Smart Phishing Detection via URL Characteristics: Machine Learning and Deep Learning Techniques

Ozan Duru
Computer Engineering Department
Biruni University, Istanbul, Turkey
210408005@st.biruni.edu.tr

***Abstract*—In today's interconnected world, phishing remains one of the most pervasive threats to online security, deceiving users into revealing sensitive data such as account credentials, identity documents, and payment details by mimicking legitimate websites that are visually indistinguishable from their authentic counterparts. To address this challenge, we propose a URL-based phishing detection framework that leverages both classical machine learning and deep learning techniques. From each URL, 22 hand-crafted features are extracted and standardized, ranging from basic statistics (lengths, character counts, Shannon entropy) to structural and content-based indicators (subdomain count, suspicious word presence, protocol flags). On the `small_dataset` of 520 285 URLs—assembled by merging the original train (70%), validation (20%), and test (10%) splits into a single CSV file—we train and evaluate five traditional classifiers (Logistic Regression, K-Nearest Neighbors, Decision Tree, Support Vector Machine, and Random Forest) and three neural architectures (fully-connected ANN, 1D-CNN, and RNN with LSTM units) using 5-fold cross-validation. Among the classical methods, Random Forest achieves the highest accuracy of 94.82%, while the leading deep models, ANN and RNN (LSTM) achieve an accuracy of 94.47% and 94.59%. These results demonstrate the superiority of ensemble tree methods for phishing URL classification and highlight the competitive performance of sequence-based neural networks in capturing URL dynamics. The proposed system offers a robust and reproducible approach to phishing detection in real time and can serve as a foundation for more advanced security analytics.**

***Index Terms*—phishing detection, URL feature engineering, machine learning, deep learning, Random Forest, LSTM**

I. Introduction

In recent years, there has been a growing body of research that integrates intelligent systems into cybersecurity applications. Similar to the process mining approach in business process management, which emphasizes the extraction of structured insights from operational data [3], the phishing detection model also relies on the systematic analysis of URL-based features to uncover malicious patterns. Furthermore, the use of deep learning techniques in this study resonates with recent efforts in static analysis-based Android malware detection [4], where convolutional and recurrent neural networks have been utilized to automatically learn discriminative features. Both studies underscore the importance of domain-specific feature engineering and robust classification strategies in developing effective and scalable security solutions.

979-8-3315-9727-6/25/$31.00 ©2025 IEEE

This emphasis on feature analysis is particularly crucial in the context of phishing detection, where attackers frequently design URLs to closely resemble legitimate domains. Phishing URLs are often crafted to look nearly identical to their genuine counterparts, but closer inspection of the address bar reveals tell-tale signs. Consider the fraudulent PayPal login page shown in Figure 1. Although the form and branding perfectly mimic the authentic site, the browser security indicator clearly shows 'Not Secure' (Figure 2), as the attacker has not obtained a valid TLS certificate. Moreover, the actual domain name contains subtle alterations—such as an extra hyphen—rendering it `paypal---accounts.com` instead of `paypal.com`, as shown in Figure 3. These minor changes exploit users' trust in familiar brands while covertly redirecting sensitive credentials to the attacker's server. Therefore, precise URL feature inspection remains a cornerstone of effective phishing detection strategies.

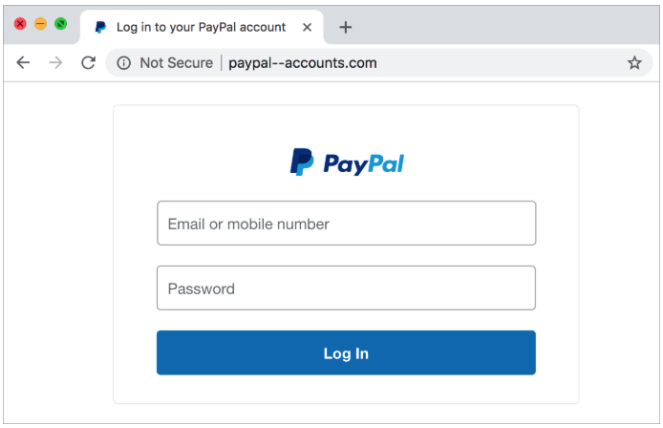

Fig. 1. Phishing page imitating PayPal's login screen.

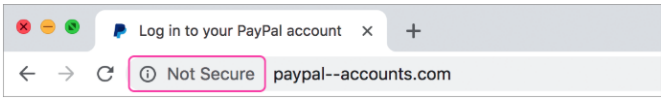

Fig. 2. "Not Secure" warning in the browser address bar.

