KNN) and gradient boosting ensembles such as XGBoost. Their relative performance depends on the feature set, the size of the dataset and the complexity of the URLs involved. Machine learning methods have been shown to deliver higher accuracy and fewer false positives than traditional list- and rule-based techniques when confronting dynamic phishing attacks [12].

A machine learning pipeline for this task proceeds through data collection, feature extraction, model training, model testing and prediction [13]. Random Forest and XGBoost have proved particularly effective. Random Forest is an ensemble that grows many decision trees on bootstrap samples and combines their independent predictions by majority vote, which improves accuracy and limits overfitting. XGBoost is an optimised implementation of gradient boosting that instead constructs trees sequentially, each correcting the residuals of its predecessors [11]. Aljofey et al. report that XGBoost attains 96.76% accuracy on varied sets of URL features, with Random Forest at 91.34% [14], while Hannousse and Yahiouche report Random Forest as the strongest classifier on their benchmark at 96.65% accuracy [4].

2.4 Feature extraction from URLs

Features for malicious URL detection are obtained from the URL itself, from the content of the web page, and from third-party services. Some heuristic features are difficult to obtain and expensive to compute, which raises the resource cost of the resulting detector. Two broad strategies exist: rule-based detection, which matches URLs against predefined markers and is fast and accurate for known threats but cannot generalise to unseen ones, and behaviour-analysis detection, which uses machine or deep learning and adapts more readily to new threats [15].

Common feature-engineering methods for URLs include string patterns and regular expressions that identify lexical characteristics, lookup tables or likelihood scores based on key sequences, n-gram and bag-of-words models, task-specific embeddings generated through URL segmentation, and queries to domain name servers or registrars for registration and server configuration data [16]. A significant body of work emphasises that URLs are sequences of characters and require extraction methods that respect this sequential nature [17], which motivates convolutional and recurrent architectures. Unlike natural language, however, the key components of a URL occupy largely fixed positions, and this structure is itself informative for algorithm design [18]. Psychological and social factors also shape URL construction, as when familiar brand names are placed in subdomains of phishing URLs [19].

2.5 One-dimensional convolutional neural networks

A convolutional neural network is built from input, convolution, pooling and fully connected layers, and has been applied successfully to image classification, object recognition and phishing detection [20]. The one-dimensional convolutional layer (Conv1D) is designed specifically for sequence data such as audio, time series and text [21]. Its central operation slides a learnable filter across the sequence, multiplying filter weights by the corresponding sequence values and summing the products to produce a single output; repeating this across the sequence yields a transformed representation that emphasises some patterns and suppresses others.

To process a URL with a Conv1D layer the string is first tokenised. Under character-level tokenisation the URL `https://example.com/page?id=123` is decomposed into its individual characters. This captures fine-grained detail but produces long sequences and struggles to represent higher-level meaning. Under subword tokenisation the same URL is split into meaningful units such as `https`, `://`, `example`, `.com`, `/`, `page`, `?`, `id`, `=` and `123`. Subword units capture higher-level patterns, shorten the sequence and may generalise better because they correspond to chunks that frequently co-occur [17]. Each token is then mapped to a dense vector by an embedding layer, so that tokens appearing in similar contexts, such as `www` and `.com`, acquire similar representations. Convolution over the embedded sequence produces feature maps that are passed to pooling and fully connected layers for classification.

2.6 Bayesian inference

Bayes' rule provides a principled way to analyse evidence in the light of prior knowledge. It was first derived by Thomas Bayes and later independently by Pierre-Simon Laplace, and after two centuries of debate has become a widely applied tool [5]. It is a mathematical statement of how a belief should be updated when new evidence arrives, which is precisely the operation required when the belief that a URL is phishing must be revised as external signals about that URL become available.

2.7 Particle swarm optimisation

Selecting hyperparameters is critical to the performance of machine learning and deep learning models, and manual tuning is both ineffective and time-consuming. Particle swarm optimisation (PSO) is a nature-inspired metaheuristic that simulates the social behaviour of flocking birds or schooling fish. It searches the solution space with a population of candidate

solutions, or particles, each updating its position according to its own best-known position and the best position discovered by the swarm [22]. PSO requires no gradient computation and can therefore be applied to non-differentiable, noisy or otherwise awkward objective functions, and it tends to avoid becoming trapped in local minima because it explores the space more globally, a property that is valuable when the hyperparameter landscape is highly non-convex [23]. In a directly relevant application, Gupta and Singhal used PSO to train an artificial neural network for phishing URL classification, varying learning rates, activation functions and the number of hidden and output layers, and found that the PSO-trained network outperformed a standard backpropagation network on precision and RMSE [24].

3. RELATED WORK

Numerous studies have explored deep learning models for phishing detection and reported high accuracy. This section reviews the work that most directly motivates the present framework.

Wei et al. proposed a CNN for phishing URL detection that analyses the URL text alone, using embeddings and one-hot encoding to extract information from the string [3]. On a dataset of 21,208 URLs drawn from the Common Crawl Foundation and PhishTank, the CNN reached 99.90% accuracy with computation approximately fifty times faster than an LSTM, which achieved 98.76%. The method is well suited to resource-constrained devices, but the authors note that it excludes contextual data from web content and does not evaluate robustness to adversarial attack.

Aljofey et al. introduced a character-level CNN that operates independently of third-party services, addressing the observation that reliance on such services slows classification and hampers detection of zero-day attacks [14]. The model combines CNN feature extraction with machine learning classifiers such as XGBoost, and categorises features into hand-crafted features, character embedding features, character-level TF-IDF features and character-level count vector features. It achieved 98.58% accuracy on benchmark datasets and 95.02% on a self-built dataset, is language-independent and achieves low detection times, though it may misclassify short URLs or those containing sensitive words.

Do et al. conducted an empirical comparison of deep neural networks, CNNs, LSTMs and gated recurrent units for phishing detection on the UCI dataset [25]. Their central finding was that no single algorithm outperformed the others across all metrics, so the choice of model must follow the requirements of the application. The study also documented the absence of standardised guidance for hyperparameter tuning in this field and contributed a detailed specification of hyperparameters and the tuning process.

Catal et al. carried out a systematic literature review of 43 articles on deep learning for phishing detection, identifying fourteen key challenges and organising the field around nine research questions [2]. The review found that URL-related features predominate in model development but that many studies do not specify which feature selection algorithms they used, and it highlighted model interpretability and real-time detection as underexplored. PhishTank was the most commonly used data source, followed by custom datasets, Alexa-Top Sites and UCI.

