International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 10 Issue 4, Jul-Aug 2026 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Real-Time Malicious URL Detection Using an Attention-Based CNN-RF Hybrid Fusion Model

Sagar Sitaula[1], Prof. Raj Kumar Thakur[2], Prof. Dr. Gopal Prasad Sharma[3], Drona Prasad Acharya[4]

[1]Independent Researcher, Purbanchal University School of Science & Technology (PUSAT), Biratnagar, Nepal
[2,3]Professor, Purbanchal University School of Science & Technology (PUSAT), Biratnagar, Nepal
[4]Independent Researcher, Biratnagar, Nepal

ABSTRACT

Web-based attacks, such as phishing, malware distribution, credential stealing, and phishing redirections, still rely on malicious URLs. Traditional blacklist-based and rule-based approaches are ineffective in detecting emerging, ephemeral and camouflaged URLs because they rely on historical knowledge of known threats or user-defined patterns. A hybrid fusion approach based on a CNN–RF with attention mechanism is for binary URL classification, which fuses character-level lexical feature learning and structured URL feature extraction. A URL is described in two parallel streams: a sequence of characters on a lexical level represented by a Convolutional Neural Network (CNN), and an engineered structured feature vector for Random Forest (RF) baseline classification and neural feature fusion. The CNN stream extracts discriminative character patterns from URL strings, while the structured-feature stream encodes explicit statistical, lexical, protocol- and domain-oriented features, such as URL length, special-character count, numeric-character count, character-distribution entropy, top-level-domain encoding, subdomain count, phishing-indicator terms, and protocol-safety behavior. A fusion mechanism with attention functions fuses the lexical and structured representations, allowing the final classifier to learn from both implicit (character-level) and explicit (URL features) patterns. We implemented and tested the framework on a balanced 60,000-URL set extracted from a Kaggle malicious-and-benign URL dataset using a stratified hold-out evaluation strategy. The accuracy, precision, recall, F1-score, and ROC-AUC were used to measure the performance. The fusion model outperformed the standalone CNN and RF models, achieving 99.90% accuracy, 99.80% precision, 100% recall, 99.90% F1-score, and 99.95% ROC-AUC values under the same experimental conditions. To facilitate real-time URL inference, the trained model was also integrated with an API service based on Flask to enable a service-oriented URL-prediction workflow.

How to cite this paper: Sagar Sitaula | Prof. Raj Kumar Thakur | Prof. Dr. Gopal Prasad Sharma | Drona Prasad Acharya "Real-Time Malicious URL Detection Using an Attention-Based CNN-RF Hybrid Fusion Model" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-10 | Issue-4, August 2026, pp.1224-1240, URL: www.ijtsrd.com/papers/ijtsrd142167.pdf

IJTSRD142167

KEYWORDS: *Attention mechanism, Convolutional Neural Network, cybersecurity, feature engineering, Flask API, hybrid learning, malicious URL detection, phishing detection, Random Forest, real-time URL classification, URL security.*

I. INTRODUCTION

The growth of online services has established URLs as a central resource for communication, commerce, education, banking, cloud computing and social media. This has also resulted in URL-based attacks being a consistent cybersecurity concern. Adversaries commonly exploit malicious URLs to perform phishing, malware delivery, credential stealing, web defacement, redirecting and other web-based attacks, by concealing malicious destinations in the guise of legitimate content [1], [2], [3]. They are spread via email, social media, adverts, search engines and infected websites to deliver attacks to users without the need to invade the target systems.

Static blacklist, signature, and heuristic URL filtering techniques are effective in dealing with known threats, but are reactive and thus ineffective against newly generated, ephemeral and obfuscated malicious URLs. Adversaries often alter domain names, URL paths, subdomains, protocol, suspicious keywords and delimiter sequences to bypass static matching and rule-based filtering [2], [4]. Because new or modified

URLs may not be present in reputation databases, malicious URL detection models must learn to predict malicious intent from URL structure and learned URL patterns, rather than just known malicious features [5].

Machine learning methodologies overcome this issue by modelling URLs using lexical, host, statistical and content features, and then applying classification models such as Random Forest, Decision Tree, Support Vector Machine and XGBoost[3], [6]. Ensemble methods are still relevant since they can efficiently model structured tabular features and offer robust classification performance given discriminative manual features. But feature-based models are heavily reliant on the design of manual features and may miss subtle character-level patterns, adversarial token replacement and obfuscation when only structured features are used as the sole representation [7], [8].

Deep learning-based URL detectors alleviate reliance on feature engineering for URL detection by learning discriminative representations. CNN-based approaches have been shown to effectively capture local lexical patterns in character or token sequences [1], [4], and CNN-LSTM, CNN-BiLSTM models have been applied to capture sequential information in phishing and malicious URL detection [9], [10], [11]. Recent graph-attentive and transformer-based models also demonstrate that URL detection can benefit from more sophisticated representation learning, such as structural, semantic, behavioral, and multi-view representations [12], [13]. However, models using only deep lexical representations might be under-exploiting explicit structured features such as URL length, special characters, numeric score, entropy, subdomain depth, top-level-domain type, phishing-indicator terms and protocol-safety behavior.

Recent hybrid and multi-view research suggests that multi-evidence URL detection may enhance detection performance. Hybrid systems combine rule-based domain indicators with machine learning predictions [14], feature-based ensemble learners harness different lexical and host-based features [3], and ensemble techniques with sampling methods tackle class imbalance problems in malicious URL detection [15]. Similarly, systematic reviews highlight that feature engineering, interpretability, scalability and deploy ability are still crucial for phishing and malicious URL detection systems [7], [8]. Web-based approaches like ABHI SHIELD and Safe URL AI Extension also highlight the need for efficient real-time screening pipelines that are closer to user interaction [16], [17].

Despite such progress, there are three key challenges for deployable malicious URL classification systems. First, feature-based models rely on hand-selected URL features, and may overlook latent character-level evidence in obfuscated URLs. Second, sequence-based deep models tend to focus on the raw lexical representation, while disregarding explicit structured clues that are cheap to compute and relevant to security. Third, some state-of-the-art models are primarily reported as offline URL classifiers without much consideration of an inference workflow that maintains the same preprocessing and feature extraction as the training stage.

We propose an attention-based CNN–Random Forest hybrid model for binary URL detection. The CNN stream learns character-level lexical representations from a series of normalized and padded URLs while the structured-feature stream represents engineered URL attributes used for Random Forest-based baseline classification and neural fusion. These features include URL length, number of special characters, number of numeric characters, character-distribution entropy, top-level-domain encoding, number of subdomains, phishing-indicator keywords, and protocol-safety behavior. The lexical and structural representations are combined via an attention-based fusion model, enabling an integrated decision-making model that incorporates implicit (character-based) and explicit (URL-attribute-based) evidence.

The model is trained and tested on a balanced 60,000-URL subset of the publicly available Kaggle "Malicious And Benign URLs" data set [18]. This data includes an equal number of benign and malicious URLs and is tested using a stratified 80:20 hold-out approach. We report the accuracy, precision, recall, F1-score and ROC-AUC metrics. Our fusion model is benchmarked against the standalone CNN and Random Forest models with identical implementation. The learned model is also provided as a Flask API for service-oriented URL inference.

The main contributions of this work are:

1. A novel attention-based hybrid CNN-RF framework is for malicious URL detection using character-based lexical learning and structured feature-based classification.
2. A two-representation-based preprocessing method is adopted that integrates character-based URL tokenisation with statistical, lexical, protocol, and domain-specific features of URLs.
3. The model is tested on a Kaggle-based 60,000-sample balanced dataset using common classification metrics and an internal benchmark

based on the individual CNN and Random Forest branches.

