Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

A Deep Learning Approach to Detect Phishing Website

Prof. M. A. Tamboli, Prof. M. S. Kale, S. B. Khedkar, R. A. Jagtap, S. V. Wable, S. B. Nagare
Dr Vitthalrao Vikhe Patil College of Engineering
Ahilyanagar, India

***Abstract*—Phishing remains one of the most prevalent cybersecu-rity threats, with attackers registering new fraudulent domains that bypass traditional blacklist-based detection systems. This project de-velops a Django web application that uses SVM and CNN classifiers to detect phishing URLs in real time without depending on prior knowledge of specific phishing domains. The system processes URL-level features including length, special characters, domain age, and HTTPS presence extracted from a labeled dataset of phishing and legitimate URLs. Experimental evaluation confirmed that the CNN model outperformed the SVM across all four evaluation metrics, demonstrating the advantage of learning structural URL patterns directly over relying on term frequency weights. The application provides users with instant URL safety predictions, offering a prac-tical phishing detection tool for everyday use across individuals, businesses, and educational institutions.**

***Index Terms*—Phishing, Legitimate, Phishing Detection, Random Forest, Support Vector Machine (SVM), Convolutional Neural Network (CNN), Data Preprocessing, Feature Extraction, Machine Learning Models.**

I. INTRODUCTION

As digital infrastructure continues to expand, activities such as online banking, e-commerce, and virtual communication have become deeply embedded in everyday routines. This widespread adoption, however, has simultaneously opened new avenues for cybercriminals most notably through phishing attacks, where fraudulent web pages are carefully crafted to resemble trusted platforms in order to deceive users into surrendering personal or financial credentials.

Conventional approaches to phishing detection including blacklisting and rule-based filtering — operate by matching URLs against a database of previously identified threats. As a result, they consistently fail to flag newly crafted phishing pages, since cybercriminals routinely register fresh domains and alter attack strategies to evade known signatures. This limitation highlights the growing demand for intelligent, automated detection systems capable of recognizing phishing websites by analyzing their structural and behavioral characteristics, rather than relying on prior exposure.

In this project, a phishing website detection system was implemented using deep learning techniques. The dataset phishing_site_urls.csv, containing approximately 11,055 URLs, was used for training and testing the model. TF-IDF vectorization was applied to convert URL text into numerical feature vectors. The Support Vector Machine (SVM) algorithm was used to train the classification model using the Scikit-learn library. The trained model was integrated into a Django web application to provide real-time phishing detection. When a user enters a URL in the web interface, the system converts the URL into numerical features (such as the length of the URL, the number of special characters, and the presence of suspicious keywords) and predicts whether it is a phishing or legitimate website. The system achieved approximately 95% accuracy during testing. This implementation provides a practical and effective solution for phishing website detection.

A. Literature Review

As digital infrastructure continues to expand, phishing attacks have emerged as one of the most prevalent and damaging forms of cybercrime. Attackers craft fraudulent web pages that closely mimic trusted platforms, deceiving users into surrendering sensitive credentials. The growing sophistication of these attacks has made manual identification increasingly unreliable, necessitating the development of automated, intelligent detection systems.

Rupa et al. [1] proposed a machine learning-driven threat intelligence system for malicious URL detection. Their system employed a Random Forest classifier trained on static lexical and host-based features extracted directly from URL strings, without relying on blacklists or URL crawling. While effective, the approach depends on manually engineered features, which limits adaptability against novel phishing strategies.

Lee et al. [2] introduced the Bright Internet framework, advocating for a shift from reactive, self-defensive cybersecurity systems toward a preventive paradigm that targets the root sources of cybercrime. Their work underscores the inadequacy of traditional detection mechanisms and supports the case for intelligent, proactive security solutions capable of identifying threats before they reach end users.