Alshingiti et al. proposed three deep learning models for phishing detection, an LSTM, a CNN and an LSTM-CNN hybrid, evaluated on the ISCX-URL-2016 dataset of 20,000 URLs reduced from 80 to 30 features by selection [26]. The CNN performed best at 99.2% accuracy, ahead of the LSTM-CNN hybrid at 97.6% and the LSTM at 96.8%. The authors identify the absence of URL status verification during training as a limitation affecting their results, which is directly relevant to the present work.

Aldakheel et al. presented a CNN-based detector achieving 98.77% accuracy on the PhishTank dataset through dataset preparation and preprocessing, character embedding of the URL, CNN analysis and retrieval of URL properties [27]. The study emphasises the significance of URL features and SSL/TLS certificate attributes, while noting that much existing research relies on synthetic datasets that may not reflect real-world threats, overlooks user interaction with phishing URLs, and concentrates on known attack types rather than emerging vectors.

Since these convolutional models were published, the field has shifted toward transformer architectures for URL-level detection. Wang et al. [28] introduced PhishBERT, a transformer pre-trained on a corpus of more than three billion unlabelled URLs with a tailored pre-training objective, then transferred to benign-versus-malicious classification through supervised fine-tuning using adversarial methods; the authors report gains in efficiency, robustness and accuracy over the then state of the art. Liu et al. [29] proposed TransURL, an end-to-end system built on the pretrained CharBERT backbone and co-trained with three feature modules: Multi-Layer Encoding, Multi-Scale Feature Learning and Spatial Pyramid Attention. TransURL requires no manual feature initialisation, processing raw URLs directly into character-aware subword embeddings, and reports a 40% peak F1-score improvement under class imbalance together with a 14.13% accuracy gain over baselines under adversarial attack. Most recently, Tawfik et al. [30] proposed XF-PhishBERT, which combines the ModernBERT encoder with prototypical networks and model-agnostic meta-learning to enable few-shot detection. It reports 99.9% accuracy with ten labelled examples per class and 98.5% in one-shot scenarios, and validates a browser-extension deployment at 98.3% precision with 42 ms average latency. XF-PhishBERT also incorporates SHAP-based explainability, a dimension absent both from the transformer baselines above and from the present work.

Hannousse and Yahiouche contributed the benchmark dataset used in this study, comprising 87 features across URL-based, content-based and external categories, and reported Random Forest as their strongest classifier at 96.65% accuracy [4].

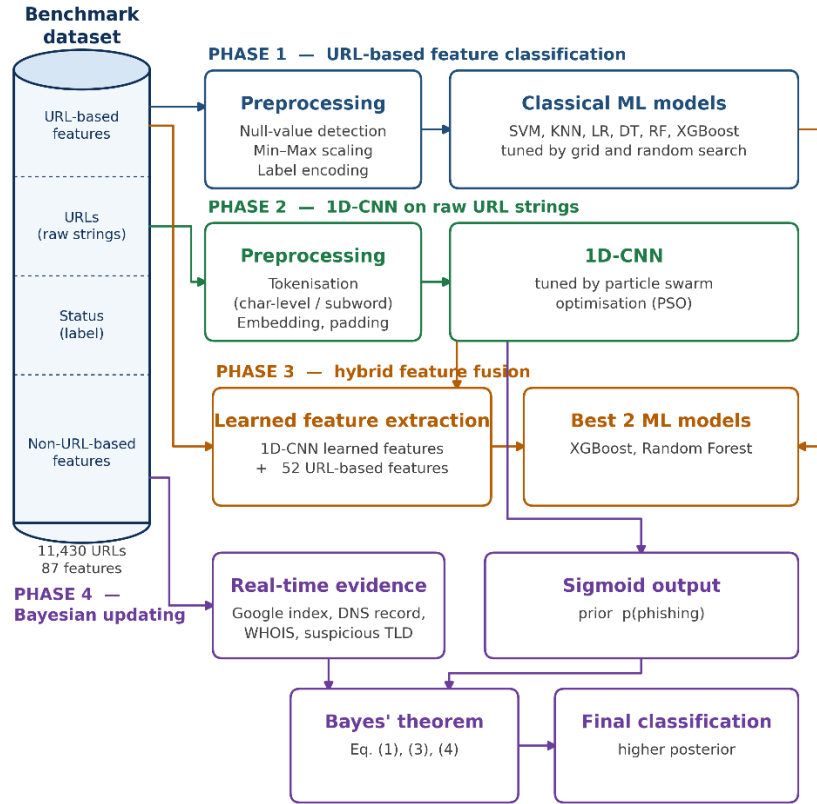
Three observations follow from this literature. First, the reported accuracies are uniformly high, but accuracy alone is a misleading target in a security setting: a false negative exposes a user to a live phishing site, whereas a false positive merely inconveniences them. The present study therefore treats recall as the primary metric and reports the precision that is traded for it. Second, the transformer results in particular exceed those reported here by a wide margin, and that gap should be stated plainly rather than explained away; it reflects pre-training corpora several orders of magnitude larger than the 11,430-URL benchmark used in this work, different evaluation datasets, and computational budgets that place these models outside the reach of a lightweight browser-side detector of the kind targeted here. The contribution of the present work is not a higher accuracy figure. Third, and most importantly, every model surveyed above is static once trained: whatever evidence exists about a URL at the moment a user opens it plays no part in the decision. XF-PhishBERT comes closest, refreshing its training corpus from live phishing feeds, but this updates the dataset rather than the individual classification. No surveyed method applies a probabilistic update that revises a trained model's per-URL decision using external signals gathered at access time, which is the gap the Bayesian layer proposed here is intended to address. Table I summarises the comparison.

TABLE I. COMPARATIVE ANALYSIS OF RELATED WORK

Reference	Model	Dataset	Reported performance
Wei et al. [3]	CNN with embeddings and one-hot encoding	Common Crawl, PhishTank (21,208 URLs)	CNN 99.90%; LSTM 98.76% accuracy
Aljofey et al. [14]	Character-level TF-IDF CNN with XGBoost	Benchmark datasets and self-built dataset	98.58% benchmark; 95.02% self-built
Do et al. [25]	DNN, CNN, LSTM, GRU (empirical study)	UCI	No single best model across metrics
Catal et al. [2]	Systematic literature review of 43 studies	PhishTank, custom, Alexa-Top Sites, UCI	Identified 14 challenges
Alshingiti et al. [26]	LSTM, CNN, LSTM-CNN hybrid	ISCX-URL-2016 (20,000 URLs)	CNN 99.2%; LSTM-CNN 97.6%; LSTM 96.8%
Aldakheel et al. [27]	CNN with character embedding	PhishTank	98.77% accuracy
Wang et al. [28]	PhishBERT (pretrained transformer)	>3 billion unlabelled URLs (pre-training)	Gains in efficiency, robustness, accuracy
Liu et al. [29]	TransURL (CharBERT + pyramid features)	Multiple public URL datasets	+40% peak F1 under class imbalance
Tawfik et al. [30]	XF-PhishBERT (ModernBERT, few-shot)	DATA1 (88,647), PhiUSIIL (235,795)	99.9% accuracy (10-shot); 98.5% (1-shot)
Hannousse and Yahiouche [4]	Six classical ML classifiers	Benchmark (11,430 URLs)	Random Forest 96.65% accuracy
This work	1D-CNN, ML hybrid and Bayesian updating	Benchmark (11,430 URLs)	Hybrid 92.53% accuracy; recall 92.01% with Bayesian layer