4. We also build a Flask inference model to embed the trained framework into a service-oriented prediction pipeline for URL classification.

The rest of this paper is structured as follows. Section II discusses the prior studies on phishing and malicious URL detection. Section III introduces the dataset, preprocessing, feature engineering, CNN branch, Random Forest branch and attention-based fusion model. Section IV provides the experimental results and discussion. Section V outlines the real-time deployment process. Section VI concludes the paper and provides future work.

II. RELATED WORK

Research on malicious URL detection has progressed from static filtering mechanisms toward adaptive data-driven models capable of identifying newly generated and obfuscated threats. Initial systems used to defend against URLs used primarily blacklists, whitelists, signature matching and manually defined heuristic systems. These algorithms are still computationally efficient when the malicious domain has previously been reported, but their reliance on indicators that are known limits their performance against temporary malicious domains that are dynamically configured and have never been documented before. Review research recognizes this reactive type of design as the fundamental vulnerability of the static URL filtering [2]. Similar limitations are discussed in BERT-CNN-based malicious URL detection, where morphing, obfuscation, and dynamic URL generation are described as common evasion strategies against blacklist-based systems [4].

Machine learning methods were introduced to overcome the rigidity of static matching by classifying URLs through extracted lexical, statistical, host-based, and content-oriented features. Feature-based models commonly represent URLs using attributes such as URL length, domain characteristics, special-character distribution, protocol behavior, keyword patterns, and measurable structural indicators. Velpula et al. evaluated malicious URL detection using structured URL features, while Alzubi et al. demonstrated the relevance of lexical, host-based, and content-based attributes for phishing URL classification with classifiers such as Random Forest, Decision Tree, SVM, and XGBoost[3], [19]. XGBoost-based early phishing detection also shows that optimized classical classifiers remain competitive when discriminative feature representations are available [20]. However, feature-based models remain dependent on manually designed attributes and may miss hidden character-level or contextual patterns when URL representation is restricted to structured features alone.

Ensemble learning remains prominent in malicious URL detection because it improves classification stability by combining multiple learners. Random Forest approaches are of particular interest as they capture nonlinear interactions among structured URL features and keep the inference process efficient. Combining deep convolutional-based feature extraction with an ensemble-based training showing the worth of matching learned representation with ensemble-based classification [6]. Sujon et al. also investigated ensemble classifiers using ADASyn-based oversampling to overcome the issue of class imbalance in malicious URL detection reporting better performance across XGBoost, Random forest and Decision tree models [15]. These results affirm the further application of ensemble learning where structured URL features, and unbalanced datasets are used.

Deep learning techniques lessen the reliance on manual feature engineering, by learning an expression of URL strings. Convolutional filter CNN-based models are popular to detect URLs due to their ability to capture local character patterns, suspicious token fragments, and lexical anomalies inherent in raw URLs. Haqu et al. indicated high detection rates of phishing URLs with a deep learning model based on URL representation learning [1]. Lamrabti et al. added URL_trigger as a real-time deep learning application to malicious URL identification, which further supports the appropriateness of the neural URL models to automated screening procedures [5]. Adebowale et al. also illustrated how deep learning algorithms can be used to detect intelligent phishing, which also suggests the broader relevance of neural models to cybersecurity classification tasks [21].

Sequential and hybrid deep learning models extend CNN-based detection by capturing ordering relationships and longer dependencies in URL strings. Gurung et al. combined CNN-LSTM with Random Forest for phishing URL detection, demonstrating the benefit of integrating deep sequence modeling with classical classification [9]. CNN-BiLSTM-based phishing webpage detection further shows that combining convolutional extraction with bidirectional recurrent modeling improves sequence-aware phishing detection [10]. Ujah-Ogbuagu et al. a hybrid CNN-LSTM model for spoofing website URL detection in real-time applications, reporting stronger performance than standalone CNN and LSTM models [11]. These studies show that lexical URL sequences contain discriminative local and temporal patterns,

although deep sequence models may still underuse explicit structured URL indicators.

Recent work has introduced contextual, attention-based, graph-based, and transformer-based representations to improve malicious URL and phishing detection. Jin et al. a BERT-CNN method that combines contextual language modeling with convolutional feature extraction to address weak generalization and manual feature-selection limitations [4]. Hossain et al. developed a graph-attentive LSTM model that represents URL information through graph and sequence structures by combining GNN, GAT, and LSTM components [12]. Wang et al. a lightweight multi-view Transformer with a mixture-of-experts mechanism, integrating URL, attribute, content, and behavioral views to improve robustness against concealed phishing strategies and changing URL patterns [13]. These methods show the importance of richer representation learning, but may add extra architectural complexity to the lightweight URL-only systems.

Hybrid frameworks have been the focus of attention due to the exposure through the multiple complementary signals instead of a single representation type posed by malicious URLs. Swetha et al. a hybrid real-time malicious URL detector architecture using SOM-RMO and RBFN with Tabu Search, represent optimization-based hybrid learning of URLs classification [22]. Jadhav and Chandre have introduced machine learning with heuristic rules across multi-domain phishing features demonstrating that layered feature analysis and rule-guided classification can be effective in practice phishing detection [14]. Paithane et al. reviewed the hybrid machine learning methods and stated that model combination was a significant direction taken toward enhancing the reliability of the phishing URL detection by the model [8]. Lingaiah and Pandey designed a deep learning framework to detect phishing URLs with the help of feature engineering, correlation analysis, and neural classification [23].

The importance of feature engineering, explainability, and deployment feasibility is still relevant in real-world phishing and malicious URL detection systems. According to Alsuqayh et al., feature engineering and explainable AI are the main factors that contribute to the improvement of accuracy, interpretability, and trust in phishing web detection systems [7]. There are practically real-life applications of lightweight real-time screening within close proximity to the point of user interaction [16], [17]. In addition to systems specific to URLs, DanielRaj et al. describe a flask-based real-time cybersecurity dashboard that uses machine learning, reinforcement learning, and explainability to adaptive threat detection and response [24]. The above works suggest that deployable cybersecurity models must not only be highly classified but also exhibit consistent inference pipelines, and interpretable outputs.

Despite these advances, several gaps remain relevant to deployment-oriented malicious URL classification. Feature-based classifiers model structured URL attributes efficiently but remain dependent on manually selected indicators. Deep lexical models learn discriminative URL patterns directly from raw strings but may underuse explicit statistical and host-oriented cues such as entropy, subdomain count, phishing-indicator terms, domain type, numeric-character count, and protocol-safety behavior. Several high-performing methods also rely on complex architectures or additional webpage-level information, which may limit lightweight deployment when only URL strings and derived URL features are available. Comparative reporting across published studies is also affected by differences in dataset source, class distribution, preprocessing strategy, and evaluation protocol.

The framework follows the hybrid detection direction by combining a character-level CNN stream for lexical representation learning with a structured-feature stream used for Random Forest-based baseline classification and neural fusion input. The two representations are integrated through an attention-guided fusion mechanism. This design differs from purely feature-based methods by learning URL character patterns directly, and it differs from purely deep lexical models by preserving explicit structured URL indicators. The framework also remains deployment-oriented by packaging the trained system through a Flask-based inference API, consistent with the real-time and browser-oriented detection direction reported in recent cybersecurity studies [16], [17], [24].

