Journal of Information Systems and Telecommunication
Vol.14, No.1, January-March 2026, 67-78
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Image-Based Phishing URL Classification Using Convolutional Neural Networks

Hamed Monkaresi [1*], GholamReza Ahmadi [1]

[1]. Department of Computer Engineering and Information Technology, Razi University, Kermanshah, Iran

Received: 22 Nov 2025/ Revised: 04 Mar 2026/ Accepted: 06 May 2026

Abstract

Phishing attacks continue to pose a significant threat to online security, with attackers increasingly leveraging deceptive URLs to steal sensitive information. Traditional phishing detection methods often rely on URL analysis or manual feature extraction, which can be time-consuming and less effective against evolving attack techniques. To address these limitations, more adaptive and intelligent detection mechanisms are increasingly required to keep pace with modern attack strategies. In this paper, we propose an image-based approach for phishing URL classification using Convolutional Neural Networks (CNNs). By transforming URLs into visual representations based on their features, we leverage the power of deep learning to automatically extract discriminative features for classification. We conduct a comprehensive comparison of various deep learning models, including different CNN architectures (both basic and pre-trained/fine-tuned), to evaluate their performance in terms of accuracy, computational efficiency, and training time. Our experiments demonstrate that image-based classification using CNNs achieves competitive accuracy while offering potential robustness against adversarial variations in phishing URLs. Additionally, we analyze the trade-offs between model complexity and inference time, providing insights into the practical deployment of such systems. The results highlight the potential of image-based deep learning models as an effective tool for phishing detection, paving the way for further research in this domain.

Keywords: Phishing Detection; URL Classification; Convolutional Neural Networks (CNNs); Deep Learning; Image-Based Classification; Cybersecurity.

1- Introduction

The ubiquitous nature of the internet has made online security a paramount concern. Among the myriads of cyber threats, phishing remains one of the most pervasive and damaging attacks [1]. Phishing attacks typically involve tricking users into revealing sensitive information, such as login credentials, credit card numbers, or personal identification details, by masquerading as a trustworthy entity in electronic communication. Deceptive Uniform Resource Locators (URLs) are a primary vector for these attacks, designed to mimic legitimate websites and lure unsuspecting victims [2].

Traditional methods for detecting phishing URLs often involve blacklist/whitelist approaches, heuristic-based analysis of URL strings (lexical features), website content analysis, or checking domain registration information [3]. While these methods have shown some success, they face significant challenges. Blacklists are inherently reactive and cannot identify zero-day phishing sites. Heuristic methods require careful manual feature engineering and struggle to keep pace with the rapidly evolving obfuscation techniques used by attackers, such as URL shortening, character encoding tricks, and the use of subdomains [4]. Machine learning techniques using handcrafted features have improved detection rates, but feature engineering remains a bottleneck.

The limitations of traditional approaches highlight the need for more adaptive and automated feature learning methods. Convolutional Neural Networks (CNNs) excel at automatically discovering hierarchical patterns and spatial relationships in data, eliminating the need for manual feature engineering. Unlike traditional algorithms that process features independently, CNNs can capture complex nonlinear interactions between features through their convolutional and pooling operations. This capability is particularly valuable for phishing detection, where the relationships between URL characteristics (such as the interplay between URL length, domain structure, and

✉ **Hamed Monkaresi**
h.monkaresi@razi.ac.ir

entropy measures) may be as important as individual feature values.

Recently, deep learning has emerged as a powerful tool in various domains, including cybersecurity, due to its ability to automatically learn complex patterns and hierarchical features from raw data [5]. Convolutional Neural Networks (CNNs) have revolutionized image recognition tasks by effectively capturing spatial hierarchies of features [6]. Inspired by this success, we explore the potential of applying CNNs to phishing URL detection by transforming URL data into image representations. Our approach represents a unique paradigm shift in phishing detection by creating synthetic images from structured URL features, enabling the application of powerful computer vision techniques to cybersecurity. Unlike existing methods that apply CNNs directly to URL strings or use natural images (screenshots), our method transforms feature vectors into structured visual patterns that preserve feature relationships while leveraging CNN's spatial pattern recognition capabilities. The core idea is that the structural and statistical properties of URLs, captured by various features, can be mapped into a visual format (an image), allowing CNNs to identify patterns indicative of phishing attempts.