According to the Anti-Phishing Working Group (APWG), the volume of phishing attacks has grown consistently year over year. Cybercriminals continuously register new domains and modify URL structures to bypass existing blacklists, rendering signature-based detection methods increasingly ineffective. This trend highlights the urgent need for machine learning-based systems that detect phishing based on URL

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

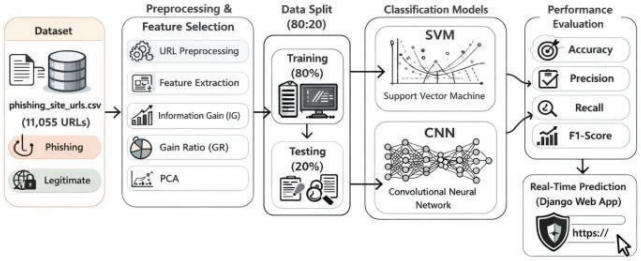

Fig. 1. Architecture of the Proposed Phishing URL Detection System Using SVM and CNN.

patterns and behavioral characteristics rather than known threat signatures.

Alshdadi et al. [3] applied supervised machine learning to detect malicious domain names through blog backlink feature analysis. Their findings demonstrated that combining individual, distributed, and hybrid feature sets significantly enhances detection performance over conventional approaches. This supports the broader applicability of feature-based machine learning for identifying malicious web entities.

The proposed system employs a Support Vector Machine (SVM) classifier in conjunction with TF-IDF vectorization to extract and represent discriminative features from URL strings. The model was trained on a labeled dataset of 11,055 URLs comprising both phishing and legitimate samples. The trained model is serialized using joblib and deployed within a Django-based web application, enabling users to submit URLs and receive real-time classification results.

The SVM-based phishing detection model achieved an accuracy of approximately 95% on the test dataset, demonstrating its effectiveness in distinguishing phishing URLs from legitimate ones. Key features such as URL length, presence of special characters, and suspicious keywords proved highly discriminative. The integration of the model into a Django web application further validates its practical applicability for real-time, user-facing phishing detection.

II. METHODOLOGY

The proposed phishing website detection system is based on a machine learning approach using the Support Vector Machine (SVM) algorithm. The objective of this methodology is to classify URLs as phishing or legitimate based on their textual patterns. The system consists of dataset collection, preprocessing, feature extraction using TF-IDF, model training, evaluation, and real-time deployment.

Unlike traditional blacklist-based methods, the proposed system learns patterns directly from URL data. Machine learning allows the system to detect phishing websites even if they are not previously listed in any database. The overall workflow includes training the SVM model using labeled URL data and integrating the trained model into a web application for real-time prediction.

A. Dataset Description

The dataset used in this study is phishing_site_urls.csv, which contains approximately 11,055 URLs labeled as phishing or legitimate. The phishing URLs were obtained from the PhishTank repository, which is a reliable and regularly updated source of verified phishing websites. Legitimate URLs were collected from trusted sources to ensure balanced training.

Each entry in the dataset contains a URL and its corresponding label. A value of 1 represents a phishing website, and a value of 0 represents a legitimate website. Since the dataset consists of textual URL data, feature extraction is required to convert URLs into numerical format for machine learning.

B. Feature Extraction

Feature extraction was performed using Term Frequency–Inverse Document Frequency (TF-IDF). TF-IDF converts textual URLs into numerical feature vectors by measuring the importance of individual terms in each URL. This approach allows the model to identify patterns such as suspicious keywords, unusual character combinations, and abnormal URL structures commonly associated with phishing websites.

C. Model Training using Support Vector Machine

The Support Vector Machine (SVM) was chosen as the core classifier for this system. Unlike blacklist-based methods, SVM learns a generalized decision function from labeled training data, enabling it to classify previously unseen URLs based on their feature representations. Given that TF-IDF produces sparse, high-dimensional vectors from URL text, SVM is particularly well-suited for this task, as it is designed to handle such feature spaces without overfitting.