III. METHODOLOGY

Our approach is an attention-based fusion of CNN and RF binary malicious URL detection. This approach is motivated by the fact that malicious URLs present risk by providing two types of evidence. One view is the lexical structure of the URL at the character level, in which suspicious substrings, deceptive keywords, anomalous delimiters, numeric-symbolic patterns and other forms of obfuscation may be present. The second is the URL representation, which explicitly represents statistical, lexical, protocol and domain-specific features. Rather than sticking to a single representation, the approach feeds both perspectives to an architecture that comprises two learning streams

and an attention-driven fusion mechanism for final classification.

The overall architecture is depicted in Fig. 1. The lexical CNN stream captures lexical patterns from the sequence of the normalized and padded URL characters, and the structured feature stream captures engineered representation of the URL attributes that are also used in the Random Forest baseline. The Random Forest (RF) model is trained and calibrated as the structured-feature baseline. In the fusion model, the scaled structured feature matrix is also fed into the dense neural stream so that engineered URL attributes can be combined with the CNN stream's lexical representation via attention. This approach retains the role of the Random Forest model for comparison, while enabling differentiable neural fusion of lexical and structural features.

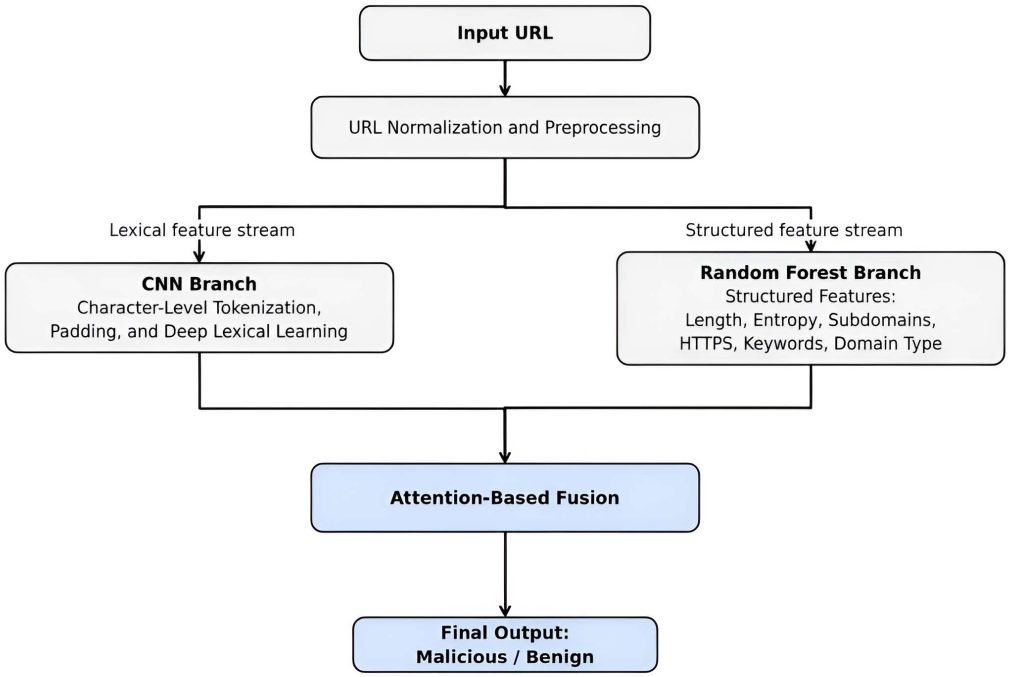

Fig. 1. Attention-based CNN–RF hybrid fusion architecture. Source: Author's own work, adapted from [25].

A. Problem Formulation

Let a URL instance be represented as u_i and let its binary label be denoted by $y_i \in \{0,1\}$, where $y_i = 1$, indicates a malicious URL and $y_i = 0$,indicates a benign URL. Each URL is represented through two inputs: a character-level lexical sequence x_i,and a structured feature vector z_i. The classification task is formulated as learning a function

$$\hat{y}_i = F_\theta(x_i, z_i)$$

where F_θ denotes the hybrid detection model and $\hat{y}_i$ is the predicted class label. The model estimates the posterior probability

$$P(y_i = 1 \mid x_i, z_i)$$

and assigns the final class label using a probability threshold of 0.5.

B. Dataset Source and Label Preparation

The experimental dataset was derived from the public Kaggle "Malicious And Benign URLs" dataset [18]. The implementation is done using a balanced working subset of 60,000 URLs, comprising of 30,000 benign and 30,000 malicious samples. Balanced subset can also be used in order to decrease dominance of classes in model learning and direct internal comparison of standalone CNN model, standalone Random Forest model and fusion model under the same evaluation environment.

The processed file that was utilized in the implementation is *balanced-datasets60k.csv*. The data set includes the records of URLs and their labels. The label values are coded to binary values with the malicious URLs coded as 1 and benign URLs coded as 0. Preprocessing is done to normalize URL strings and structured features are extracted out of URL field using the implemented feature extraction pipeline.

C. Data Partitioning Strategy

The dataset is partitioned using stratified hold-out sampling. The implementation applies Stratified Shuffle Split with one split, a test size of 0.2, and a random state of 42. This produces an 80:20 training–testing division while preserving the benign–malicious class distribution in both partitions. During neural model training, 20% of the training partition is further used as a validation split. This protocol provides a consistent evaluation setting for the standalone CNN, standalone Random Forest, and fusion model.

D. Character-Level URL Representation

The lexical stream represents each URL as a character-level sequence. Character-level modeling is suitable for malicious URL detection because attack evidence often appears in short substrings, unusual character arrangements, excessive delimiters, numeric substitutions, and obfuscated fragments. For the CNN branch, the URL is normalized by removing the protocol prefix where applicable, converting the remaining string to lowercase, tokenizing it at the character level, and padding the resulting sequence to a fixed maximum length of

$$L = 200.$$

For a URL u_i, the padded character sequence is represented as

$$x_i = [c_1, c_2, \ldots, c_L]$$

where each c_j is the integer index assigned by the fitted character tokenizer. URLs shorter than 200 characters are padded, while longer URLs are truncated according to the sequence-padding routine used in the implementation. This representation enables the CNN branch to learn discriminative lexical patterns directly from URL strings without relying only on manually engineered features.

E. Structured Feature Engineering

In parallel with character-level representation, each URL is transformed into a structured feature vector z_i. These engineered attributes capture explicit URL properties that provide complementary evidence to the CNN lexical representation. The implemented structured features are summarized in Table I

Table I. Structured features used in the structured-feature stream. Source: Author's own work, adapted from [25].

Feature	Description	Type	Encoding
URL length	Total number of characters in the URL	Numeric	Direct
Special-character count	Count of /, ?, &, and = characters	Numeric	Direct
Numeric-character count	Number of digits in the URL	Numeric	Direct
URL entropy	Entropy computed from the URL character distribution	Numeric	Direct
Domain type	Top-level domain extracted from the URL hostname	Categorical	One-hot
Subdomain count	Number of subdomain components derived from the hostname	Numeric	Direct
Phishing indicator	Presence of high-risk phishing terms such as verify, update, secure, login, free, or account	Binary	Direct
Protocol-safety indicator	HTTPS/HTTP-related behavior, including deployment-side handling for HTTP URLs containing high-risk phishing terms	Numeric	Direct / scaled

The URL length feature is computed as

$$f_{\text{len}}(u) = |u|$$

For the implemented special-character set $S = \{/, ?, \&, =\}$, the special-character count is computed as

$$f_{\text{sp}}(u) = \sum_{c \in u} \mathbb{I}(c \in S).$$

The numeric-character count is computed as

$$f_{\text{num}}(u) = \sum_{c \in u} \mathbb{I}(c \in \{0,1,\ldots,9\})$$

The entropy feature is computed from the normalized character distribution of the URL:

$$H(u) = -\sum_j p_j \log(p_j),$$

where p_j denotes the normalized frequency of the j-th character in the URL string. The domain type is extracted from the hostname using URL parsing and represented through one-hot encoding with unknown categories ignored during transformation. The phishing-indicator feature is encoded as 1 when at least one term from the implemented keyword set appears in the URL, and 0 otherwise. The protocol-safety feature is treated as a numeric structured attribute during training and scaled with the remaining structured features. During deployment, protocol handling is strengthened by applying additional treatment to HTTP URLs that contain high-risk phishing terms.