4. PROPOSED METHODOLOGY

Real-time detection requires a model that can act on the URL directly, because features derived from page content change over time and may be unavailable or unreliable at the moment a decision is needed. The methodology therefore proceeds in four phases, shown in Fig. 2. Phase 1 trains classical machine learning classifiers on handcrafted URL-based features. Phase 2 trains a 1D-CNN directly on raw URL strings, testing whether the model can detect phishing URLs without feature engineering. Phase 3 constructs a hybrid model by combining the features learned in Phase 2 with the handcrafted features of Phase 1 and passing the fused representation to the best classifiers identified in Phase 1. Phase 4 applies Bayes' theorem, using the 1D-CNN output as prior knowledge and updating it with evidence about the status of the URL.



All phases evaluated on the held-out test set (2,284 URLs) using accuracy, precision, recall and F1-score.

Fig. 2. The proposed four-phase methodology. Phase 1 applies classical classifiers to URL-based features; Phase 2 trains a 1D-CNN on raw URL strings; Phase 3 fuses both feature sets; Phase 4 applies Bayesian updating with non-URL-based evidence.

4.1 Dataset

The experiments use the benchmark dataset of Hannousse and Yahiouche [4], which contains 87 features together with the raw URL and a status label indicating whether each URL is legitimate or phishing. The features fall into two groups: URL-based features extracted directly from the URL string, and non-URL-based features obtained by loading the page, analysing its HTML content, or querying third-party services. The latter group divides further into 24 content-based features and 7 external features.

The dataset comprises 11,430 records in a balanced distribution, labelled 1 for phishing and 0 for legitimate. After preprocessing, it was partitioned into 70% for training, 10% for validation and 20% for testing, with the balanced class distribution maintained across all three partitions. The held-out test partition used throughout Section 5 contains 2,284 URLs, of which 1,170 are legitimate and 1,114 are phishing.

4.2 Phase 1: classification from URL-based features

In the first phase, 52 URL-based features were supplied to six classical classifiers: decision tree, random forest, logistic regression, support vector machine, k-nearest neighbours and XGBoost. Restricting this phase to features computable from the URL string alone is deliberate: features that require page content or a third-party lookup change dynamically and would bias a model intended for use at access time.

Preprocessing comprised detection of null values, scaling of each feature to a fixed range with the MinMaxScaler method, and conversion of the status label from string to integer by label encoding. Hyperparameters were tuned using grid search and random search. Grid search evaluates every combination within a specified subset of the hyperparameter space and is easily parallelised but scales poorly with dimensionality; random search samples combinations instead, and can be more effective when only a few hyperparameters materially affect performance [31]. The two best-performing classifiers were carried forward to Phase 3.

4.3 Phase 2: URL classification by 1D-CNN

The 1D-CNN treats the URL as a one-dimensional sequence, so preprocessing begins with tokenisation. Both character-level and subword tokenisation were evaluated. The integer-encoded token sequences produced by the tokeniser are then

converted into dense, continuous-valued vectors by an embedding layer, and padded so that the input length is uniform; the padded sequence length used here is 1,460 tokens, determined by the length of the longest URL in the dataset.

The subword tokenisation was implemented using the SubwordTextEncoder module from the TensorFlow Datasets (TFDS) library. The encoder was trained specifically on the raw URL strings of the benchmark dataset to build a vocabulary of 32,768 unique subword tokens. Unknown or out-of-vocabulary tokens are handled by the encoder's internal fallback mechanism, which breaks out-of-vocabulary sequences into smaller known subword units. No explicit rule-based pre-tokenisation (e.g., explicit splitting of `://` or `.com`) was applied; the tokeniser automatically learns these as frequent co-occurring patterns. The encoded sequences were then zero-padded to a uniform length of 1,460 tokens, determined by the maximum sequence length in the training set.

The architecture, shown in Fig. 3, comprises an embedding layer; two convolutional blocks, each containing a Conv1D layer with ReLU activation for feature mapping and extraction; batch normalisation layers, which stabilise training and help prevent overfitting; a global max pooling layer that reduces each feature map to a fixed-length representation; dropout layers between the dense layers to limit overfitting; and a sigmoid output that yields the probability of the URL being phishing. Hyperparameters were optimised with particle swarm optimisation; the resulting values are given in Table V.

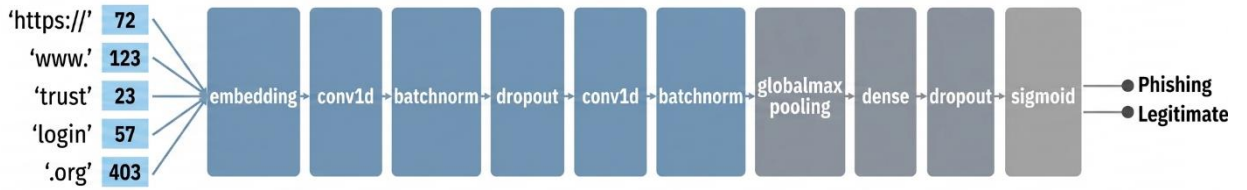


Fig. 3. Architecture of the 1D-CNN used for classification of raw URL strings.

4.4 Phase 3: hybrid feature fusion

The hybrid model concatenates the features learned by the 1D-CNN in Phase 2 with the 52 handcrafted URL-based features of Phase 1, and passes the combined vector to the best-performing classifiers identified in Phase 1. The intention is to combine explicit domain knowledge, encoded in the handcrafted features, with the representations the network discovers for itself, on the expectation that each supplies information the other does not.

The learned features from the 1D-CNN were extracted from the output of the penultimate Dense(32) layer, yielding a 32-dimensional feature vector per URL. These were then concatenated with the 52 URL-based handcrafted features (which had been standardised using MinMaxScaler, as in Phase 1) to form a fused feature vector of dimension 84. No additional feature selection or dimensionality reduction was applied. The fused vector was then used to train the two best-performing classifiers from Phase 1 (XGBoost and Random Forest) using the same hyperparameters as reported in Table IV.