As illustrated in Figure 4, the projected global cost of phishing attacks is expected to surge from $0.86 trillion in 2018 to $13.8 trillion by 2028—an increase of more than sixteenfold within a single decade. This steep upward trajectory

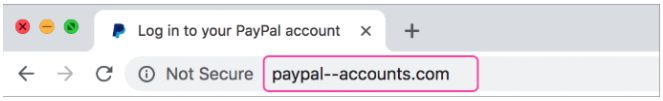

Fig. 3. Deceptive domain name `paypal--accounts.com` instead of `paypal.com`.

underscores the urgency of deploying robust, multi-layered counter-phishing strategies before the economic impact becomes unmanageable.

According to the *2025 Phishing Trends Report* by Hoxhunt [5], phishing is expanding fast, driven by AI and "phishing-as-a-service" kits. In 2024, 64% of organisations faced business-email compromise, losing an average of $150 000 per incident. About 80% of attacks harvested credentials—mainly against Microsoft 365 and Google Workspace. Over 80% of phishing sites now use HTTPS, while voice phishing ("vishing") and QR scams ("quishing") rose 30% and 25%, respectively. Impersonation is rife—Microsoft, Amazon, and Facebook are spoofed most—and dark-web phishing-kit listings grew 50%. Companies running regular, gamified awareness training report more than 60% of real threats and see an 86% drop in successful attacks within a year, proving that user education is as vital as technical controls.

The primary objectives of this work are to develop a robust and reproducible URL-based phishing detection framework through several key steps. First, we perform feature engineering by extracting 22 URL-level features—including length, character counts, entropy, and structural or content-based cues—to differentiate between phishing and legitimate links. We then train and evaluate five classical machine learning models (Logistic Regression, KNN, Decision Tree, SVM, and Random Forest) and three deep learning models (ANN, 1D-CNN, and LSTM-RNN) on a dataset of 520,285 URLs using 5-fold cross-validation. A comprehensive performance comparison is conducted using metrics such as accuracy, precision, recall, F_1-score, and training time to identify the most suitable model for real-time deployment. Model interpretability is addressed through the analysis of feature importances from ensemble-based classifiers to highlight the most influential URL characteristics. To assess scalability and generalization, the top-performing model is applied to a larger corpus of 5.2 million URLs. Finally, to ensure reproducibility, we publicly release the source code, dataset splits, hyperparameter settings, and computing environment details.

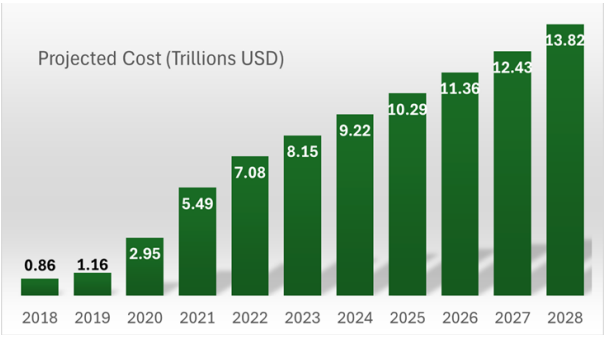

Fig. 4. Annual increase in projected cost of Phishing Attacks [1]

The remainder of this paper is organized as follows. Section III describes the materials and methods (methodology) used in our study. Section IV formulates the problem. Section V details the feature selection process. Section VI presents the machine learning algorithms. Section VII reports the experimental results. Section VIII concludes the paper and outlines directions for future work. Finally, References are listed in the end of paper.

II. Related Work

Early phishing detection efforts relied heavily on classical machine learning classifiers applied to engineered features from URLs and email bodies. Abu-Nimeh *at al.* [9] performed one of the first comparative evaluations, extracting 43 features—spanning lexical URL properties (length, special character counts), host-based attributes (IP usage, domain age), and HTML markers (form presence, hidden fields)—from a corpus of 2,889 phishing and legitimate emails. They evaluated Logistic Regression, CART, BART, Support Vector Machines, Random Forest, and a basic neural network, and demonstrated that Random Forest achieved the highest accuracy of 96.07% with relatively modest computational cost. This foundational work highlighted the power of ensemble trees in distinguishing malicious from benign content, but its reliance on full email analysis limited real-time applicability. In the literature, many studies focus on URL-based analysis for detecting phishing attacks using traditional feature extraction models. However, some works have explored content-based analysis, while others have utilized N-gram features extracted from URLs [6], [7]. Depending on the specific needs of applications, each approach offers distinct advantages.