After feature extraction, the categorical domain-type feature is separated and transformed using one-hot encoding. The original domain-type column is then removed from the numeric feature matrix. Numeric, binary, and one-hot encoded features are combined into a sparse matrix and normalized using sparse-compatible standard scaling with *Standard Scaler (with_mean=False)*.

F. CNN Stream for Lexical Feature Learning

The CNN stream receives the padded character sequence x_i as input. The sequence is first mapped into a dense vector space using an embedding layer with an embedding dimension of 50:

$$E_i = \text{Embedding}(x_i), \quad E_i \in \mathbb{R}^{200 \times 50}$$

Batch normalization is applied to the embedded representation to stabilize training. A one-dimensional convolutional layer with 128 filters, kernel size 5, ReLU activation, and L2 regularization is then used to extract local character-level patterns:

$$C_i = \text{ReLU}(\text{Conv1D}(E_i))$$

The convolutional feature maps are reduced through global max pooling, producing a fixed-length lexical vector:

$$h_{\text{cnn}} - \text{GlobalMaxPooling1D}(C_i)$$

This lexical vector is passed through a dense layer with 64 units and ReLU activation, followed by dropout with a rate of 0.5. In the standalone CNN model, the resulting representation is connected to a sigmoid output layer for binary classification. The CNN is trained using the Adam optimizer and binary cross-entropy loss, with early stopping and model checkpointing based on validation loss.

G. Random Forest Baseline and Structured-Feature Stream

The engineered structured feature matrix is used to train a Random Forest classifier as the standalone structured-feature baseline. The implementation uses

$$\text{RandomForestClassifier}(n\_estimators = 100, random\_state = 42)$$

The trained Random Forest model is calibrated using Calibrated Classifier CV to improve probability estimation. This calibrated RF model provides the structured-feature baseline and enables comparison against the standalone CNN and fusion model.

For the fusion model, the same scaled structured feature matrix is used as the second neural input stream. Since the attention layer requires compatible latent representations, the structured features are projected through dense neural transformations before fusion. The structured stream applies a dense layer with 128 units, batch normalization, a second dense layer with 64 units, and dropout with a rate of 0.3:

$$h_s = \text{Dropout}\left(\text{Dense}_{64}\left(\text{BN}\left(\text{Dense}_{128}(z_i)\right)\right)\right)$$

This produces a 64-dimensional structured representation compatible with the CNN lexical vector. Therefore, the Random Forest model is not inserted as a differentiable layer inside the Keras fusion architecture; rather, it serves as the calibrated structured-feature baseline, while the neural fusion model uses the same engineered feature space through a dense structured stream.

H. Attention-Based Fusion Mechanism

The fusion model integrates the CNN lexical representation and the structured feature representation through an attention layer. Let h_{cnn} denote the 64-dimensional CNN representation and h_s denote the 64-dimensional structured representation. Both vectors are expanded along a temporal dimension using custom tensor layers:

$$q - \text{ExpandDims}(h_{\text{cnn}})$$

$$v = \text{ExpandDims}(h_s)$$

The attention layer receives the expanded lexical representation and the expanded structured representation:

$$a = \text{Attention}(q, v)$$

The attention output is then compressed back into vector form:

$$h_a = \text{Squeeze}(a)$$

The final fusion vector is constructed by concatenating the original CNN representation, the structured representation, and the attention-derived representation:

$$h_f = [h_{\text{cnn}}; h_s; h_a]$$

The fused representation is passed to a sigmoid output layer:

$$\hat{p} = \sigma(W_f h_f + b_f)$$

where $\hat{p}$ is the predicted probability that the URL is malicious. This design allows the final classifier to use the direct lexical representation, the direct structured representation, and the attention-derived interaction representation within the same decision pipeline.

I. Training Configuration

The implemented hyperparameter configuration is summarized in Table II.

Table II. Implemented hyperparameter configuration. Source: Author's own work, adapted from [25].

Component	Setting
Maximum URL sequence length	200
Character embedding dimension	50
CNN filters	128
CNN kernel size	5
CNN dense units	64
Structured-stream dense units	128, 64
CNN dropout	0.5
Structured-stream dropout	0.3
L2 regularization	0.001
RF estimators	100
Optimizer	Adam
Loss function	Binary cross-entropy
Batch size	32
Maximum training epochs	5
Early-stopping patience	3
Validation split	0.2
Test split	0.2
Split random state	42
Classification threshold	0.5

The standalone CNN and Random Forest models are trained first to provide branch-level baselines. The fusion model is then constructed as a dual-input neural model and trained using paired inputs: padded character-level URL sequences for the CNN stream and scaled structured feature vectors for the structured stream. The trained artifacts include the CNN model, calibrated RF model, fusion model, tokenizer, one-hot encoder, and scaler.

J. Training Workflow

The training procedure followed in the implementation is summarized below.

Algorithm 1. Training procedure of the hybrid framework

Input: Balanced URL dataset $D = \{(u_i, y_i)\}_{i=1}^{N}$

Output: Trained CNN model, calibrated RF model, fusion model, tokenizer, scaler, and one-hot encoder

1. Load the processed 60,000-sample URL dataset.
2. Map class labels into binary form: malicious = 1 and benign = 0.
3. Extract structured URL features from each URL.

4. Encode the domain-type feature using one-hot encoding.
5. Combine numeric, binary, and encoded categorical features into a sparse matrix.
6. Apply sparse-compatible standard scaling to the structured feature matrix.
7. Split the dataset using stratified 80:20 hold-out partitioning.
8. Normalize URL strings for lexical modeling and tokenize them at the character level.
9. Pad each character sequence to a maximum length of 200.
10. Train the standalone CNN model using padded URL sequences.
11. Train and calibrate the Random Forest model using structured URL features.
12. Build the dual-input fusion model with CNN, structured dense, and attention components.
13. Train the fusion model using paired lexical and structured inputs.
14. Save the trained models and preprocessing artifacts.
15. Evaluate CNN, RF, and fusion models on the held-out test set.

K. Evaluation Protocol

The trained models are evaluated on the held-out test partition using accuracy, precision, recall, F1-score, and ROC-AUC. The implementation also generates classification reports, ROC curves, precision–recall curves, and confusion matrices for the standalone CNN, calibrated Random Forest, and fusion model. Binary predictions are produced using a probability threshold of 0.5.

Accuracy is computed as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is computed as

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall is computed as

$$\text{Recall} = \frac{TP}{TP + FN}$$

The F1-score is computed as

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROC-AUC measures the ability of the model to separate benign and malicious classes across classification thresholds. These metrics are used consistently for the standalone CNN, calibrated Random Forest, and attention-based fusion model. The evaluation follows the implemented inference pipeline, and binary labels are obtained by applying the 0.5 decision threshold to the generated probability scores.

L. Deployment-Oriented Model Artifacts

The trained framework is prepared for deployment by saving the neural models and preprocessing artifacts after training. The deployment pipeline loads the fusion model, CNN model, calibrated Random Forest model, tokenizer, scaler, and one-hot encoder. Incoming URLs are processed through the same two-view pipeline used during training: the URL is normalized and converted into a padded character sequence for the CNN stream, while structured features are extracted, one-hot encoded, scaled, and passed into the Random Forest and fusion inputs.