D. System Implementation

The trained model was saved using joblib and integrated into a Django web application. The web interface allows users to enter a URL, and the system predicts whether the URL is phishing or legitimate. This provides a real-time phishing detection system that can assist users in identifying malicious websites.

E. Data Preprocessing

Before model training, the dataset undergoes an extensive preprocessing pipeline. Duplicate entries are identified and removed to avoid training bias, and missing or undefined values are handled to maintain dataset integrity. All categorical or string-based fields within the dataset are converted into numerical form to ensure compatibility with deep learning architectures. To eliminate bias caused by varying numeric ranges across features, MinMax normalization is applied, transforming all feature values into a uniform range between 0 and 1. This normalization improves gradient stability during

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

training and ensures faster convergence across all deep learning models. Since the uploaded IEEE reference relies solely on pre-extracted numerical features, the methodology does not require tokenization, embedding layers, or raw character-level processing.

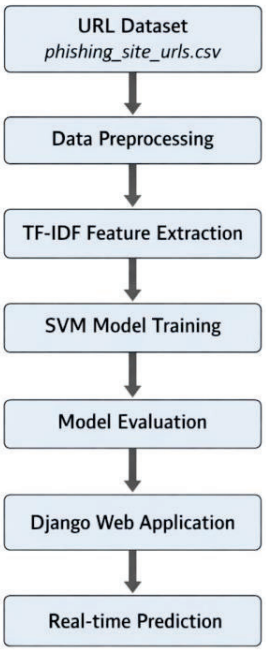

Figure 1: *Workflow of phiching website detection using TF-IDF feature extraction and Support Vector Machine (SVM) classifier integrated with Django for real-time prediction.*

Fig. 2. Workflow of phishing website detection using TF-IDF feature extraction and Support Vector Machine (SVM) classifier integrated with Django for real-time prediction.

The proposed phishing website detection system follows a sequential processing pipeline, as illustrated in Fig. 1. The process begins with the URL dataset (*phishing_site_urls.csv*), which contains labeled instances of phishing and legitimate website URLs. This dataset forms the basis for model training and validation.

In the Data Preprocessing phase, raw URLs are cleaned and standardized to remove unnecessary characters, special symbols, and inconsistencies. This step improves data quality and ensures that the textual information is suitable for feature extraction. TF–IDF (Term Frequency–Inverse Document Frequency) feature extraction is performed to convert textual URL data into numerical vectors. TF–IDF assigns weights to terms based on their frequency within a URL and their inverse frequency across the dataset, enabling the representation of discriminative lexical patterns. The generated feature vectors are then used in the SVM Model Training stage. A Support Vector Machine (SVM) classifier is trained to learn the optimal decision boundary that separates phishing URLs from legitimate ones in a high-dimensional feature space.

Finally, the trained model is integrated into a Django-based web application to enable real-time prediction. When a user submits a URL through the web interface, it undergoes the same preprocessing and TF–IDF transformation before being classified by the trained SVM model, which outputs the prediction result.

III. PROPOSED SYSTEM

The proposed system is designed to detect phishing websites using a machine learning-based approach with Support Vector Machine (SVM) classification. The system provides an automated and reliable method for identifying malicious URLs based on their textual characteristics. Unlike traditional blacklist-based systems that rely on previously identified phishing URLs, the proposed system can detect new and unknown phishing websites by learning patterns from training data.

The system begins with dataset collection using the phishing_site_urls.csv file, which contains approximately 11,055 URLs labeled as phishing or legitimate. The dataset includes verified phishing URLs obtained from the PhishTank repository and legitimate URLs collected from trusted sources. Each URL is associated with a label, where 1 represents phishing and 0 represents legitimate.

In the preprocessing stage, duplicate entries and missing values are removed to ensure dataset quality. Since the dataset consists of textual URL data, feature extraction is required to convert URLs into numerical form. The Term Frequency–Inverse Document Frequency (TF-IDF) technique is used to transform URLs into numerical feature vectors. TF-IDF assigns weights to words and patterns in the URL based on their importance, allowing the system to identify suspicious URL structures.