Subsequent studies focused on URL-only pipelines to reduce preprocessing overhead. Kolla *et al.* [4] compared Decision Trees, Random Forests, Multilayer Perceptrons, SVM, and XGBoost on a dataset of 100,000 labeled URLs using a streamlined 12-feature set of lexical and statistical indicators, finding that Random Forest reached 90.00 % accuracy. Innab *et al.* [5] extended this by normalizing features and evaluating six classifiers—Decision Tree (96.4 %), Random Forest (97.0 %), Gradient Boosting (96.0 %), XGBoost (97.3 %), AdaBoost (93.4 %), and MLP (97.0 %)—before demonstrating that a Voting ensemble achieved 97.8% accuracy on their first dataset and equal performance across models on a second, larger dataset. Their results emphasize that well-tuned classical ensembles can rival more complex architectures in accuracy, latency, and false positive control.

Deep learning approaches have recently gained traction for URL classification. In *DEPHIDES: Deep Learning Based Phishing Detection System*, Sahingoz *et al.* [13]

[1]The paper also reports 100 % accuracy on a second dataset; the value shown is for the first dataset to keep results comparable.

TABLE I
COMPARISON OF PHISHING-DETECTION STUDIES BY YEAR, ALGORITHM AND ACCURACY

Study	Year	Algorithm	Acc. (%)
Abu-Nimeh *et al.* [9]	2007	Random Forest	96.07
Kolla *et al.* [11]	2022	Random Forest	90.00
Innab *et al.* [12]	2024	RF-based ensemble	97.80[1]
Sahingoz *et al.* [13]	2019	(CNN)	98.74
This work (Random Forest)	2025	Random Forest	94.82
This work (ANN + LSTM)	2025	ANN+ LSTM	94.00

curate a corpus of roughly five million labeled URLs. Each URL is converted into a sequence of 50-dimensional character-level embeddings, and five neural architectures are evaluated—fully-connected ANN, 1D-CNN, RNN, bidirectional RNN, and an attention-based network. Their experiments demonstrate that the 1D-CNN achieves the highest detection accuracy of 98.74%. While these deep models excel at capturing subtle sequential patterns and scale effectively with large datasets, they also demand substantial GPU resources and incur longer training times.

Comparative analyses across these paradigms reveal trade-offs between accuracy, resource requirements, and data efficiency. Table I summarizes the peak accuracies reported by key studies alongside those obtained in this work.

Despite these advances, several gaps remain. First, most classical studies either analyze full email content or operate on large URL corpora without considering resource constraints. Second, deep learning works often lack comparisons against optimized classical baselines under unified preprocessing and evaluation protocols. Third, reproducibility suffers from inconsistent feature pipelines and dataset splits. Our work addresses these issues by extracting a standardized set of 22 URL-based features, merging train/validation/test splits into a single 520 285-URL corpus, and performing a head-to-head evaluation of five classical and three deep models using 5-fold cross-validation on identical data and feature sets.

III. MATERIALS AND METHODS

A. Dataset Description

Within the scope of this study, a comprehensive dataset has been created and several experiments have been run on this dataset. Phishing URLs were collected from https://www.phishtank.com, and legitimate URLs were harvested from https://commoncrawl.org [13]. PhishTank is an open-source community where submitted URLs are voted into three classes: Valid Phish, Invalid Phish, and Unknown. Only URLs with sufficient votes confirming phishing are labeled Valid Phish, ensuring high accuracy of the phishing subset. URLs marked Valid Phish up to August 2018 yielded 2,320,893 phishing samples in the original DEPHIDES corpus.

In this work, we utilized the `small_dataset` version available in the DEPHIDES GitHub repository. The `small_dataset` provides pre-split train, validation, and test CSV files. We downloaded these three splits and merged them into a single CSV file to simplify our training pipeline. After merging, the small dataset contains a total of 520,285 URLs. An example view of preview of the merged URL dataset CSV file containing train, validation, and test splits for the `small_dataset` is shown in Table II, and the total number of records in the merged CSV is shown in Table III.

TABLE II
SAMPLE LABELLED URLS. PREVIEW OF THE MERGED URL DATASET CSV FILE CONTAINING TRAIN, VALIDATION, AND TEST SPLITS.