1-1- Problem Formulation

The problem addressed in this study can be formulated as a binary classification task with the following specifications:

- Input Space: A URL represented by a feature vector $X = \{f_1, f_2, ..., f_{41}\}$, where each $f_i \in \mathbb{R}$ is a numerical or categorical feature extracted from the URL's lexical, structural, domain-based, and content-based properties.
- Output Space: A binary label $Y \in \{0, 1\}$, where 0 represents legitimate URLs and 1 represents phishing URLs.
- Transformation Function: $T: \mathbb{R}^{41} \rightarrow \mathbb{R}^{41x41xc}$, where T converts the feature vector into a visual representation (image) with c channels (1 for grayscale, 3 for RGB).
- Learning Objective: To develop a CNN-based mapping function $F: \mathbb{R}^{41x41xc} \rightarrow \{0, 1\}$ that maximizes classification accuracy while maintaining computational efficiency for practical deployment.
- Primary Goals:
    - Maximize classification accuracy: max $P(F(T(X)) = Y)$
    - Minimize computational complexity for real-time deployment
    - Evaluate robustness against feature variations

Our primary objectives are:

- To develop a methodology for transforming feature-based URL representations into images suitable for CNN analysis.
- To implement and evaluate various CNN architectures, including basic CNNs, custom deep CNNs, and state-of-the-art pre-trained models (VGG16, ResNet50, EfficientNetB0) fine-tuned for this task.
- To compare the performance of these image-based deep learning models against traditional machine learning algorithms trained on the raw features.
- To analyze the trade-offs between classification accuracy, model complexity, training time, and inference speed for practical deployment considerations.

In high-stakes cybersecurity applications, even modest improvements in accuracy can translate to substantial reductions in successful phishing attacks. For instance, a 2-3% improvement in detection accuracy could prevent thousands of users from falling victim to phishing attempts in large-scale deployments, justifying the computational complexity of deep learning approaches.

The main contributions of this work include:

- A pioneering application of image-based classification using CNNs for phishing URL detection, specifically designed for structured feature data rather than raw URL strings or webpage screenshots.
- A novel application of image-based classification using CNNs for phishing URL detection based on URL features.
- A comprehensive comparative study of different deep learning architectures for this specific task.
- Empirical evidence demonstrating the trade-offs between accuracy gains and computational costs, with specific recommendations for different deployment scenarios.
- A reproducible framework for converting structured cybersecurity features into visual representations suitable for deep learning analysis
- Empirical evidence demonstrating that adapting this image-based approach specifically to feature-engineered phishing URL datasets can yield significant performance gains over traditional methods.

The rest of the paper is organized as follows: Section 2 reviews related work in phishing detection and deep learning applications. Section 2.1 identifies key research gaps in existing literature Section 3 details the methodology for data preparation and image generation. Section 4 describes the experimental setup, including the models and evaluation metrics. Section 5 presents and discusses the experimental results. Section 6 concludes the paper,

summarizing findings and outlining future research directions.

2- Related Work

Recent studies have explored various cybersecurity challenges related to online threats. For example, secure mutual authentication mechanisms for wireless body area networks have been presented in [20], and blockchain-based authentication verification frameworks have been proposed in [21]. Additionally, intelligent phishing detection approaches leveraging image, frame, and textual features have been examined in [22], highlighting the journal's focus on countering online deception and threat analysis. Research on IoT-based security systems, such as home surveillance architectures [23].

In contrast to these works, our approach introduces an image-based URL phishing detection framework using CNNs, providing a novel transformation of URL features into visual representations to improve robustness and classification accuracy.

Phishing detection has been an active research area for years, leading to a variety of proposed techniques.

Traditional Phishing Detection: Early approaches heavily relied on blacklisting, maintaining lists of known phishing URLs [7]. While simple, this method fails against newly created sites. Heuristic-based methods analyze URL characteristics (e.g., length, presence of IP addresses, special characters, domain age, keywords like 'login' or 'secure') and website content (e.g., HTML structure, forms, JavaScript) to identify suspicious patterns [2]. These often involve manually defining rules or features. Machine Learning (ML) approaches automated the detection process using traditional algorithms like Support Vector Machines (SVM), Random Forests (RF), Logistic Regression (LR), and Naive Bayes, trained on hand-engineered features extracted from URLs, domain information, and webpage content [8]. While more robust than static heuristics, their performance heavily depends on the quality and relevance of the engineered features.