The Flask-based API returns prediction outputs from the trained components in JSON format. This deployment structure enables service-oriented URL inference and supports integration with browser-connected or web-based cybersecurity screening workflows.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The attention-based CNN–RF hybrid fusion model was evaluated on a balanced 60,000-URL dataset derived from the public Kaggle "Malicious and Benign URLs" dataset [18]. The dataset contained equal proportions of benign and malicious URLs and was divided using a stratified 80:20 hold-out protocol. The held-out test partition contained 12,000 URLs, consisting of 6,000 benign samples and 6,000 malicious samples. Model performance was measured using accuracy, precision, recall, F1-score, and ROC-AUC. ROC curves, precision–

recall curves, and confusion matrices were generated for the standalone CNN model, the standalone Random Forest model, and the fusion model.

A. Experimental Setup

The experimental configuration followed the implementation described in Section III. The CNN model was trained using character-level padded URL sequences with a maximum sequence length of 200. The Random Forest classifier was trained using the structured feature matrix derived from URL length, special-character count, numeric-character count, URL entropy, domain type, subdomain count, phishing-indicator terms, and protocol-safety information. The fusion model received two synchronized inputs: the padded character-level URL sequence for the CNN stream and the scaled structured feature vector for the structured stream.

All models were evaluated on the same held-out test partition to maintain a consistent internal comparison. The binary decision threshold was set to 0.5, where predicted probabilities greater than or equal to the threshold were classified as malicious and probabilities below the threshold were classified as benign.

B. Performance of Standalone Models

The standalone CNN model achieved high classification performance on the held-out test set, obtaining an accuracy of 0.9988, precision of 0.9876, recall of 0.9975, F1-score of 0.9987, and ROC-AUC of 0.9982. The following results indicate that the character-level CNN stream provided captures discriminative lexical evidence in URL strings, including suspicious substrings, irregular placement of symbols, numeric patterns, and other character sequences related to obfuscation.

Even the standalone Random Forest model demonstrated a good performance with the use of only structured URL attributes. It obtained an accuracy of 0.9966, precision of 0.9973, recall of 0.9958, F1-score of 0.9966, and ROC-AUC of 0.9989. The large value of ROC-AUC indicates that the designed structured feature set offered good class-separation capability. Entropy, the type of domain, the number of subdomains, phishing-indicator terms, protocol-related behavior, were features that provided explicit statistical and host-oriented evidence about the classification of malicious URLs.

The individual models had varying representational capabilities. The CNN model was trained on implicit lexical patterns directly based on URL sequences, and the Random Forest model based on explicit structured indicators. This contrast attests to the adoption of the fusion approach, which combines the acquisition of lexical knowledge at the character level and the ability to classify objects based on specifications.

C. Performance of the Fusion Model

The CNN-RF hybrid fusion model based on attention was observed to have the best overall performance of the compared models. On the held-out test set, the fusion model obtained an accuracy of 0.9990, precision of 0.9980, recall of 1.0000, F1-score of 0.9990, and ROC-AUC of 0.9995. These results show that the fusion model improved the overall metric profile by combining the CNN-derived lexical representation with the structured feature representation.

The recall value of 1.0000 shows that all the malicious URLs in the held-out test set were correctly classified by the fusion model. Recall is a critical measure in malicious URL detection since the false negative is an indication of a malicious URL being identified as a benign one. The precision of the fusion model was also at 0.9980 indicating that full malicious-class response was realized but at a low false-positive rate.

Table III. Performance metrics of the evaluated models on the held-out test set. Source: Author's own work, adapted from [25].

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
CNN	0.9988	0.9876	0.9975	0.9987	0.9982
Random Forest	0.9966	0.9973	0.9958	0.9966	0.9989
Fusion Model	0.9990	0.9980	1.0000	0.9990	0.9995

D. ROC and Precision–Recall Analysis

The values of the ROC-AUC values show that all three models reported high values of separability of benign and malicious URLs. The CNN model had a ROC-AUC of 0.9982, the Random Forest model had a ROC-AUC of 0.9989 and the fusion model had the highest ROC-AUC of 0.9995. This finding suggests that the fusion model offered the best threshold-independent discrimination between the considered configurations.

The precision–recall results further support the effectiveness of the fusion strategy. Precision recall behavior is particularly important in the case of malicious URL detection since false positives and false negatives have

varying operational implications. A false positive can cause a benign URL to be considered malicious, or a false negative can make a malicious URL evade detection. The fusion model had the best F1-score of 0.9990 which indicates the best balance between precision and recall in the reported evaluation setting.

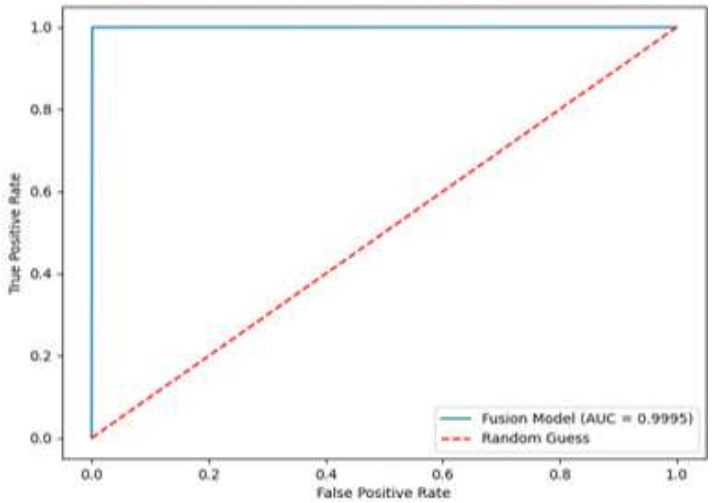

Fig. 2. ROC curve for the fusion model. Source: Author's own work, adapted from [25]

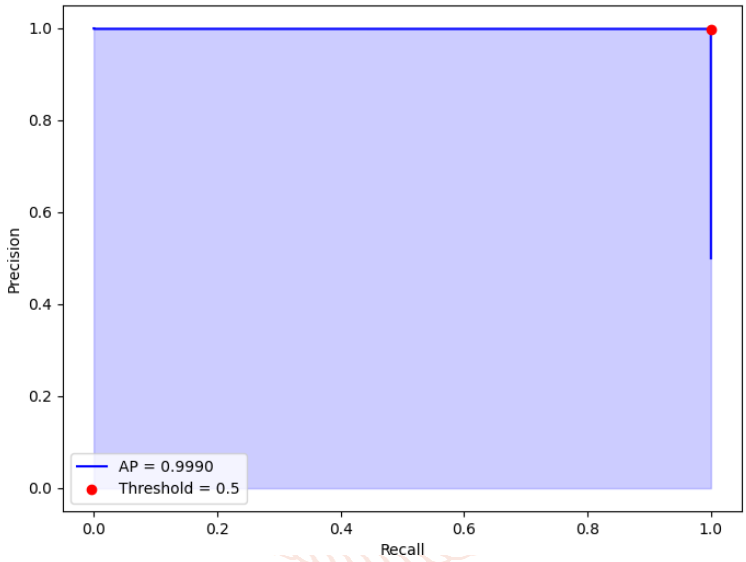

Fig. 3. Precision–recall curve for the fusion model. Source: Author's own work, adapted from [25]

E. Confusion-Matrix Analysis

The confusion matrices are used to get a class-level perspective of the prediction performance of each of the models evaluated. The confusion matrices can be directly used to look at true positives, false positives, false negatives, and true negatives since the held-out test set contained 6,000 benign URLs and 6,000 malicious URLs.