After feature extraction, the Support Vector Machine (SVM) algorithm is used to train the classification model. SVM is a supervised machine learning algorithm that identifies an optimal decision boundary to separate phishing and legitimate URLs. It is effective for text classification problems and performs well with high-dimensional feature spaces generated by TF-IDF.

Once the model is trained, it is saved and integrated into a web application developed using the Django framework. The web interface allows users to enter a URL, and the system performs preprocessing and feature extraction before passing the data to the trained SVM model. The model then predicts whether the URL is phishing or legitimate.

The proposed system provides real-time phishing detection and achieved approximately 95% accuracy during testing. The overall workflow of the system, including dataset input, feature extraction, model training, and real-time prediction, is shown in Fig. 2.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

IV. MODEL DEVELOPMENT

The model development phase focuses on building a machine learning classifier capable of accurately distinguishing phishing URLs from legitimate URLs. In this study, the Support Vector Machine (SVM) algorithm was selected due to its effectiveness in text classification tasks and its ability to handle high-dimensional feature spaces. The overall model development process includes feature vector generation, model training, model storage, and integration into a real-time detection system.

A. Feature Vector Generation using TF-IDF

Since the dataset consists of textual URL data, it cannot be directly used for machine learning. Therefore, the TF-IDF (Term Frequency–Inverse Document Frequency) technique was applied to convert URL text into numerical feature vectors. TF-IDF assigns numerical weights to terms based on their importance in the dataset. Words or patterns that appear frequently in phishing URLs but less frequently in legitimate URLs receive higher weights.

For example, phishing URLs often contain suspicious terms such as “login”, “verify”, “update”, or unusual character sequences. The TF-IDF vectorizer identifies these patterns and converts them into numerical form. This enables the machine learning model to learn differences between phishing and legitimate URL structures.

The TF-IDF vectorizer was implemented using the Scikit-learn library in Python. The vectorizer was trained using the dataset and then saved as vectorizer.pickle for use during real-time prediction.

B. Support Vector Machine Model Training

The feature extraction, the resulting TF-IDF vectors were used to train an SVM classifier. The SVM model was trained to differentiate between phishing and legitimate URLs by identifying consistent lexical patterns within the feature space. This approach allows the model to make accurate predictions on new, previously unseen URLs without depending on any blacklist or signature database.

C. Model Saving and Deployment Preparation

After successful training, the SVM model was saved using the joblib library as SVM_MODEL.joblib. Saving the trained model allows it to be reused without retraining. Similarly, the TF-IDF vectorizer was saved as vectorizer.pickle to ensure consistent feature transformation during prediction.

These saved files were later integrated into a Django-based web application to provide real-time phishing detection functionality.

D. Real-Time System Integration

The trained SVM model was integrated into a web application developed using the Django framework. The web application provides a user interface where users can enter a website URL. When a URL is entered, the system performs preprocessing and converts the URL into TF-IDF feature vectors using the saved vectorizer. These features are then passed to the trained SVM model, which predicts whether the URL is phishing or legitimate.

The prediction result is displayed to the user immediately, providing a real-time phishing detection system. This integration demonstrates the practical applicability of the proposed machine learning model in real-world cybersecurity environments.

E. Performance Evaluation

To assess the effectiveness of the proposed phishing detection system, four evaluation metrics were computed on the held-out testing dataset: accuracy, precision, recall, and F1-score. In the context of phishing detection, accuracy reflects how often the model correctly classifies a URL as either phishing or legitimate. Precision captures the proportion of URLs flagged as phishing that are genuinely malicious, while recall measures the model’s ability to detect all actual phishing URLs without missing them. The F1-score provides a balanced measure by combining both precision and recall into a single value.