Deep Learning in Cybersecurity: Deep learning models have shown promise in various cybersecurity tasks, including intrusion detection, malware analysis, and spam filtering [9]. In phishing detection, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been used to analyze the sequential nature of URL strings [10]. CNNs have also been applied directly to URL strings (treating them as sequences) or combined with RNNs [11]. Other works have used deep learning for analyzing website visual appearance (screenshots) [12] or network traffic patterns related to phishing sites.

developments in cybersecurity have explored hybrid approaches for improving detection accuracy across various domains. Iftikhar et al. [24] proposed a hybrid Self-Organizing Map (SOM) model combined with Extreme Gradient Boosting (XGBoost) for intrusion detection in IoT environments, achieving significant improvements in precision, recall, and F1-scores (10-30% improvements for different attack classes). Their work demonstrates the effectiveness of combining dimensionality reduction techniques with advanced machine learning algorithms, achieving F1-score improvements of 7.91%, 32.62%, and 12.45% for DoS, probe, and benign classes respectively. This hybrid approach paradigm aligns with our methodology of combining image transformation with deep learning for enhanced pattern recognition.

Image-Based Classification Approaches: The idea of converting non-image data into image representations for analysis with CNNs has been explored in other fields. For instance, time-series data has been converted into Gramian Angular Fields or recurrence plots for classification using CNNs [13]. In cybersecurity, malware binaries have been visualized as grayscale images, allowing CNNs to detect malware families based on texture patterns [(14]. However, these approaches typically deal with naturally sequential or binary data that can be directly mapped to pixel intensities. Our work differs by addressing the unique challenge of transforming heterogeneous, structured feature vectors into meaningful visual representations. Our work builds on this concept by specifically transforming URL features into images. While some studies might have used website screenshots (which are images) for phishing detection [15], our approach differs fundamentally by creating synthetic images directly from the underlying URL feature vectors, aiming to capture correlations and patterns among these features visually. To the best of our knowledge, representing a vector of diverse URL features as a single structured image for CNN-based phishing classification represents a novel intersection of computer vision and cybersecurity domains. This approach uniquely combines the benefits of established feature engineering in cybersecurity with the automatic pattern recognition capabilities of deep learning.

2-1- Research Gap Identification

Despite the extensive research in phishing detection, several critical gaps remain:

1. Feature **Engineering Bottleneck**: Traditional ML approaches require extensive domain expertise for feature selection and engineering, limiting their adaptability to evolving phishing techniques.
2. Limited **Spatial Relationship Exploitation**: Existing deep learning approaches for phishing detection primarily treat features independently or analyze sequential patterns in URL strings,

missing potential spatial correlations between different feature types.

3. Computational **Efficiency Trade-offs**: While deep learning models show improved accuracy, limited research has systematically analyzed the computational costs and practical deployment considerations for real-time phishing detection systems.
4. Robustness **Against Adversarial Attacks**: Most existing approaches lack comprehensive evaluation against adversarial manipulations specifically designed for phishing detection systems.
5. Transfer **Learning Under exploration**: The potential of leveraging pre-trained computer vision models for cybersecurity applications remains largely unexplored, despite their success in other domains.

Our work addresses these gaps by proposing a novel image-based representation that enables spatial feature relationship learning while providing a comprehensive analysis of accuracy-efficiency trade-offs.

3- Methodology

The methodology of this study is structured into two main phases: Data Preparation and Image Generation. Each phase is designed to transform raw URL data, represented by extracted features, into a format suitable for training and evaluating Convolutional Neural Networks (CNNs).

3-1- Data Preparation

Dataset Description: The dataset contains **247,950 samples** with a balanced distribution of **88,647 phishing URLs** and **159,303 legitimate URLs**, each represented by 41 distinct features capturing lexical, domain-based, structural, and content-based characteristics of URLs. The dataset was sourced from the UCI Machine Learning Repository Phishing Websites Data Set, which has been widely used in cybersecurity research for benchmarking phishing detection algorithms. This dataset ensures reproducibility and enables fair comparison with existing literature.