In the CNN model, the confusion matrix showed that 5,985 true positives were recorded, 0 false positives, 15 false negatives, 6,000 true negatives. This finding indicates that the CNN stream did not show any benign-to-malicious misclassification whereas 15 malicious URLs were classified as benign.

In the case of the Random Forest model, the confusion matrix obtained read 5,975 true positives, 16 false positives, 25 false negatives and 5,984 true negatives. In comparison to the CNN model, the Random Forest model yielded more false positive and false negative outcomes, which indicates that structured features alone were very discriminative but incomplete compared to the character-level lexical representation in the reported test split.

In the case of the fusion model, the confusion matrix has 6,000 true positives, 12 false positives, 0 false negatives, and 5,988 true negatives. This finding indicates that the fusion model has removed false Negatives and the number of false-positive is also low. The fact that there are no false negatives is in line with the reported value of the recall of 1.0000.

Table IV: Confusion-matrix summary of the evaluated models. Source: Author's own work, adapted from [25]

Model	TP	FP	FN	TN
CNN	5,985	0	15	6,000
Random Forest	5,975	16	25	5,984
Fusion Model	6,000	12	0	5,988

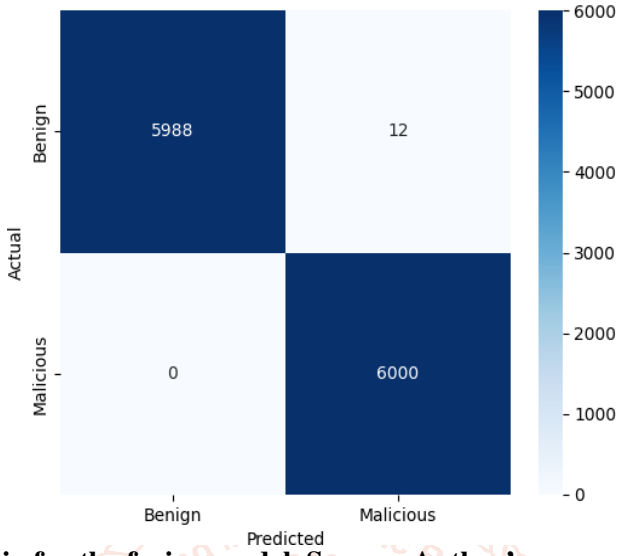

Fig. 4. Confusion matrix for the fusion model. Source: Author's own work, adapted from [25]

F. Comparative Analysis with Existing Methods

The fusion model was compared with selected malicious URL and phishing detection methods reported in related studies. The comparison includes the CNN–LSTM and Random Forest models reported by Gurung et al. [9], the hybrid SOM-RMO with RBFN approach reported by Swetha et al. [22], and the attention-based CNN–RF fusion model. Among the methods listed in Table V, the model reports the highest accuracy, precision, recall, and F1-score.

Table V. Comparative performance of the method with existing approaches

Method	Accuracy	Precision	Recall	F1-score	Remarks
CNN-LSTM [9]	94.7%	94.51%	94.59%	94.3%	Deep sequence-based hybrid model
Random Forest [9]	70.25%	67.75%	78.16%	N/A	Structured-feature baseline model
Hybrid SOM-RMO + RBFN [22]	96.5%	95.2%	94.8%	95.0%	Optimization-based hybrid model
Attention-Based CNN–RF Fusion Model	99.90%	99.80%	100%	99.90%	Attention-guided hybrid fusion model

The comparative results show that the attention-based CNN–RF fusion model is competitive with recent malicious URL and phishing detection approaches. The performance advantage is associated with the integration of two complementary representations: character-level lexical features learned through the CNN stream and structured statistical/domain-oriented features modeled through the structured stream. Unlike models that rely only on sequential URL representation or only on handcrafted attributes, the model combines both sources of evidence before final classification.

The values in Table IV are reported from different studies and reflect literature-level comparison rather than identical benchmark evaluation. Differences in dataset source, class distribution, preprocessing strategy, feature availability, model configuration, and evaluation protocol affect direct comparability across methods. Within the internal evaluation conducted in this study, the fusion model was evaluated against standalone CNN and Random Forest components under the same dataset split and preprocessing pipeline.

G. Error Analysis

As shown in the confusion-matrix results, the fusion model had the lowest error rate among the tested models on the test set. The CNN model generated 15 false negatives and 0 false positives and the Random Forest model

generated 25 false negatives and 16 false positives. The fusion model, on the other hand, produced 0 false negatives and 12 false positives. This error pattern suggests the fusion model reduces the false negative rate (improving malicious-class detection) while maintaining a low false positive rate (minimizing misclassification of benign URLs).

False negatives are the most important type of error in malicious URL classification as they erroneously allow malicious URLs to be classified as benign. The fusion model did not make any false negatives in the reported test split, meaning that all malicious URLs were detected under the adopted evaluation protocol. This finding is in agreement with the model's recall of 1.0000.

False positives corresponded with legitimate URLs being detected as malicious. This could occur when benign URLs contain lexical features often used by phishing URLs, such as keywords like login, secure, account, verify, or update. These words are commonly used in legitimate URLs for secure and banking transactions, account management and e-commerce, but also by phishing URLs that mimic legitimate services. The fusion model overcame the model's reliance on single suspicious tokens by fusing character-based lexical representation with URL structure.

The failure case analysis of the non-fusion models highlights the need for representation fusion. The CNN model captured deep lexical features well but failed to detect 15 malicious URLs. The Random Forest model learned explicit structured patterns, but made more errors (false positives and negatives) than the fusion model. The model combined the two representations through attention-driven mixing, resulting in the best error pattern among all configurations. This result suggests that fusion of lexical and structured representation boosts the reliability of malicious URL classification.

H. Discussion

The empirical results show that the attention-based CNN and Random Forest fusion model outperformed the other configurations. The CNN stream delivered strong character-level lexical training while the Random Forest-fitted structured feature stream provided explicit statistical, domain-specific, and protocol-based attributes. Their combination through the attention-based fusion strategy allowed the model to effectively consider both implicit lexical features and explicit URL features during classification.

The small difference between the CNN and the fusion model is due to the fact that the CNN stream already produced very high accuracy and F1-score. But the fusion model boosted recall from 0.9975 to 1.0000 and completely removed false negatives in the test set. This is important in cybersecurity as failing to detect malicious URLs exposes the network to phishing, malware, credential theft, and malicious redirection attacks.

The Random Forest model had the highest standalone ROC-AUC in the two baseline branches, indicating the engineered structured features contained rich discriminative information. But its confusion matrix contained 25 false negatives and 16 false positives, suggesting that the structured features were not as effective as the lexical-structured representation. The fusion model has captured the lexical feature sensitivity of the CNN stream and the explicit feature reasoning in the structured stream, leading to a better balance.

In conclusion, the results justify the primary hypothesis of this research: a binary classification of malicious URLs can benefit from a hybrid representation of URLs that couples the modelling of the character sequence of URLs with engineered structured features. The attention-guided fusion model was the best performing model with respect to accuracy, recall, F1-score, ROC-AUC, and false negatives among the implemented internal models. This shows that the hybrid fusion model is effective for binary malicious URL classification under the experimental setup.

V. REAL-TIME DEPLOYMENT AND PRACTICAL APPLICATION

The trained malicious URL detection framework was deployed through a Flask-based API to support real-time URL inference. The deployment layer connects the trained CNN, calibrated Random Forest, and fusion models with the preprocessing artifacts required for consistent prediction. Browser-oriented phishing protection systems and Flask-based cybersecurity dashboards reported in recent studies demonstrate the practical relevance of service-based and user-facing threat detection interfaces [16], [17], [24]. In the present implementation, the deployed API enables automated URL classification by reproducing the same lexical and structured preprocessing logic used during model development.