Class	URL
Legitimate	https://vds.de/fileadmin/vds_publikationen/vds_2596_schema.pdf
Phishing	http://www.bartekbitner.pl/libraries/fof/-/din7
Phishing	http://www.khoubari.com/paypal.sam
Legitimate	https://www.flipcause.com/secure/cause_pdetails/mti5ntm=
Phishing	http://www.bartekbitner.pl/libraries/fof/-/din7

TABLE III
TOTAL NUMBER OF URLS IN THE MERGED CSV AFTER COMBINING TRAIN, VALIDATION, AND TEST SPLITS.

Dataset	Class	Small Dataset	Big Dataset
Train	Phish	162,463	1,624,623
	Legitimate	201,736	2,017,363
	Overall	364,199	3,641,986
Validation	Phish	46,649	466,504
	Legitimate	57,927	579,271
	Overall	104,576	1,045,774
Test	Phish	22,978	229,766
	Legitimate	28,532	285,314
	Overall	51,510	515,080
All	Phish	232,090	2,320,893
	Legitimate	288,195	2,881,948
	Overall	520,285	5,202,841

In this study, raw URLs are transformed into numerical features through a comprehensive preprocessing pipeline designed to extract structural, statistical, and semantic indicators from each URL. These features are then used to train machine learning models for phishing detection.

B. URL Feature Extraction

The following steps are applied to each URL:

- **URL Parsing:** Using the `urlparse` module, the URL is divided into components such as hostname, path, and query parameters.
- **Length-based Features:** Lengths of the URL, host, path, and query string are computed.
- **Character Counts:** The number of specific characters (e.g., `.`, `/`, `-`, `_`, `@`, `?`, `=`, `&`) are calculated. Digit and uppercase letter counts are also included.
- **Ratios:**
  - Digit ratio: proportion of digits to total URL length.
  - Special character ratio: proportion of special characters to URL length.

TABLE IV
EXTRACTED FEATURES AND TARGET VARIABLE

Feature	Description
url_length	Total length of the full URL string
host_length	Length of the hostname
path_length	Length of the path component
query_length	Length of the query string
dot_count	Number of . characters
slash_count	Number of / characters
dash_count	Number of - characters
underscore_count	Number of _ characters
at_count	Number of @ characters
sp_q_char_count	Sum of ?, =, & occurrences
digit_count	Total number of numeric digits
upper_count	Number of uppercase letters
digit_ratio	Ratio of digits to total URL length
sp_char_ratio	Ratio of special characters to total length
url_entropy	Shannon entropy score of the URL
subdomain_count	Number of subdomains in the host
path_depth	Number of sub-paths (based on /)
query_param_count	Number of query parameters
tld_length	Length of the top-level domain
sus_word_count	Number of suspicious words found in the URL
is_ip	1 if the host is an IP address, else 0
is_https	1 if the URL uses HTTPS, else 0
class	0 = benign, 1 = phishing

- **Shannon Entropy:** A measure of character randomness, indicating complexity and potential obfuscation in the URL.
- **Structural Features:** Includes number of subdomains, depth of path, number of query parameters, and top-level domain (TLD) length.
- **Suspicious Word Count:** Count of known phishing-related keywords appearing in the URL.
- **Special Patterns:**
  - IP Address usage (is_ip).
  - HTTPS protocol presence (is_https).

C. Target Variable

The **target variable** in this study is the class label of each URL, indicating whether it is benign (legitimate) or phishing (malicious). The model learns to distinguish between these classes using the engineered numerical features.

The goal of this study is to detect phishing websites based on the structural and statistical characteristics of URLs. Phishing URLs often exhibit distinct patterns such as unusual character usage, the presence of suspicious keywords, or the absence of secure protocols. To capture these characteristics, the problem is formulated as a binary classification task.

Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be the dataset, where each x_i is a URL string and $y_i \in \{0, 1\}$ is the label indicating whether the URL is *benign* (0) or *phishing* (1).

Each raw URL x_i is passed through a feature extraction function $f(x_i)$ to produce a numerical feature vector $z_i \in \mathbb{R}^m$ composed of m engineered attributes derived from the URL (e.g., length, entropy, digit ratio).

$$z_i = f(x_i) = [z_{i1}, z_{i2}, \ldots, z_{im}]$$

The objective is to learn a function $h : \mathbb{R}^m \to \{0, 1\}$ such that the predicted class $\hat{y}_i = h(z_i)$ is as close as possible to the true label y_i. This can be achieved by minimizing a binary classification loss function (e.g., binary cross-entropy) over the dataset:

$$\mathcal{L} = -\sum_{i=1}^{n} [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

- The raw dataset only contains URL strings and class labels (no content or network metadata is used).
- URLs are assumed to be labeled correctly as either benign or phishing.
- No domain knowledge beyond URL structure and character statistics is employed.