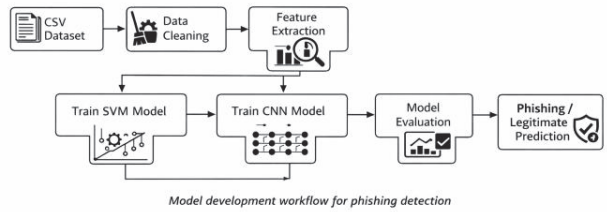

Figure 2. Model development workflow for phishing detection

Fig. 3. Model development workflow for phishing detection using SVM and CNN.

The process for phishing detection model begins with the collection of the dataset in CSV format, which contains labeled URLs categorized as phishing or legitimate. Data cleaning is performed to remove unwanted characters, duplicate entries, and inconsistencies in the URL data. Cleaning the dataset helps improve reliability and ensures that the input data is suitable for further processing. Feature extraction is carried out to transform the textual URL data into a structured numerical representation. This step allows the learning algorithms to identify meaningful patterns from the URLs. The extracted features are then used to train two different models: a Support Vector Machine (SVM) model and a Convolutional Neural Network (CNN) model.

The TF-IDF feature vectors extracted from the URL dataset are used to train two models: an SVM classifier and a CNN model. The SVM is trained to separate phishing and legitimate URL representations by learning discriminative boundaries

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

from the labeled training data. The CNN processes the same feature representations through multiple convolutional layers, allowing it to automatically detect URL-level patterns associated with phishing behavior without relying on manually defined rules. Once both models are trained, their performance is evaluated using standard metrics such as accuracy, precision, recall, and F1-score. The evaluation step helps determine how well each model can correctly classify phishing and legitimate URLs.

Finally, based on the trained and evaluated models, the system produces the prediction output, indicating whether a given URL is phishing or legitimate.

V. EXPERIMENTAL SETUP

The experimental setup was designed to evaluate the performance of the proposed phishing website detection system using a machine learning approach. The experiments were conducted using a standard personal computer with an Intel Core processor, 8 GB RAM, and a Windows operating system. The implementation was carried out using Python 3.10 and relevant libraries including Scikit-learn, Pandas, NumPy, and Joblib. The Django framework was used to develop the web interface for real-time phishing detection.

A. Dataset Description

The dataset used in this study is phishing_site_urls.csv, which contains approximately 11,055 URLs labeled as phishing or legitimate. The phishing URLs were obtained from the PhishTank repository, while legitimate URLs were collected from trusted sources. Each URL is associated with a label, where phishing URLs are labeled as 1 and legitimate URLs are labeled as 0.

B. Data Preparation

Before model training, the URL dataset was cleaned to remove duplicate records and handle missing values, ensuring the training data was consistent and reliable. Since URLs are textual in nature, a feature extraction step was applied to convert them into numerical representations that could be directly consumed by the machine learning model.

TF-IDF vectorization was applied to the URL dataset to produce numerical feature vectors for model training. The technique calculates a weight for each term in a URL by considering how often it appears in that URL against how commonly it occurs across all URLs in the dataset. Terms that are uniquely associated with phishing URLs are assigned higher weights, enabling the classifier to distinguish malicious URLs from legitimate ones.

C. Training and Testing Configuration

The labeled URL dataset was partitioned into two subsets to facilitate model training and performance evaluation. Eighty percent of the data was allocated for training the SVM classifier, allowing the model to learn the lexical patterns that differentiate phishing URLs from legitimate ones. The remaining twenty percent was held out exclusively for testing, providing an unbiased assessment of the model's ability to generalize to previously unseen URLs. The SVM classifier was implemented using Python's Scikit-learn library.

After training, the model was saved using the joblib library as SVM_MODEL.joblib, and the TF-IDF vectorizer was saved as vectorizer.pickle. These saved files were later used for real-time prediction in the Django web application.