4.5 Phase 4: Proof-of-Concept Bayesian Updating Using Simulated Evidence

The final phase addresses adaptability at the moment of access. To assess the potential of evidence-based revision without the operational overhead of live third-party queries, we simulate this process. We treat the pre-extracted values of external features from the benchmark dataset as static proxies for what would be obtained via live DNS, WHOIS, or Google queries. This simulation allows us to evaluate whether Bayesian updating could improve detection, acting as a proof-of-concept for future deployment. In this setting, correctly identifying true positives matters more than avoiding false alarms: it is far better to warn a user about a URL that turns out to be benign than to admit them to a harmful site. Recall is therefore the objective this phase optimises. Bayes' theorem is stated as

$$p(A|B) = [p(B|A) \times p(A)] / p(B) \quad (1)$$

where $p(A|B)$ is the probability that the URL is phishing given the observed evidence B; $p(B|A)$ is the likelihood of observing that evidence if the URL is indeed phishing; $p(A)$ is the prior probability that the URL is phishing before any evidence is observed; and $p(B)$ is the marginal probability of the evidence.

- **Prior knowledge:** In our simulated operational flow, we treat the 1D-CNN's sigmoid output as the prior probability, mirroring the sequence a deployed system would follow if it classified the URL before issuing external queries. The sigmoid output of the network, a value in $[0, 1]$, is therefore taken as the prior probability that the URL is phishing.
- **Evidence:** Four non-URL-based features from the benchmark dataset are used as evidence, each of which corresponds in principle to a query that a third-party service could answer at access time. A feature value of 1 indicates the presence of the signal and 0 its absence. The four are described in Table II. The assumption made here is that such evidence would be available, reliable and fast enough to avoid introducing latency.

TABLE II. NON-URL-BASED EVIDENCE USED IN THE BAYESIAN LAYER

Evidence	Description
Google index	Phishing websites live for short periods and are often reached through direct links sent in email, so they do not need to be indexed by Google.
DNS record	A DNS record is mandatory to retrieve the IP address of a URL for access. A missing DNS record is a phishing indicator.
WHOIS registration	Domains of phishing websites frequently do not match any WHOIS database record, unlike most legitimate domains.
Suspicious TLD	The top-level domain is checked against a list of TLDs disproportionately associated with phishing.

- Likelihood: The likelihood of each item of evidence is estimated from its distribution within the dataset, separately for the phishing and legitimate classes. Following the naive Bayes assumption that the evidence features are conditionally independent given the class, the joint likelihood is the product of the individual likelihoods:

$$p(E_1, E_2, \dots, E_n | \text{phishing}) = \prod p(E_i | \text{phishing}) \quad (2)$$

and correspondingly for the legitimate class. Only features that are present contribute to the product; where a feature is absent, no evidence is deemed to have been observed and the term is omitted. The implications of this convention are considered in Section 6.

- Posterior and classification. The posterior probability of each class is obtained by substituting the likelihoods and the 1D-CNN prior into Bayes' theorem. Because the resulting quantities are proportional rather than normalised, they are divided by their sum so that the two posteriors add to one:

$$p(\text{phishing} | E_1, \dots, E_n) = [\prod p(E_i | \text{phishing}) \times p(\text{phishing})] / p(E_1, \dots, E_n) \quad (3)$$

$$p(\neg \text{phishing} | E_1, \dots, E_n) = [\prod p(E_i | \neg \text{phishing}) \times p(\neg \text{phishing})] / p(E_1, \dots, E_n) \quad (4)$$

The URL is assigned to whichever class has the higher posterior probability. When no evidence is present the posterior reduces to the 1D-CNN prior, so the Bayesian layer degrades gracefully to the behaviour of the underlying network.

Table III lists the likelihoods estimated from the dataset and works through a complete calculation for the legitimate URL <http://www.mutuo.it>. For this record only the Google index feature is present, so it alone enters the likelihood product. The 1D-CNN assigns a very low prior probability of phishing, and the evidence does not overturn it: the posterior probability of the legitimate class is 0.99984 and the URL is classified correctly.

TABLE III. ESTIMATED EVIDENCE LIKELIHOODS AND A WORKED BAYESIAN CALCULATION FOR THE LEGITIMATE URL [HTTP://WWW.MUTUO.IT](http://www.mutuo.it)

Quantity	Source	Value
$p(\text{google_index} = 1 \text{phishing})$	Dataset distribution	0.89869
$p(\text{whois_registered_domain} = 1 \text{phishing})$	Dataset distribution	0.09029
$p(\text{dns_record} = 1 \text{phishing})$	Dataset distribution	0.03727
$p(\text{suspicious_tld} = 1 \text{phishing})$	Dataset distribution	0.03255
$p(\text{google_index} = 1 \text{legitimate})$	Dataset distribution	0.16920
$p(\text{whois_registered_domain} = 1 \text{legitimate})$	Dataset distribution	0.05547
$p(\text{dns_record} = 1 \text{legitimate})$	Dataset distribution	0.00297
$p(\text{suspicious_tld} = 1 \text{legitimate})$	Dataset distribution	0.00332
$p(\text{phishing})$ prior	1D-CNN sigmoid output	0.0000294
$p(\text{phishing} \text{evidence})$	Posterior, Eq. (3)	0.00016
$p(\text{legitimate} \text{evidence})$	Posterior, Eq. (4)	0.99984
Final classification	Higher posterior	Legitimate (correct)

4.6 Evaluation metrics

Model performance is assessed with accuracy, precision, recall and F1-score, computed from the counts of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN):

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{F1-score} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

Recall is treated as the primary metric throughout. It quantifies the proportion of actual phishing URLs the model identifies, so a false negative is a phishing URL delivered to the user unflagged, with the attendant risk of credential theft and data breach. A false positive, by contrast, produces an unnecessary warning. The two errors are not symmetric in cost, and a detector for real-time use should be tuned accordingly, accepting some increase in false alarms in exchange for more complete coverage of genuine threats.

5. EXPERIMENTAL RESULTS

5.1 Phase 1: classical machine learning baselines

The classifiers were trained on the 52 URL-based features and tuned by grid and random search; the selected hyperparameters appear in Table IV. XGBoost was the strongest of the six, reaching 91.21% accuracy and 92.18% precision, marginally ahead of Random Forest at 90.81% accuracy. Random Forest, however, held an advantage in recall at 90.35% against 89.81%, meaning it recovered more phishing URLs and produced fewer false negatives. Given the priority placed on recall, both were carried forward to Phase 3. Logistic regression was weakest, with recall of 81.22%. The full results are reported in Table VI.