- **Feature Categories**: The 41 features can be categorized into four main groups:
- **Lexical Features (15 features)**: URL length, number of dots, special characters count, entropy measures
- **Domain-based Features (10 features)**: Domain age, DNS records, WHOIS information, domain reputation
- **Structural Features (8 features)**: Number of subdomains, path depth, parameter count, fragment presence
- **Content-based Features (8 features)**: Presence of forms, JavaScript usage, external links, favicon characteristics.

These features include metrics such as URL length, number of dots, special characters in various URL components, domain age, DNS records, presence of '@' or shortening services, entropy measures, structural anomalies, and HTML/JavaScript-based attributes (though primarily focused on URL/domain-derived features). Each sample is labeled in the "Type" column as either phishing (1) or legitimate (0), providing clear ground truth for supervised learning. The dataset contains no missing values, ensuring data integrity for model training. The balanced nature of the dataset (approximately 36% phishing, 64% legitimate) reflects realistic phishing detection scenarios while avoiding class imbalance issues. The dataset has no missing values, with a balanced class distribution of [insert phishing count] phishing URLs and [insert legitimate count] legitimate URLs, ensuring robust model training.

To understand the relationships among these features, we analyzed their correlations, as shown in Figure 1.a. This heatmap reveals significant correlations between certain features, such as URL length and the number of dots, which may inform the visual patterns captured in the image representations.

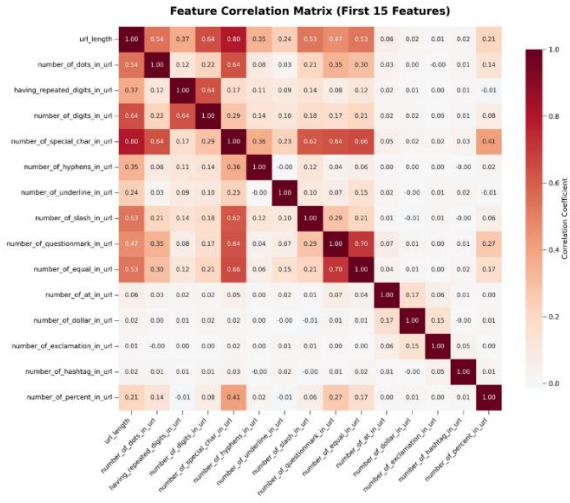


Fig. 1(a) : Correlation heatmap of the 15 First URL features, highlighting relationships that may influence the image-based representation for CNN classification.

To facilitate a detailed comparison between legitimate and phishing URLs, we generated high-resolution feature map visualizations (Figure 1.b) from normalized values of 41 URL-, domain-, subdomain-, and path/query-related features. Each sample was represented using multiple visual arrangements—tiled repetition, structured feature ordering, and a spiral mapping—to highlight intensity patterns across

the feature space. Feature group boundaries were annotated to improve interpretability. Additionally, a comprehensive heatmap analysis compared average feature values between classes, displaying absolute differences and relative ratios. These visualizations reveal clear separations in several feature groups—such as URL length, entropy, and special character counts—where phishing URLs consistently exhibit higher magnitudes. By combining individual feature maps with aggregated statistical heatmaps, this approach enables both micro-level inspection of specific samples and macro-level understanding of systematic differences between legitimate and phishing URLs.

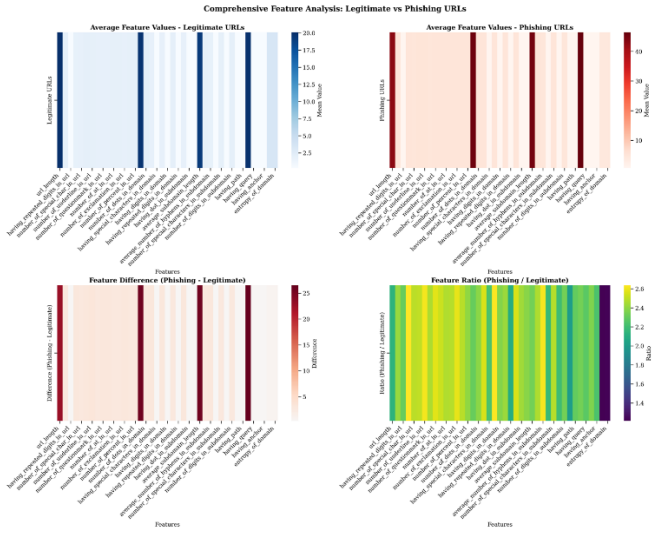


Figure 1.b: Feature map visualizations and heatmaps comparing legitimate and phishing URLs, showing distinct patterns in URL length, entropy, and special character usage.