A. Flask-Based Deployment Framework

The deployment framework loads the saved fusion model, CNN model, calibrated Random Forest model, tokenizer, scaler and one-hot encoder when the API starts. The API also loads the custom tensor-processing

layers needed by the fusion architecture, such as expand-dimension, squeeze, and clip layers used in model design. Cross-origin resource sharing is also enabled to allow client interaction from a browser-based or web interface.

During inference, the API takes a URL as input and feeds it into the two streams. For the CNN stream, the protocol is stripped if present, the URL is downcased, character-tokenized and fixed-length-padded. For the structured-feature stream, the API extracts URL length, number of special characters, number of numeric characters, entropy of URL, number of sub-domains, phishing-indicator terms, protocol-related features and domain-type features. The domain-type feature is encoded with the saved one-hot encoder and the structured feature vector is scaled with the saved sparse-compatible scaler.

The transformed character sequence is fed to the CNN, and the scaled structured feature vector is fed to the calibrated Random Forest model and the structured stream of the fusion model. The fusion model takes in the padded URL sequence and structured feature vector to output the final maliciousness probability. The API outputs the submitted URL, final prediction label, fusion confidence, CNN contribution, Random Forest contribution and inference time in JSON.

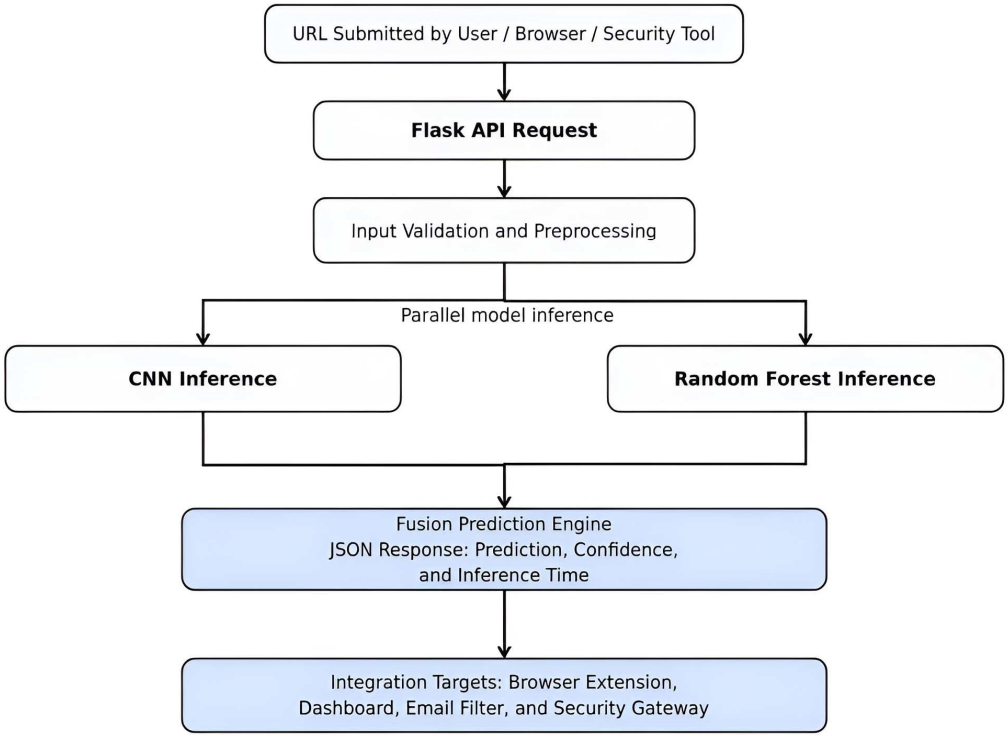

Fig. 5. Real-time deployment workflow of the malicious URL detection system. Source: Author's own work, adapted from [25]

B. Real-Time Prediction Flow

The deployed prediction workflow can be broken down into four steps. First, the client sends a URL to Flask API. Second, the API processes the request to generate the input model features: the character sequence and the scaled feature vector. Third, the CNN, calibrated Random Forest and fusion models produce their respective predictions. Fourth, the API returns a JSON response with the classification result and probabilities.

This approach enables model deployment using the same preprocessing as during training. The same tokenizer, one-hot encoding, scaling and models are applied at inference time, processing URLs in the same way they were processed during training. The Cross-Origin Resource Sharing (CORS) API also allows browser-based user interfaces, web dashboards, and service-oriented cybersecurity applications.

C. API Response Performance

The deployment evaluation reports three response-oriented measurements: average API response time, maximum API throughput, and model inference time. The average API response time was 5,423 ms, the maximum API throughput was 11.33 requests per second, and the model inference time was 157.96 ms. These values show that the trained framework was successfully packaged as a Flask-based inference service capable of processing URL classification requests through an API endpoint.

Table VI. API response performance metrics. Source: Author's own work, adapted from [25]

Parameter	Value
Average API Response Time	5,423 ms
Maximum API Throughput	11.33 requests/sec
Model Inference Time	157.96 ms

D. Practical Applicability

The deployment metrics demonstrate that the approach is not limited to evaluating the model offline, but can also be deployed as a service-based malicious URL detector. The API takes a URL, processes it with branch-specific transformations, makes CNN and Random Forest predictions, performs fusion, and returns the final prediction in a JSON format. This model enables its integration with browser add-ons, web front-end, and cybersecurity front-end.

The gap between the model inference time and the API's average response time corresponds to the operations undertaken by the service layer. This includes request parsing, input validation, URL canonicalization, character-based tokenization, structured feature extraction, one-hot encodings, sparse scaling, model inference, format prediction, and JSON response creation. The deployment metrics thus reflect the Flask inference pipeline implementation, rather than the standalone classifier.

The deployed model framework offers an operational real-time inference service for malicious URL classification. The service-oriented architecture allows external clients to request URL classification services without direct access to the training environment, which facilitates real cybersecurity applications where suspicious URLs need to be automatically classified before being permitted, blocked or referred to human analysts for further examination.

VI. CONCLUSION AND FUTURE WORK

The attention-based CNN and Random Forest (RF) hybrid fusion model in this work tackles binary malicious URL classification by jointly learning lexical patterns from the character level and structured URL features. The CNN stream captured discriminative lexical (text) patterns from normalized and padded URL sequences, whereas the structured feature stream captured explicit URL features, such as URL length, special-character count, numeric-character count, entropy, domain type, subdomain count, phishing-indicator terms, and protocol-safety behavior. The attention-based fusion stream combined these representations for final detection, thus allowing the model to exploit complementary lexical and structured cues in a single detection framework.

The model was tested on a balanced 60,000-URL dataset extracted from the publicly available Kaggle "Malicious And Benign URLs" dataset [18]. Using a stratified 80:20 hold-out split, the fusion model achieved 0.9990 accuracy, 0.9980 precision, 1.0000 recall, 0.9990 F1-score, and 0.9995 ROC-AUC. The fusion model achieved higher scores compared to the CNN and Random Forest models in the same evaluation setting. The confusion matrix results also indicated that the fusion model achieved zero false negatives on the held-out test set, with only a few false positives. This suggests the benefits of incorporating character-based lexical learning with structured URL features for identifying malicious URLs.