This formulation allows the use of various supervised learning algorithms such as Random Forest, Gradient Boosting, or Neural Networks to learn the decision function $h(z)$.

Although all 22 extracted URL-based features were initially used during model training, further analysis was conducted to evaluate the relative importance of these features. The objective of feature selection is to identify the most informative attributes that contribute most significantly to the classification task, potentially reducing model complexity and improving performance in future iterations.

Three commonly used feature selection methods were applied on the scaled training data:

1) **Random Forest Feature Importance:** Feature importances were extracted from a trained Random Forest classifier based on the decrease in Gini impurity.
2) **Recursive Feature Elimination (RFE):** A logistic regression model was used to iteratively remove less important features until the top 10 remained.
3) **SelectKBest with ANOVA F-value:** Univariate statistical test (ANOVA F-test) was used to select the 10 most relevant features.

Table V summarizes the top 10 features selected by each method. Additionally, these features can be further refined using optimization techniques. Recent studies have demonstrated the effectiveness of nature-inspired feature selection models [10], which are considered as a potential direction for future work in this study.

Some of the most frequently selected features across all methods are briefly explained below:

- **is_https:** Whether the URL uses the HTTPS protocol (1 = yes, 0 = no).
- **suspicious_word_count:** Number of phishing-related keywords present in the URL string.
- **host_length:** Length of the domain name.
- **dot_count:** Number of periods (dots) in the URL.
- **digit_count:** Total number of digits in the URL.
- **url_entropy:** Shannon entropy score of the entire URL string.

TABLE V
TOP 10 FEATURES SELECTED BY EACH ALGORITHM

#	Random Forest	RFE (LogReg)	SelectKBest (ANOVA)
1	is_https	url_length	host_length
2	upper_count	host_length	dot_count
3	suspicious _word_count	dot_count	dash_count
4	host_length	dash_count	digit_count
5	url_entropy	digit_count	digit_ratio
6	dash_count	upper_count	special_char_ratio
7	dot_count	subdomain_count	url_entropy
8	special_char_ratio	suspicious_word _count	subdomain_count
9	path_length	is_ip	suspicious_word _count
10	digit_count	is_https	is_https

TABLE VI
HYPER-PARAMETER VALUES EXPLORED AND SELECTED

Model	Parameter	Value / Range
LR	max_iter	1000
	C	1.0
	solver	lbfgs
KNN	n_neighbors	5
	distance metric	Euclidean
DT	criterion	gini
	max_depth	None
	min_samples_split	2
SVM	kernel	rbf
	C	1.0
	gamma	scale
RF	n_estimators	{100, 200}
	max_depth	{None, 10, 20}
	min_samples_split	{2, 5}
	min_samples_leaf	{1, 2}
	CV / n_jobs	3-fold / –1
RF (best)	n_estimators	200
	max_depth	None
	min_samples_split	2
	min_samples_leaf	2

- **special_char_ratio:** Proportion of special characters relative to URL length.
- **subdomain_count:** Number of subdomains in the host (e.g., `mail.google.com` → 1 subdomain).

Although the full feature set was used in model training for this study, these selected features can guide future work by enabling the construction of lightweight models or facilitating feature reduction for embedded systems or real-time environments.

IV. MACHINE LEARNING ALGORITHMS

We evaluated five classical machine learning classifiers on the URL feature matrix described in ongoing sections:

This study evaluates five classical machine learning algorithms for phishing detection. Logistic Regression (LR) is a linear model that estimates the probability of a class using the logistic (sigmoid) function applied to a weighted sum of features. K-Nearest Neighbors (KNN) is a non-parametric, instance-based method that classifies each data point by taking a majority vote among its k nearest neighbors in the feature space, typically using Euclidean distance. Decision Tree (DT) constructs a hierarchical model by recursively splitting features to maximize information gain or reduce impurity, forming axis-aligned decision rules. Support Vector Machine (SVM) is a maximum-margin classifier that employs a Gaussian (RBF) kernel to learn non-linear boundaries by projecting data into a higher-dimensional space. Random Forest (RF) is an ensemble method comprising multiple decision trees trained on bootstrap samples with random feature subsets; its hyperparameters were optimized using 3-fold GridSearchCV. Table VII reports the training times and test accuracies on the held-out 20% split.