Feature Normalization: To ensure that the features are on a comparable scale and to facilitate the creation of distinct visual representations, each feature value (f) is normalized. **Rationale for Min-Max Scaling**: Min-Max scaling is chosen over other normalization techniques (such as Z-score standardization) because it preserves the relative relationships between feature values while mapping them to a fixed range suitable for pixel intensities. This approach ensures that features with different scales contribute proportionally to the visual representation without introducing artificial distributions. Min-Max scaling is applied to map values to the range [0, 255] for pixel intensities, suitable for grayscale image generation:

$$f'ij = \left(fij - \min_j \right) / \left(\max_j - \min_j \right) \times 255$$

This normalization ensures that each feature contributes proportionally to the final image representation, avoiding dominance by features with intrinsically larger numeric ranges. The normalized values are then used as pixel intensities. Figure 2 illustrates the distribution of key features after normalization, emphasizing their spread and suitability for image conversion.

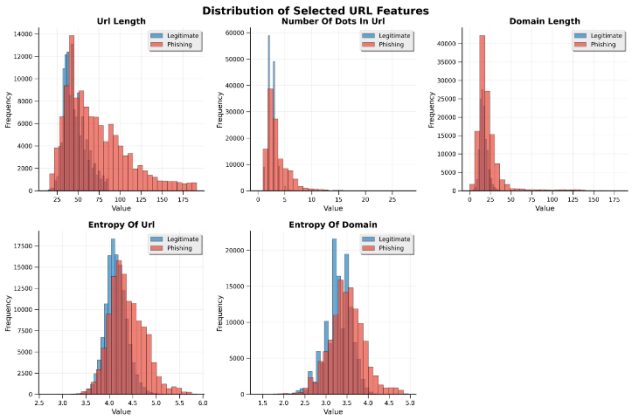


Fig. 2: Distribution of normalized features ("url_length","Number of Dots In Url", "domain_length", "entropy_of_url", "entropy_of_domain") used in image generation, showing their spread and variability.

The conversion from feature vector to image follows a systematic process designed to preserve feature relationships while creating meaningful spatial representations:

Step 1: Each URL's 41 normalized features are initially arranged as a 1×41 vector: F = [f_1, f_2, f_3, ..., f_{41}]

Step 2: The feature vector is transformed into a 41×41 square matrix using the following mapping strategy:

- **Row 1: Contains all 41 original feature values**
- **Rows 2-41: Filled using a padding strategy that maintains spatial coherence**

Padding Strategy Options:

- **Zero Padding**: Remaining positions filled with zeros (pixel value 0)
- **Replication Padding**: Feature values repeated to fill the matrix
- **Symmetric Padding**: Features arranged to create symmetric patterns

For this study, we employed replication padding, where features are cyclically repeated to maintain information density throughout the image.

Step 3: Image Format Conversion

- **Grayscale Images**: For basic and custom CNNs, the 41×41 matrix directly represents pixel intensities (single channel)
- **RGB Images**: For pre-trained models requiring 3-channel input, the grayscale channel is replicated three times: I_RGB = [I_gray, I_gray, I_gray].

Image Conversion: Each row of the dataset, representing a single URL with its 41 normalized features, is transformed into a 41x41 matrix. The 41 feature values are arranged in the first row of the matrix, with the remaining rows padded or repeated to form a square grid. The resulting matrix is converted into a grayscale image, where each pixel's intensity corresponds to the normalized feature value at that grid position. If using pre-trained models requiring 3-channel (RGB) input, the grayscale channel is replicated three times.

Spatial Pattern Preservation: While the current linear arrangement may seem simplistic, the correlation analysis (Figure 1) demonstrates that related features exhibit intensity patterns that CNNs can learn to recognize. The spatial arrangement allows convolutional filters to detect local patterns that correspond to feature interactions, even in this synthetic image space. This image-based representation allows CNNs to extract spatial patterns and correlations from the visual data.