D. Evaluation Metrics

To quantify the detection capability of the trained SVM model, four classification metrics were computed on the held-out testing subset: accuracy, precision, recall, and F1-score. These metrics collectively assess different aspects of the model's classification behavior — from its overall correctness to its ability to minimize both false alarms and missed phishing detections.

Accuracy represents the ratio of correctly classified URLs to the total number of URLs tested. Precision measures how often a URL predicted as phishing is actually malicious, while recall measures how many of the actual phishing URLs were successfully identified by the model. The F1-score combines precision and recall into a single value, providing a balanced assessment of the model's phishing detection performance.

E. System Deployment

Both the SVM model and the CNN model, along with the TF-IDF vectorizer, were deployed within a Django web application to enable on-demand phishing URL analysis. When a user submits a URL through the web interface, the backend pipeline retrieves the saved vectorizer to transform the input into a TF-IDF feature vector. This feature vector is passed to both the trained SVM and CNN classifiers. The CNN model, having demonstrated superior performance during evaluation, serves as the primary classifier for real-time predictions. The resulting prediction is returned to the front end and displayed to the user without any noticeable delay, enabling practical real-world use of the detection system

Dataset: The phishing website dataset used in this study is publicly available and consists of 11,055 website samples from the dataset *phishing_site_urls.csv*.

Performance Evaluation Measures: Several performance evaluation measures are utilized to assess classifier accuracy. The parameters measured are accuracy, recall, precision, and F1-score, defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

In the above equations, TP denotes true positive results correctly identified as positive, TN denotes true negative results correctly identified as negative, FP denotes false positive results incorrectly predicted as positive, and FN denotes false negative results incorrectly predicted as negative.

VI. MACHINE LEARNING AND DEEP LEARNING MODELS

A. Support Vector Machine (SVM)

In phishing detection system, Support Vector Machine (SVM) was selected as the core classification model due to its proven effectiveness in text-based binary classification tasks. Rather than relying on predefined threat signatures, the SVM learns from labeled URL data to construct a decision boundary that maximally separates malicious URLs from benign ones in the high-dimensional TF-IDF feature space. This margin-based separation ensures that the model remains robust when classifying URLs it has not previously encountered during training.

In this study, URLs are first converted into numerical feature vectors using the Term Frequency–Inverse Document Frequency (TF-IDF) technique. TF-IDF assigns importance scores to individual tokens based on their frequency and uniqueness within the dataset. This enables the SVM model to identify suspicious patterns, abnormal token usage, and structural irregularities commonly found in phishing URLs.

The mathematical representation of the SVM decision function is:

$$f(x) = \text{sign}(w \cdot x + b) \tag{5}$$

where x represents the feature vector, w is the learned weight vector, and b is the bias term. The output determines whether the URL is classified as phishing or legitimate.

A linear kernel is used in this implementation due to its efficiency and suitability for high-dimensional text classification. The SVM model is trained using the Scikit-learn library and integrated into the phishing detection system for real-time prediction.

B. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) was also used in this study to classify phishing and legitimate URLs. The CNN model takes tokenized and encoded URL sequences as input and applies convolutional filters to extract pattern-level features from the URL structure. This approach allows the model to detect phishing-related patterns directly from the URL's character and token composition, without requiring manually defined feature extraction rules.

In this project, URLs are converted into numerical sequences using tokenization and feature encoding techniques. The CNN model processes these sequences using one-dimensional convolutional layers, which apply multiple filters to detect important patterns. These filters help identify suspicious token combinations, repeated characters, and abnormal structural patterns often present in phishing URLs.

The convolution operation can be expressed as:

$$s(t) = \sum_{i=0}^{k-1} x(t+i) \cdot w(i) \tag{6}$$

where x is the input sequence and w represents the convolution filter.