TABLE VII
CLASSICAL ML: TRAINING TIMES (SECONDS)

Model	Train Time (s)	Test Acc.
Logistic Regression	5.667	0.9130
K-Nearest Neighbors	0.018	0.9305
Decision Tree	5.739	0.9253
SVM (RBF kernel)	2447.762	0.9397
RF + GridSearchCV (best)	813.684	0.9481
ANN	714.590	0.9454
1D-CNN	1393.804	0.9362
RNN (LSTM)	1574.013	0.9444

All numerical features were standardized via `StandardScaler` prior to model fitting. Key hyperparameters in Table VI:

We implemented three sequence-based neural models in Keras (TensorFlow backend) see Table VIII operating on character-engineered feature matrices reshaped for 1D convolutions or LSTM:

TABLE VIII
DEEP-LEARNING MODEL ARCHITECTURES USED IN THIS STUDY

Model	Architecture details
ANN	**Input:** dense feature vector (23 dims). Dense(256, ReLU) → BatchNorm → Dropout(0.4) Dense(128, ReLU) → BatchNorm → Dropout(0.3) Dense(64, ReLU) → Dropout(0.2) Dense(1, Sigmoid)
1D-CNN	**Input:** $(23, 1)$ feature sequence. Conv1D(128, k=3, ReLU, same) → BatchNorm → Dropout(0.3) Conv1D(64, k=3, ReLU, same) → BatchNorm → Dropout(0.3) GlobalAveragePooling1D → Dense(64, ReLU) → Dropout(0.2) Dense(1, Sigmoid)
RNN (LSTM)	**Input:** $(23, 1)$ feature sequence. LSTM(128, return_seq=True) → Dropout(0.4) LSTM(64) → Dropout(0.3) Dense(32, ReLU) → Dropout(0.2) Dense(1, Sigmoid)

All models used the `adam` optimizer and `binary_crossentropy` loss.

A. Model Evaluation Strategy

Using only the ten URL attributes listed in Table IX, we retrained every model and observed a sizeable speed-up with only a minor accuracy trade-off (Table X). Classical learners benefited the most: Logistic Regression and Decision Tree finished in roughly 2.6 s (a 60 % reduction), while the grid-searched Random Forest dropped from 14 minutes to 12 minutes. Even the computationally expensive RBF–SVM shaved eight minutes off its runtime. Deep networks still required longer GPU sessions, but their end-to-end training shrank by 600–900 seconds. In spite of these shorter runs, accuracy dipped by only 0.5–1.5 percentage points—the best Random Forest still scores 93.2 %, and the LSTM reaches 92.8 %. This balance of speed and performance suggests that the lightweight feature subset is well suited to rapid, on-device model refresh cycles or edge deployments where compute and power budgets are tight.

TABLE IX
TOP-10 URL FEATURES CHOSEN BY ≥ 2 SELECTION METHODS

#	Feature	#	Feature
1	is_https	6	dash_count
2	suspicious_word_count	7	dot_count
3	host_length	8	special_char_ratio
4	digit_count	9	url_entropy
5	upper_count	10	subdomain_count

TABLE X
TRAINING TIMES AND TEST ACCURACIES WITH THE 10-FEATURE SUBSET

Model	**Train Time (s)**	**Accuracy**
Classical ML		
Logistic Regression	2.559	0.8941
K-Nearest Neighbours	0.861	0.9154
Decision Tree	2.567	0.9059
SVM (RBF)	1953.918	0.9195
Random Forest (Grid)	709.010	0.9321
Deep Learning		
ANN	838.667	0.9263
1D-CNN	1299.904	0.9231
RNN (LSTM)	2200.996	0.9282

In this work we employ *5-fold cross-validation*: the data are split into five disjoint subsets. Each fold in turn serves as the validation set while the other four folds are used for training. This yields five estimates of each performance metric, which are then averaged to give a robust measure of model performance under different train/test partitions.

As summarised in Table VII, the *classical* learners train extremely quickly. K-Nearest Neighbours completes in just 0.02 s, Logistic Regression and Decision Tree converge in roughly 6 s, and even the grid-searched Random Forest finishes in under 14 min. The only exception is the RBF–SVM, whose 2448 s approximately 41 minutes runtime reflects the quadratic cost of kernel-matrix construction.