TABLE IV. SELECTED HYPERPARAMETERS FOR THE CLASSICAL CLASSIFIERS

Classifier	Hyperparameters
Logistic Regression	$C = 1438.449$; solver = liblinear
Support Vector Machine	$C = 5$; gamma = 0.25; kernel = rbf
Decision Tree	criterion = Gini; max_depth = 9; min_samples_leaf = 7; min_samples_split = 2; splitter = best
Random Forest	max_depth = None; max_features = sqrt; min_samples_leaf = 1; min_samples_split = 2; n_estimators = 900
XGBoost	learning_rate = 0.37218; max_depth = 5; n_estimators = 317
k-Nearest Neighbours	$k = 18$

For the same set of URL-based features, these classifiers reproduced the results reported by Hannousse and Yahiouche [4]. The addition of XGBoost, which was not among the models they evaluated, provided a modest further gain, plausibly because gradient boosting captures non-linear interactions among the URL features that the other classifiers do not. It should be noted that the 96.65% accuracy reported in [4] was obtained on a broader feature set than the URL-only subset used here, so the two figures are not directly comparable.

5.2 Phase 2: 1D-CNN architecture and tokenisation

Several configurations were evaluated. With character-level tokenisation and a single convolutional layer the model reached 77.80% training accuracy and 77.75% validation accuracy. Adding further convolutional layers raised performance by approximately 18 percentage points, as Fig. 4 shows.

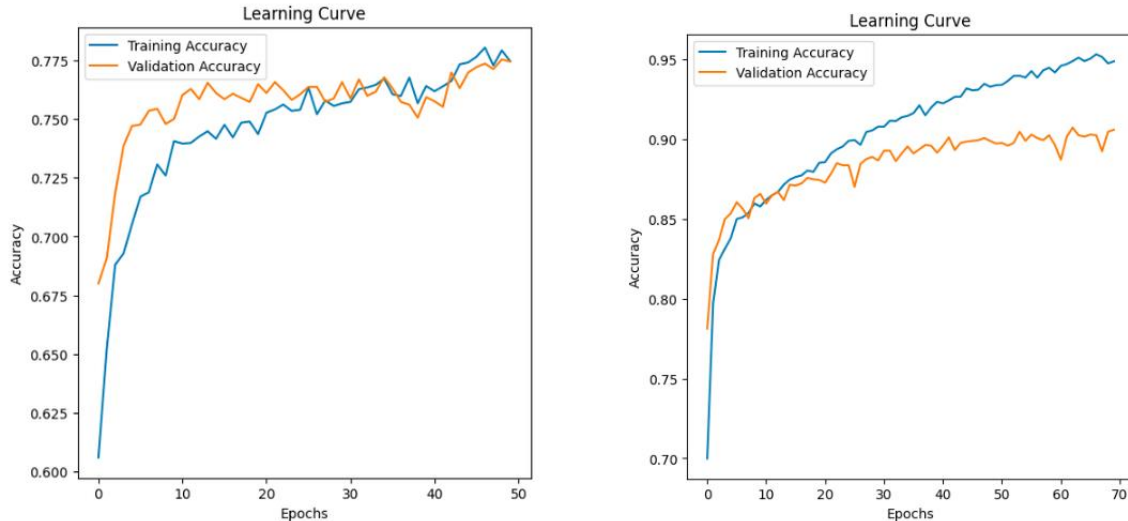


Fig. 4. Training and validation accuracy of the 1D-CNN with character-level tokenisation: one convolutional layer (left) and three convolutional layers (right).

Switching to subword tokenisation with three convolutional layers produced 99.11% training accuracy but only 90.01% on unseen data. The gap between the two curves in Fig. 5 is a clear signature of overfitting: the network had begun to memorise the training set rather than learn generalisable structure.

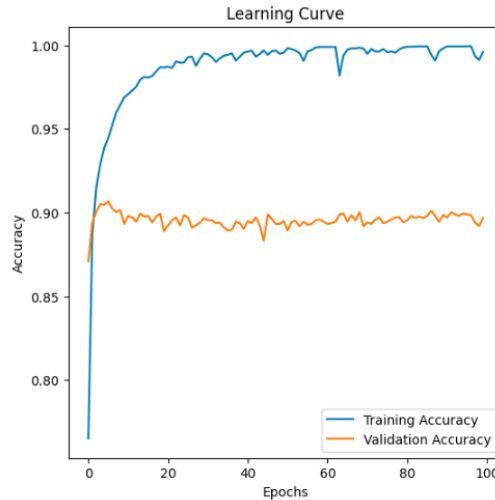


Fig. 5. Training and validation accuracy with subword tokenisation and three convolutional layers. The persistent gap between the curves indicates overfitting.

Particle swarm optimisation was configured so that each particle represents a candidate set of hyperparameters: kernel size, dropout rate, learning rate, batch size, and number of epochs. The search space for each hyperparameter was defined as: kernel size $\in [2, 10]$, dropout rate $\in [0.1, 0.5]$, learning rate $\in [1e-5, 1e-2]$ (logarithmic scale), batch size $\in [32, 256]$ (step 16), and epochs $\in [5, 30]$. The inertia weight was set to 0.7298, with cognitive and social coefficients $c1 = c2 = 1.496$, following standard recommendations in Shi & Eberhart, 1998. Velocity was clamped to $[-0.5, 0.5]$ of the parameter range. The objective function was validation accuracy. The swarm was initialised randomly, and optimisation terminated after 50 iterations or when the best fitness value did not improve for 10 consecutive iterations.

The final configuration added an embedding layer with batch normalisation and tuned the hyperparameters by particle swarm optimisation, using the values in Table V. This raised validation accuracy by roughly two percentage points to 92.25%, with precision of 93.34%, and narrowed the generalisation gap considerably (Fig. 6). The model thus outperformed every classical baseline while requiring no manual feature engineering at all, learning directly from the raw URL string.

TABLE V. HYPERPARAMETERS OF THE 1D-CNN SELECTED BY PARTICLE SWARM OPTIMISATION

Hyperparameter	Value
Kernel size	6
Dropout rate	0.372123
Learning rate	0.000672
Batch size	113
Epochs	17

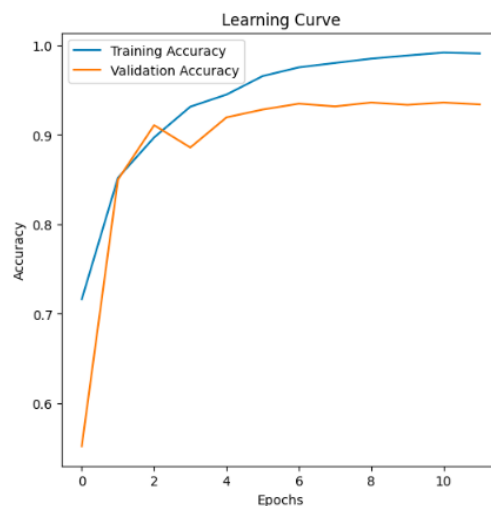


Fig. 6. Training and validation accuracy of the final 1D-CNN, with embedding, batch normalisation and PSO-tuned hyperparameters.