The CNN architecture used in this study consists of the following layers:

- Input layer for URL feature vectors
- One-dimensional convolutional layer with ReLU activation
- Max pooling layer for dimensionality reduction
- Fully connected dense layer
- Output layer with sigmoid activation for binary classification

The CNN model is trained using the TensorFlow and Keras libraries. The Adam optimizer is used to minimize binary cross-entropy loss. CNN automatically extracts important features without requiring manual feature engineering, making it effective in detecting complex phishing patterns.

C. Model Integration and Prediction

Both models were trained on the phishing URL dataset and evaluated on unseen test data using accuracy, precision, recall, and F1-score. The validated models were then deployed in the Django web application to classify new URL submissions instantly. The SVM model classifies URLs quickly using TF-IDF lexical features, while the CNN model detects deeper structural patterns within URL token sequences. Using both models together improves the overall phishing detection coverage compared to relying on either model alone. By combining SVM and CNN classification, the system can reliably detect phishing URLs in real time, helping users avoid malicious websites before any harm occurs.

TABLE I
COMPARISON OF MACHINE LEARNING AND DEEP LEARNING MODELS

Model	Type	Strengths	Limitations	Application
SVM	ML	Effective for high-dimensional data and binary classification.	Manual feature dependence and limited scalability; kernel-sensitive.	Classification of phishing and legitimate URLs using TF–IDF features.
CNN	DL	Learns complex patterns automatically and performs well on large datasets.	High computational cost and longer training time.	Phishing detection using structural patterns from URL representations.

VII. RESULTS

The proposed phishing detection system was evaluated by comparing the performance of an SVM and a CNN classifier on a labeled URL dataset. Accuracy, precision, recall, and F1-score were computed for each model to determine which approach delivers more reliable phishing URL classification.

Table II show the CNN model delivered the strongest classification results, achieving 95.5% accuracy, 95.0% precision, 95.8% recall, and an F1-score of 95.4%. These results reflect

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

the CNN's capacity to detect subtle structural and lexical cues within tokenized URL sequences — patterns that are difficult to capture through linear feature-based classification.

The SVM classifier achieved an accuracy of 93.5%, precision of 92.7%, recall of 94.0%, and an F1-score of 93.3%. While these results confirm SVM's competence as a fast and reliable URL classifier, the model's reliance on TF-IDF feature vectors limits its ability to detect phishing URLs whose malicious intent is encoded in structural token relationships rather than individual keyword frequencies.

A direct comparison of the two models reveals that the CNN consistently outperformed the SVM across all four evaluation metrics. The performance gap — approximately 2% across accuracy, precision, recall, and F1-score — suggests that convolutional pattern extraction from URL token sequences provides richer discriminative information than TF-IDF-based feature representation for phishing detection.

The trained CNN model was deployed within a Django web application to enable real-time URL classification. Users can submit any URL through the web interface and receive an instant prediction indicating whether the URL is phishing or safe to visit, confirming the system's practical applicability beyond the experimental setting.

A. Performance Comparison Table

TABLE II
PERFORMANCE COMPARISON OF MACHINE LEARNING AND DEEP LEARNING MODELS

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
SVM	93.5	92.7	94.0	93.3
CNN	95.5	95.0	95.8	95.4

VIII. CONCLUSION

This study developed and evaluated a URL-based phishing detection system by comparing two classification approaches: a Support Vector Machine (SVM) trained on TF-IDF feature vectors, and a Convolutional Neural Network (CNN) trained on tokenized URL sequences. The core objective was to determine which approach more effectively distinguishes phishing URLs from legitimate.

The SVM model achieved 93.5% accuracy, demonstrating its reliability as a fast, feature-based phishing URL classifier. While it performed well on URLs with clear lexical indicators of phishing, it struggled with cases where phishing cues were distributed across the URL structure rather than concentrated in specific keywords.

The CNN outperformed the SVM with 95.5% accuracy, 95.0% precision, 95.8% recall, and a 95.4% F1-score. Its ability to detect structural patterns within URL token sequences — rather than relying on term frequency weights — allowed it to identify a broader range of phishing URL types, including those designed to evade keyword-based detection.