By contrast, the deep models require one to two orders of magnitude more time: our compact ANN trains in about 12 min, whereas the 1D-CNN and LSTM stretch to 23 –26min. In practice, this means classical ensembles can be refreshed on commodity CPUs in near–real time, while deep networks demand GPU-accelerated sessions for only a marginal (1–2 pp) accuracy gain. Such latency differences are critical for organisations that must retrain filters daily or deploy models at the edge under strict power budgets.

B. Overall Model Performance

We first assess each model's robustness via 5-fold cross-validation on the small dataset. Tables XI and XII report the mean ± standard deviation of key metrics across the five folds. Random Forest, RNN (LSTM), and ANN achieve the highest accuracies, with RF slightly leading.

TABLE XI
5-FOLD CV RESULTS FOR CLASSICAL ML MODELS

Model	**Accuracy %**	**Precision %**	**Recall %**	**F_1-Score %**
LR	91.28 ± 0.06	89.95 ± 0.15	90.57 ± 0.09	90.26 ± 0.06
KNN	91.03 ± 0.09	91.18 ± 0.14	88.45 ± 0.11	89.80 ± 0.11
DT	92.50 ± 0.04	91.72 ± 0007	91.43 ± 0.07	91.58 ± 0.04
SVM	91.50 ± 0.08	88.71 ± 0.10	92.75 ± 0.10	90.69 ± 0.09
RF	**94.82 ± 0.11**	**94.04 ± 0.18**	**94.37 ± 0.10**	**94.20 ± 0.12**

TABLE XII
5-FOLD CV RESULTS FOR DEEP LEARNING MODELS

Model	**Accuracy %**	**Precision %**	**Recall %**	**F_1-Score %**
ANN	94.47 ± 0.12	93.30 ± 0.87	94.39 ± 0.78	93.84 ± 0.10
1D-CNN	93.19 ± 0.16	92.16 ± 0.88	92.64 ± 1.01	92.39 ± 0.18
RNN	**94.59 ± 0.07**	93.28 ± 0.22	**94.68 ± 0.30**	**93.98 ± 0.08**

Next, we measure *inference latency* on a single, randomly chosen test record (not cross-validation) by averaging 1,000 repeated predictions. Results are given in Table XIII.

TABLE XIII
AVERAGE SINGLE-RECORD INFERENCE TIME ON THE HELD-OUT TEST SET (N=1000 RUNS)

Model	**Time (ms)**
Logistic Regression	134
K-Nearest Neighbors	5.516
Decision Tree	0.139
SVM (RBF)	4.106
Random Forest	11.626
ANN	77.666
1D-CNN	74.219
RNN	73.285

Overall, in the deep learning models *RNN (LSTM)* models deliver the best balance of accuracy and inference speed. The ANN is close in accuracy, but incurs slightly higher

latency. Classical ML methods such as Logistic Regression and Decision Tree are extremely fast at prediction (0.0001 s) but attain lower peak accuracy.

The confusion matrices for the RNN (LSTM) is depicted in Figure 5, illustrating their error patterns on the held-out test set.

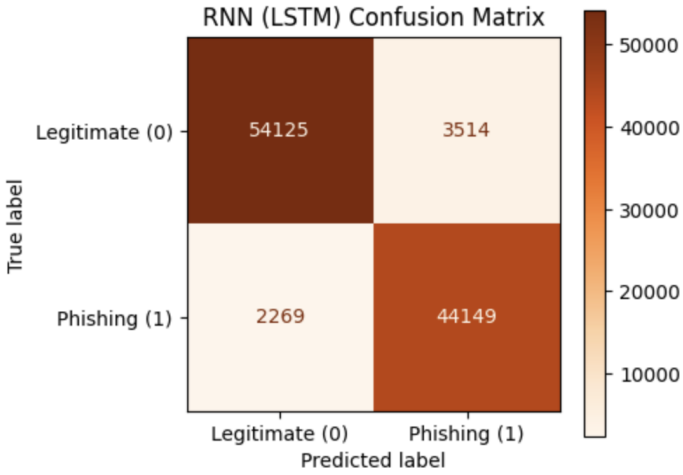

Fig. 5. Confusion matrix of RNN (LSTM) on the held-out test set.