5.3 Phase 3: hybrid model

Fusing the 52 handcrafted URL-based features with the features learned by the 1D-CNN and classifying with the two strongest models from Phase 1 produced the best overall results of the study. XGBoost reached 92.53% accuracy, 92.53% precision, 92.52% recall and 92.52% F1-score, with Random Forest at 92.48% on all four metrics. The near-identical values across metrics indicate a well-balanced classifier: the fused representation improves recall relative to the standalone 1D-CNN, which stood at 90.57%, without sacrificing precision. This supports the view that the handcrafted and learned features are complementary rather than redundant.

5.4 Phase 4: Bayesian updating

Applying Bayes' theorem with the 1D-CNN output as the prior and the four items of evidence changed the balance of errors in the intended direction. In this simulation, false negatives fell from 105 to 89 out of 1,114 phishing URLs, a reduction of 15.2%, so that recall rose from 90.57% to 92.01%, demonstrating the potential of the Bayesian approach to improve recall, assuming the evidence proxies are representative of live signals. The cost was a rise in false positives from 72 to 96 out of 1,170 legitimate URLs, lowering precision from 93.34% to 91.44% and overall accuracy marginally from 92.25% to 91.90%. The two confusion matrices are shown in Fig. 7.

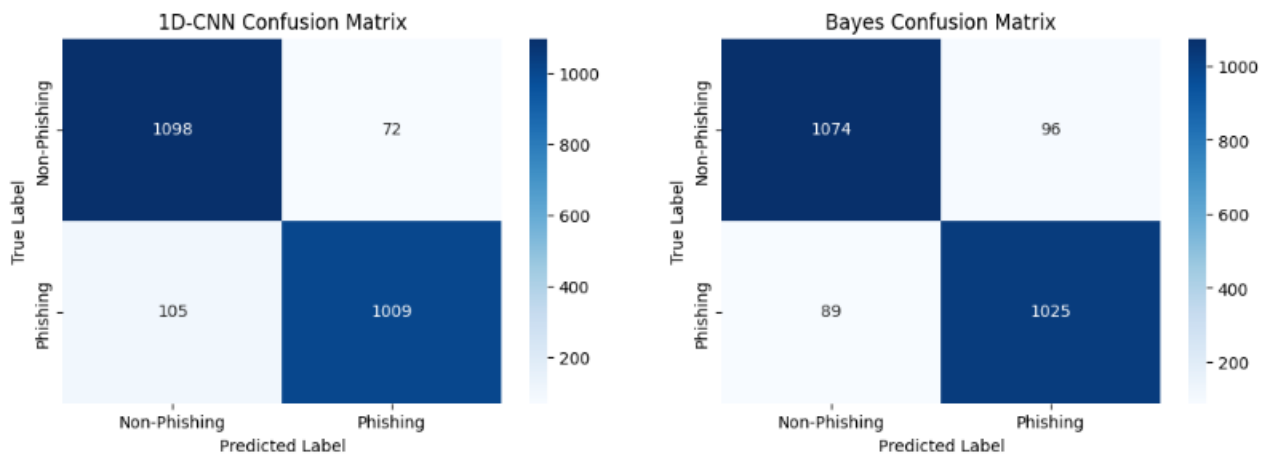


Fig. 7. Confusion matrices on the held-out test set for the standalone 1D-CNN (left) and with Bayesian updating (right).

This is the trade-off the framework was designed to make. Sixteen additional phishing URLs were intercepted, each representing a potential security incident averted, at the price of twenty-four additional false warnings. Under the asymmetric cost structure of phishing detection, that exchange is favourable. It should be stated plainly, however, that overall accuracy did not improve; recall did, and accuracy fell slightly as a consequence.

The behaviour of the Bayesian layer falls into two regimes. Where the 1D-CNN prior is confident and the evidence corroborates it, the posterior reinforces the correct classification. Where the prior is uncertain and the evidence points the other way, for instance when the network judges a URL legitimate but the DNS, WHOIS and TLD signals all indicate otherwise, the Bayesian layer successfully revises the decision to malicious. The failure mode is the third case: when the prior is highly confident and no evidence is present, the prior dominates the posterior and the misclassification stands. Fig. 8 shows the URLs that remained misclassified after Bayesian adjustment.

	A	B	C	D	E	F	G	H
1	url	status	Bayes predicted	whois_registered_domain	dns_record	google_index	suspicious_tld	
2	https://www.brightdestructive	1	0	0	0	1	0	
3	http://www.davidcourtemarch	1	0	0	0	1	0	
4	https://comunicazioniarubait.f	1	0	0	0	1	0	
5	http://www.sapl.com.hk/swfo	1	0	0	0	0	0	
6	https://jagex.nu	1	0	0	0	0	0	
7	https://www.ulrc.go.ug/scripts	1	0	0	0	0	0	
8	http://www.easychemproject.	1	0	0	0	1	0	
9	https://www.developsys.net/ii	1	0	0	0	1	0	
10	http://blog.fbhackpass.com/se	1	0	0	0	0	0	
11	http://www.masanova.com/A	1	0	0	0	1	0	
12	https://interbahis452.blogspot	1	0	0	0	1	0	
13	https://geni.us/document2121	1	0	0	0	1	0	
14	http://www.figoodies.com/esj	1	0	0	0	1	0	
15	http://www.aigr-contadores.c	1	0	0	0	0	0	
16	http://www.thatsmint.work/ili	1	0	1	0	1	1	
17	https://www.restrepohenao.c	1	0	0	0	1	0	
18	http://www.facturation90rang	1	0	0	0	1	0	
19	https://betasusgiris2.blogspot	1	0	0	0	1	0	
20	https://www.mercus.com.pl/fi	1	0	0	0	1	0	
21	http://www.oxfordlimoservice	1	0	0	0	1	0	
22	http://www.imeipro.info/att-ii	1	0	0	0	0	0	

Fig. 8. Phishing URLs that remained misclassified after Bayesian adjustment, together with their evidence values.