We further deployed the trained system via a Flask-based API for URL inference. The deployment pipeline loads the trained CNN, calibrated Random Forest and fusion models, as well as the tokenizer, scaler and one-hot encoder. The URLs are processed with the same dual view representation and the API responses are returned as JSON. The deployment performance metrics reported are an average API response time of 5,423 ms, maximum throughput of 11.33 requests per second and model inference time of 157.96 ms. This deployment demonstrates the trained system can be deployed as an inference service for cybersecurity screening of browser-connected or web-based systems.

Future research will enhance the model's robustness, generalizability, and deployment speed. Cross-validation on other malicious URL datasets will extend empirical evidence to other distributions of URLs and types of attacks. Adversarial URL transformations such as obfuscation, typo squatting, randomly generated subdomains and encoded characters will be used to test robustness against evasion. A more efficient preprocessing, feature extraction and API response mechanism will lower response time and enhance deployment in high-throughput scenarios. Explainability techniques like SHAP and LIME will also be incorporated to explain feature importance for model predictions and enhance transparency in cybersecurity workflows.

REFERENCES

[1] Q. E. ulHaq, M. H. Faheem, and I. Ahmad, "Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks," *Applied Sciences*, vol. 14, no. 22, p. 10086, Nov. 2024, doi:10.3390/app142210086.

[2] T. Tabassum, Md. M. Alam, Md. S. Ejaz, and M. K. Hasan, "A Review on Malicious URLs Detection Using Machine Learning Methods," *Journal of Engineering Research and Reports*, vol. 25, no. 12, pp. 76–88, Dec. 2023, doi:10.9734/jerr/2023/v25i121042.

[3] R. Alzubi, T. Bishtawi, and H. Kassem, "Improving Web Security through Machine Learning: A Feature-Based Methodology for Detecting Phishing URLs," *Engineering, Technology & Applied Science Research*, vol. 15, no. 5, pp. 26845–26851, Oct. 2025, doi:10.48084/etasr.12015.

[4] L. Jin, R. Huang, X. Zhang, and F. Wan, "A Malicious URL Detection Method Based on Bert-CNN," 2024. doi:10.3233/ATDE240115.

[5] L. Omar, A. Mezrioui, and A. Belmekki, "URL_trigger: Real time solution for Detection Malicious URL using Deep Learning," 2024, pp. 328–334. doi:10.2991/978-94-6463-360-3_33.

[6] R. Yang, K. Zheng, B. Wu, C. Wu, and X. Wang, "Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning," *Sensors*, vol. 21, no. 24, p. 8281, Dec. 2021, doi:10.3390/s21248281.

[7] N. Alsuqayh, A. Mirza, and A. Alhogail, "Exploring feature engineering and explainable AI for phishing website detection: a systematic literature review," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 15, no. 6, p. 5863, Dec. 2025, doi:10.11591/ijece.v15i6.pp5863-5878.

[8] P. Paithane, "A Systematic Review of Evaluating the Efficiency of Hybrid Machine Learning Techniques in Unmasking Phishing URLs," *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, vol. 08, pp. 1–5, May 2024, doi:10.55041/IJSREM29671.

[9] H. Gurung, R. Nepal, and S. Nepal, "Phishing URL Detection Using CNN-LSTM and Random Forest Classifier," May 2024. [Online]. Available: https://www.

[10] Q. Zhang, Y. Bu, B. Chen, S. Zhang, and X. Lu, "Research on phishing webpage detection technology based on CNN-BiLSTM algorithm," *J. Phys. Conf. Ser.*, vol. 1738, no. 1, p. 012131, Jan. 2021, doi:10.1088/1742-6596/1738/1/012131.

[11] B. C. Ujah-Ogbuagu, O. N. Akande, and E. Ogbuju, "A hybrid deep learning technique for spoofing website URL detection in real-time applications," *Journal of Electrical Systems and Information Technology*, vol. 11, no. 1, p. 7, Jan. 2024, doi:10.1186/s43067-023-00128-8.

[12] Md. I. Hossain, K. A. Al Arafat, B. Shepard, K. Craig, and I. Parvez, "A Graph-Attentive LSTM Model for Malicious URL Detection," Oct. 2025, [Online]. Available: http://arxiv.org/abs/2510.15971

[13] Y. Wang, W. Ma, H. Xu, Y. Liu, and P. Yin, "A Lightweight Multi-View Learning Approach for Phishing Attack Detection Using Transformer with Mixture of Experts," *Applied Sciences*, vol. 13, no. 13, p. 7429, Jun. 2023, doi:10.3390/app13137429.

[14] A. Jadhav and P. Chandre, "A Hybrid Heuristic-Machine Learning Framework for Phishing Detection Using Multi-Domain Feature Analysis," *Engineering, Technology & Applied Science Research*, vol. 15, no. 5, pp. 27219–27226, Oct. 2025, doi:10.48084/etasr.11548.

[15] K. M. Sujon, R. Hassan, M. E. Zainodin, M. A. Salamat, S. Kasim, and A. Alanda, "A Hybrid Approach for Malicious URL Detection Using Ensemble Models and Adaptive Synthetic Sampling," *JOIV : International Journal on Informatics Visualization*, vol. 9, no. 5, p. 2055, Sep. 2025, doi:10.62527/joiv.9.5.4627.

[16] Mr. Omkar Santosh Kale, Mrs. PratikshaShivnathDhumal, Mrs. Sakshi SachinThorat, Prof. K. S. Ambatkar, and Dr. A. A. Khatri, "SafeURL_AI_Extension: An Intelligent Browser Extension for Real-Time Detection of Malicious and Phishing Content," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 632–637, Nov. 2025, doi:10.48175/IJARSCT-29686.

[17] Prof. V. Jangade, A. Khanzode, P. Dhande, and A. Bangre, "ABHI SHIELD - Artificial Brain for Harm Identification," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 13, no. 11, pp. 303–308, Nov. 2025, doi:10.22214/ijraset.2025.74584.

[18] S. Kumar, "Malicious and Benign URLs," 2019, *Kaggle*. [Online]. Available: https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls

[19] P. Priya, N. Mohan Kumar, A. Professor, and M. Final Semester, "Malicious URL Detection using Machine Learning," 2024. [Online]. Available: www.jetir.org

[20] A. Jose, "Phishing URL Detection Using XGBoost," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 12, no. 5, pp. 1255–1260, May 2024, doi:10.22214/ijraset.2024.61807.

[21] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, "Intelligent phishing detection scheme using deep learning algorithms," *Journal of Enterprise Information Management*, vol. 36, no. 3, pp. 747–766, Apr. 2023, doi:10.1108/JEIM-01-2020-0036.

[22] S. T, S. M, H. K L, M. S V N, and M. K. BH, "Hybrid Machine Learning Approach for Real-Time Malicious URL Detection Using SOM-RMO and RBFN with Tabu Search," 2024. [Online]. Available: www.ijacsa.thesai.org

[23] G. Lingaiah and R. K. Pandey, "Design Of An Intelligent Deep Learning Framework For Phishing Url Detection In Cybersecurity Applications," *Int. J. Appl. Math. (Sofia).*, vol. 38, no. 5s, pp. 1033–1049, Oct. 2025, doi:10.12732/ijam.v38i5s.370.

[24] DanielRaj K, Dr. J. Japhynth, and Dr. T. Jasperline, "Intelligent Hybrid Learning Framework for Adaptive Real-Time Cyber Threat Detection and Response," *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 12, no. 5, pp. 254–269, Oct. 2025, doi:10.32628/IJSRSET251382.

[25] S. Sitaula, "Real-Time Malicious URL Detection Using a Hybrid Learning Approach," Purbanchal University School of Science and Technology (PUSAT), Biratnagar, Nepal, 2025.