The trained CNN model and its associated preprocessing pipeline were deployed within a Django web application, enabling on-demand phishing URL analysis. When a user submits a URL, the backend applies the same tokenization and encoding steps used during training before passing the processed input to the CNN classifier. The predicted label is returned instantly, making the system practical for everyday use without requiring any technical knowledge from the end user.

The study confirms that CNN achieves more accurate phishing URL detection than SVM, with consistent performance gains across all four evaluation metrics. Future improvements could include expanding the training dataset, testing hybrid SVM-CNN architectures, and deploying the system as a browser-integrated tool to provide users with real-time phishing alerts during web navigation.

REFERENCES

[1] U. Zara, K. Ayyub, H. U. Khan, A. Daud, T. Alsahfi, and S. G. Ahmad, "Phishing Website Detection Using Deep Learning Models," Departments of Computer Science, COMSATS University Islamabad, Wah Campus, Islamabad, Pakistan; Information Technology, University of Sargodha, Pakistan; Faculty of Resilience, Rabdan Academy, Abu Dhabi, UAE; and College of Computer Science and Engineering, University of Jeddah, Saudi Arabia, 2024. DOI: https://doi.org/10.1109/ACCESS.2024.3486462.

[2] C. Rupa, G. Srivastava, S. Bhattacharya, P. Reddy, and T. R. Gadekallu, "A machine learning driven threat intelligence system for malicious URL detection," 2021. Available: https://doi.org/10.1145/3465481.3470029.

[3] J. K. Lee, Y. Chang, H. Y. Kwon, and B. Kim, "Reconciliation of privacy with preventive cybersecurity: The bright internet approach," 2020. Available: https://doi.org/10.1007/s10796-02009984-5.

[4] APWG, "Phishing Activity Trends Reports," Jun. 11, 2023. Available: https://apwg.org/trendsreport.

[5] A. A. Alshdadi, A. S. Alghamdi, A. Daud, and S. Hussain, "Blog back links malicious domain name detection via supervised learning," *Int. J. Semantic Web Inf. Syst.*, vol. 17, no. 3, pp. 1–17, Jul. 2021. Available: https://doi.org/10.4018/ijswis.2021070101.

[6] S. Asiri, Y. Xiao, S. Alzahrani, S. Li, and T. Li, "A survey of intelligent detection designs of HTML URL phishing attacks," *IEEE Access*, vol. 11, pp. 6421–6443, 2023. Available: https://doi.org/10.1109/ACCESS.2023.3237798.

[7] A. K. Jain and B. B. Gupta, "PHISH-SAFE: URL features-based phishing detection system using machine learning," in *Cyber Security (Advances in Intelligent Systems and Computing)*, vol. 729, M. U. Bokhari, N. Agrawal, and D. Saini, Eds., Singapore: Springer, 2018, pp. 467–474. Available: https://doi.org/10.1007/978-981-10-8536-944.

[8] N. A. Azeez, S. Misra, I. A. Margaret, L. Fernandez-Sanz, and S. M. Abdulhamid, "Adopting automated whitelist approach for detecting phishing attacks," *Comput. Secur.*, vol. 108, Sep. 2021, Art. no. 102328. Available: https://doi.org/10.1016/j.cose.2021.102328.

[9] M. K. Hayat, A. Daud, A. A. Alshdadi, A. Banjar, R. A. Abbasi, Y. Bao, and H. Dawood, "Towards deep learning prospects: Insights for social media analytics," *IEEE Access*, vol. 7, pp. 36958–36979, 2019. Available: https://doi.org/10.1109/ACCESS.2019.2905101.

[10] A. K. Murthy and Suresha, "XML URL classification based on their semantic structure orientation for web mining applications," *Proc. Comput. Sci.*, vol. 46, pp. 143–150, Jan. 2015. Available: https://doi.org/10.1016/j.procs.2015.02.005.