TABLE VI. PERFORMANCE OF ALL MODELS ON THE HELD-OUT TEST SET (%)

Phase	Model	Accuracy	Precision	Recall	F1-score
1	Logistic Regression	85.35	88.17	81.22	84.56
1	Decision Tree	85.74	86.20	84.68	85.43
1	k-Nearest Neighbours	86.83	87.98	84.94	86.44
1	Support Vector Machine	89.37	91.32	86.71	88.96
1	Random Forest	90.81	90.99	90.35	90.67
1	XGBoost	91.21	92.18	89.81	90.98
2	1D-CNN	92.25	93.34	90.57	91.94
3	Hybrid (Random Forest)	92.48	92.48	92.48	92.48
3	Hybrid (XGBoost)	92.53	92.53	92.52	92.52
4	1D-CNN with Bayesian updating	91.90	91.44	92.01	91.72

6. DISCUSSION

Three findings emerge from these experiments. The first concerns tokenisation. Subword tokenisation was decisive for 1D-CNN performance on URL data: character-level tokenisation plateaued at roughly 77% accuracy because individual characters carry little semantic weight, whereas subword units correspond to the structural components of a URL, the protocol, domain and TLD, and give the convolutional filters something meaningful to match. This accounts for the eighteen-percentage-point gap between the two strategies.

The second concerns feature fusion. The hybrid model reached 92.53% accuracy against 92.25% for the standalone 1D-CNN, a small gain in accuracy but a substantially better balance across metrics, with recall rising from 90.57% to 92.52%. The handcrafted features appear to act as a stabilising signal on URLs where the learned representation is uninformative, so the benefit of fusion is robustness rather than a large additive improvement.

The third concerns the simulated Bayesian layer, which addresses a limitation shared by every model reviewed in Section 3: their inability to incorporate contextual signals after training. While our implementation uses static dataset proxies, the simulation shows that if such evidence were available in real-time, it could effectively revise the prior. The gain in recall corresponds to sixteen additional phishing URLs correctly identified in the test set. In a browser-side deployment the four evidence queries could be issued asynchronously. DNS queries are typically fast and Google index checks generally complete in under half a second; TLD reputation checks are also fast, particularly when the list is pre-cached. WHOIS lookups are the bottleneck, sometimes taking several seconds depending on the registrar and TLD, and are also the least reliable, since WHOIS data can be incomplete or difficult to retrieve for certain TLDs. With caching and asynchronous requests, retrieving all four signals and updating the prediction is estimated at one to two seconds in the best case.

The residual errors of the 1D-CNN are instructive about the limits of URL-only analysis. Legitimate URLs containing special characters such as @, % or = can resemble phishing patterns and be flagged incorrectly, because those characters occur in both legitimate web services and attacks. Conversely, phishing URLs that are short, well-structured and built from trustworthy-looking words evade detection precisely because they lack the markers the model has learned to associate with phishing. There is also a slower drift: terms such as "secure-login", "verify-account" and "update-info" were historically indicative of phishing, but as legitimate businesses adopted the same reassuring language these terms migrated into genuine URLs. A model trained on such assumptions will progressively misclassify legitimate traffic, which argues for periodic retraining rather than one-off deployment.

While transformer-based models report superior accuracy (up to 99.9%), they require substantial computational resources and pre-training on billion-scale corpora. The proposed 1D-CNN achieves 92.25% accuracy with significantly fewer parameters and lower inference latency, making it suitable for resource-constrained browser-side deployment where sub-second response and local execution are required. This trade-off (modestly lower accuracy for substantially reduced computational cost) is deliberate and justified for the target scenario, as transformer alternatives introduce latency, privacy, and availability dependencies that our design avoids.

- **Limitations:** Several constraints bound the strength of the conclusions that can be drawn. Consequently, the reported improvements for Phase 4 must be interpreted strictly as hypothetical gains under an idealized simulation, rather than validated operational metrics. The most important is that the likelihoods used in Phase 4 were estimated from the distribution of the four evidence features within the same benchmark dataset, not from live queries issued at access time. In this study, the likelihoods for the four evidence items were estimated from the same benchmark dataset used to train and evaluate the framework, rather than from independent live queries. Consequently, the Bayesian phase is presented strictly as a proof-of-concept demonstrating the mechanism of evidence-based revision under idealised simulation, and future retrieving work must validate with live evidence retrieval on temporally distinct test sets. Second, the naive Bayes conditional independence assumption is unlikely to hold in practice, since a missing DNS record and an absent WHOIS registration are plausibly correlated; treating them as independent will overstate the strength of their joint evidence. Third, the convention of omitting absent features from the likelihood product discards information, because the absence of a phishing indicator is itself weakly evidential; a formulation that conditions on both presence and absence would use the available signal more fully. Fourth, all results derive from a single benchmark dataset, so cross-dataset generalisation is not established. Finally, the dataset is balanced by construction, whereas legitimate URLs vastly outnumber phishing

URLs in deployment; because the Bayesian formulation operates on the likelihood of the observed evidence rather than on the raw class distribution, it is expected to tolerate this skew, but that expectation has not been tested here.

7. CONCLUSION

This paper presented a four-phase framework for phishing URL detection that combines classical machine learning on handcrafted URL features, a 1D-CNN operating on raw URL strings, fusion of the two feature sets, and Bayesian updating with non-URL-based evidence. Evaluated on a balanced benchmark of 11,430 URLs, the 1D-CNN demonstrated that raw URL strings carry enough structural information for high-accuracy detection without any manual feature engineering, reaching 92.25% accuracy with subword tokenisation and PSO-tuned hyperparameters. Fusing learned and handcrafted features improved this to 92.53% with a markedly better precision-recall balance.

The simulated Bayesian layer demonstrated a potential increase in recall from 90.57% to 92.01%, suggesting that evidence-based posterior revision could reduce undetected phishing URLs by approximately 15.2% in an idealized setting. Under the asymmetric costs of phishing detection this is the preferable operating point, and the result supports the wider argument that recall, not accuracy, is the metric against which such systems should be judged. The framework is intended for browser-side deployment, where the Bayesian component adds little computational overhead relative to the evidence-retrieving latency it depends on. However, as detailed in Section 6, the Bayesian phase remains a proof-of-concept simulation; its reported recall gains are in-sample estimates and await validation with live evidence retrieval.

Future work follows directly from the limitations. The evidence should be retrieved by live third-party queries at access time, with their availability and latency measured rather than assumed. The evidence set should be extended to indicators such as page rank, domain age and length of domain registration, and the contribution of each assessed. The conditional independence assumption should be relaxed, for example by modelling the dependencies among evidence explicitly in a Bayesian network. Alternative deep architectures could be substituted for the 1D-CNN as the source of the prior, the URL-specific transformer encoders discussed in Section 3 being the obvious candidates, since the quality of that prior governs the behaviour of the whole framework. Finally, evaluation on additional and deliberately imbalanced datasets would establish whether the results reported here generalise beyond a single benchmark.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

The research received no specific grant from any funding agency.