V. Conclusion and Future Work

In this study, we presented and thoroughly evaluated a URL-only phishing detection framework leveraging both classical machine learning (ML) and deep learning (DL) models on a large-scale dataset comprising over 5 million URLs. Our results demonstrated that the Random Forest classifier achieved the highest accuracy among classical methods (94.82%), while deep learning models such as LSTM-based RNNs and ANNs also performed competitively, confirming the effectiveness of both paradigms. The framework showcased a fully language-agnostic preprocessing pipeline, robust generalization across multiple folds, and inference speeds well within real-time constraints, making it viable for practical deployment. Furthermore, error analysis indicated balanced false positive and negative rates, reinforcing its applicability in enterprise-level phishing defense systems.

Looking ahead, our findings open several promising directions for future research and system enhancements. These include integrating multi-modal features (e.g., webpage visuals, email headers), employing adaptive or federated learning for dynamic model updates, and improving explainability through interpretable AI techniques. Lightweight edge-compatible models and hybrid ensemble strategies also hold potential for improving coverage and efficiency across varied deployment environments. Expanding evaluations to cross-language scenarios and contributing to standardized benchmarking efforts will further ensure broader applicability and reproducibility. By pursuing these avenues, future phishing detection systems can become more accurate, responsive, and resilient to evolving attack strategies.

References

[1] Bright Defense, “Cybercrime Statistics: 2024 Global Trends and Key Insights,” Bright Defense, 2024. [Online]. Available: https://www.brightdefense.com/resources/cybercrime-statistics/

[2] Statista, “Number of phishing attacks worldwide from 2013 to 2024,” [Online]. Available: https://www.statista.com/statistics/266155/number-of-phishing-attacks-worldwide/.

[3] R. Saylam and O. K. Sahingoz, ”Process mining in business process management: Concepts and challenges,” 2013 International Conference on Electronics, Computer and Computation (ICECCO), Ankara, Turkey, 2013, pp. 131-134, doi: 10.1109/ICECCO.2013.6718246.

[4] E. C. Bayazit, O. K. Sahingoz and B. Dogan, ”A Deep Learning Based Android Malware Detection System with Static Analysis,” 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-6, doi: 10.1109/HORA55278.2022.9800057.

[5] Hoxhunt, Phishing Trends Report (Updated for 2025), Hoxhunt, 2025. [Online]. Available: https://hoxhunt.com/guide/phishing-trends-report. [Accessed: May 8, 2025].

[6] U. Ozker and O. K. Sahingoz, ”Content Based Phishing Detection with Machine Learning,” 2020 International Conference on Electrical Engineering (ICEE), Istanbul, Turkey, 2020, pp. 1-6, doi: 10.1109/ICEE49691.2020.9249892.

[7] M. Korkmaz, E. Kocyigit, O. K. Sahingoz and B. Diri, ”Phishing Web Page Detection Using N-gram Features Extracted From URLs,” 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2021, pp. 1-6, doi: 10.1109/HORA52670.2021.9461378.

[8] S. Parekh, D. Parikh, S. Kotak, and S. Sankhe, ”A new method for detection of phishing websites: URL detection,” in Proc. 2018 2nd Int. Conf. Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, Apr. 2018, pp. 949–952.

[9] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, ”A comparison of machine learning techniques for phishing detection,” in Proc. 2nd Annual eCrime Researchers Summit, Anti-Phishing Working Group, Pittsburgh, PA, USA, Oct. 2007, pp. 60–69.

[10] E. Kocyigit, M. Korkmaz, O. K. Sahingoz, and B. Diri, ”Enhanced Feature Selection Using Genetic Algorithm for Machine-Learning-Based Phishing URL Detection,” Applied Sciences, vol. 14, no. 14, p. 6081, 2024, doi: 10.3390/app14146081

[11] J. Kolla, S. Praneeth, M. S. Baig, and G. R. Karri, ”A comparison study of machine learning techniques for phishing detection,” Journal of Business and Information Systems, vol. 4, no. 1, pp. 21–33, 2022.

[12] N. Innab, A. A. F. Osman, M. A. M. Ataelfadiel, M. Abu-Zanona, B. M. Elzaghmouri, F. H. Zawaideh, and M. F. Alawneh, ”Phishing Attacks Detection Using Ensemble Machine Learning Algorithms,” Computers, Materials & Continua, vol. 80, no. 1, pp. 1–15, 2024.

[13] O. K. Sahingoz, E. BUBEr and E. Kugu, ”DEPHIDES: Deep Learning Based Phishing Detection System,” in IEEE Access, vol. 12, pp. 8052-8070, 2024, doi: 10.1109/ACCESS.2024.3352629.