Data availability

The benchmark dataset analysed in this study is publicly available at <https://data.mendeley.com/datasets/c2gw7fy2j4/3>

References

- [1] Anti-Phishing Working Group, "Phishing activity trends report, 3rd quarter 2024," APWG, Dec. 4, 2024. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q3_2024.pdf.
- [2] C. Catal, G. Giray, B. Tekinerdogan, S. Kumar, and S. Shridhar, "Applications of deep learning for phishing detection: A systematic literature review," *Knowl. Inf. Syst.*, vol. 64, pp. 1457-1500, 2022, doi: 10.1007/s10115-022-01672-x.
- [3] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Wozniak, "Accurate and fast URL phishing detector: A convolutional neural network approach," *Comput. Netw.*, vol. 178, art. no. 107275, 2020, doi: 10.1016/j.comnet.2020.107275.
- [4] A. Hannousse and S. Yahiouche, "Towards benchmark datasets for machine learning based website phishing detection: An experimental study," *Eng. Appl. Artif. Intell.*, vol. 104, art. no. 104347, 2021, doi: 10.1016/j.engappai.2021.104347.
- [5] J. V. Stone, *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. Sheffield, U.K.: Sebtel Press, 2013, doi: 10.13140/2.1.1371.6801.
- [6] M. A. Rader and S. M. Rahman, "Exploring historical and emerging phishing techniques and mitigating the associated security risks," *Int. J. Netw. Secur. Appl.*, vol. 5, no. 4, pp. 23-41, Jul. 2013, doi: 10.5121/ijnsa.2013.5402. [Online]. Available: <https://arxiv.org/abs/1512.00082>.
- [7] G. Sonowal, "Phishing and communication channels," in *Phishing and Communication Channels*. Berkeley, CA, USA: Apress, 2021, doi: 10.1007/978-1-4842-7744-7.
- [8] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 1-28, 2011, doi: 10.1145/2019599.2019606.
- [9] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic syntax," RFC 3986, Jan. 2005.
- [10] R. Zamani, "6 parts of a URL every marketer should know," Medium, Mar. 24, 2023. [Online]. Available: <https://medium.com/@raminzamani/6-parts-of-a-url-every-marketer-should-know-7df83bb50d42>

- [11] V. Shahrivari, M. M. Darabi, and M. Izadi, "Phishing detection using machine learning techniques," 2020, arXiv:2009.11116.
- [12] Y. A. Alsariera, V. E. Adeyemo, A. O. Balogun, and A. K. Alazzawi, "AI meta-learners and extra-trees algorithm for the detection of phishing websites," *IEEE Access*, vol. 8, pp. 142532-142542, 2020, doi: 10.1109/ACCESS.2020.3013699.
- [13] L. Tang and Q. H. Mahmoud, "A survey of machine learning-based solutions for phishing website detection," *Mach. Learn. Knowl. Extr.*, vol. 3, no. 3, pp. 672-694, 2021, doi: 10.3390/make3030034.
- [14] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J.-P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, art. no. 1514, 2020, doi: 10.3390/electronics9091514.
- [15] C. D. Xuan, H. Dinh, and T. Victor, "Malicious URL detection based on machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 1, 2020, doi: 10.14569/ijacsa.2020.0110119.
- [16] J. Hawkins, "What's in a domain? Analysis of URL features," in *Proc. CS & IT Conf.*, 2023. [Online]. Available: <https://csitcp.org/abstract/13/1314csit09>
- [17] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," 2018, arXiv:1802.03162.
- [18] L. K. Shih and D. R. Karger, "Using URLs and table layout for web classification tasks," in *Proc. 13th Int. Conf. World Wide Web (WWW '04)*, New York, NY, USA, 2004, pp. 193-202.
- [19] H. Tupsamudre, A. K. Singh, and S. Lodha, "Everything is in the name - A URL based approach for phishing detection," in *Lecture Notes in Computer Science*, 2019, pp. 231-248, doi: 10.1007/978-3-030-20951-3_21.
- [20] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, vol. 10, pp. 36429-36463, 2022, doi: 10.1109/ACCESS.2022.3151903.
- [21] A. O. Ige and M. Sibiya, "State-of-the-art in 1D convolutional neural networks: A survey," *IEEE Access*, vol. 12, pp. 144082-144105, 2024, doi: 10.1109/ACCESS.2024.3433513.
- [22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95 - Int. Conf. Neural Netw.*, vol. 4, Perth, WA, Australia, 1995, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.
- [23] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput.*, Anchorage, AK, USA, 1998, pp. 69-73, doi: 10.1109/ICEC.1998.699146.
- [24] S. Gupta and A. Singhal, "Phishing URL detection by using artificial neural network with PSO," in *Proc. 2nd Int. Conf. Telecommun. Netw. (TEL-NET)*, Noida, India, 2017, pp. 1-6, doi: 10.1109/TEL-NET.2017.8343553.
- [25] N. Q. Do, A. Selamat, O. Krejcar, T. Yokoi, and H. Fujita, "Phishing webpage classification via deep learning-based algorithms: An empirical study," *Appl. Sci.*, vol. 11, no. 19, art. no. 9210, 2021, doi: 10.3390/app11199210.
- [26] Z. Alshingiti, R. Alaql, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, art. no. 232, 2023, doi: 10.3390/electronics12010232.
- [27] E. A. Aldakheel, M. Zakariah, G. A. Gashgari, F. A. Almarshad, and A. I. A. Alzahrani, "A deep learning-based innovative technique for phishing detection in modern security with uniform resource locators," *Sensors*, vol. 23, no. 9, art. no. 4403, 2023, doi: 10.3390/s23094403.
- [28] Y. Wang, W. Zhu, H. Xu, Z. Qin, K. Ren, and W. Ma, "A large-scale pretrained deep model for phishing URL detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Rhodes Island, Greece, 2023, pp. 1-5, doi: 10.1109/ICASSP49357.2023.10095719.
- [29] R. Liu, Y. Wang, Z. Guo, H. Xu, Z. Qin, W. Ma, and F. Zhang, "TransURL: Improving malicious URL detection with multi-layer Transformer encoding and multi-scale pyramid features," *Comput. Netw.*, vol. 253, art. no. 110707, 2024, doi: 10.1016/j.comnet.2024.110707.
- [30] M. Tawfik, A. A. Abu-Ein, A. H. Abdelhaliem, Y. M. Al-Sharo, and I. S. Fathi, "Explainable few-shot learning with modern BERT for detecting emerging phishing attacks using XF-PhishBERT," *Sci. Rep.*, vol. 15, art. no. 42821, 2025, doi: 10.1038/s41598-025-27500-0.
- [31] P. Liashchynskiy and P. Liashchynskiy, "Grid search, random search, genetic algorithm: A big comparison for NAS," 2019, arXiv:1912.06059.