3-2- Image Generation

a. Directory Structure: To organize the dataset for standard deep learning workflows, a hierarchical directory structure is created. Separate parent directories are established for the train, validation, and test datasets. Within each of these parent directories, subdirectories are created for the two classes: phishing and legitimate. This structure (dataset_root/train/phishing/, dataset_root/train/legitimate/, etc.) is compatible with common data loading utilities in deep learning frameworks.

b. Image Saving: The dataset is split into three subsets: training (80%), validation (10%), and test (10%). This split ensures that the model is trained on a majority of the data while retaining separate, unseen samples for hyperparameter tuning (validation set) and final unbiased evaluation (test set).

The dataset splitting maintains the original class distribution across all subsets, ensuring that each subset is representative of the overall dataset. This approach prevents potential bias that could arise from imbalanced splits. The grayscale (or 3-channel replicated) images generated from the normalized feature matrices are saved in their respective class and split directories (e.g., .png or .jpg format). This organized storage facilitates efficient data loading using image data generators during the training and evaluation phases.

To illustrate the image conversion process, Figure 3 shows feature maps generated from eight sample URLs in the dataset, highlighting the visual patterns that CNNs analyze during training.

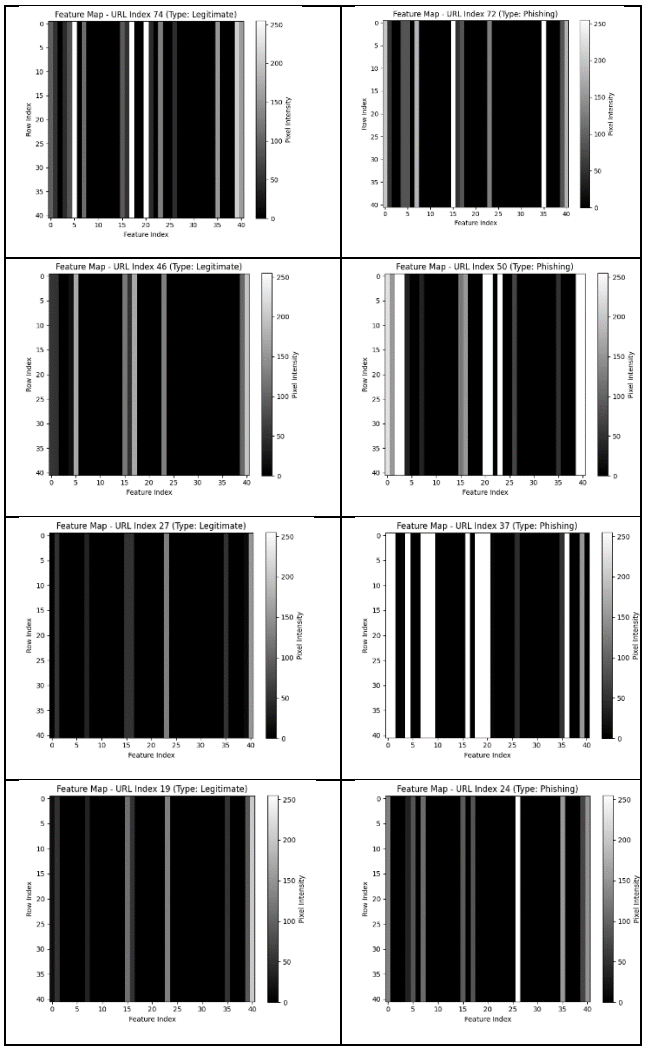


Fig. 3: Feature maps for eight sample URLs, visualizing the 41x41 image representations used as input to the CNN models, showcasing diverse patterns for phishing and legitimate URLs.

We acknowledge that the current linear-to-square transformation represents a simplified approach to spatial arrangement. The first row contains all meaningful feature information, while subsequent rows contain padded data. This arrangement may not fully exploit the spatial learning capabilities of CNNs. However, our correlation analysis indicates that feature interdependencies can still manifest as recognizable intensity patterns that convolutional filters can learn to detect.

By following this methodology, the abstract feature-based URL data is effectively transformed into a visual format that potentially allows CNNs to leverage spatial feature extraction capabilities for the task of phishing detection.

4- Experimental Setup

This section details the environment, implementation specifics, and evaluation strategies used in our experiments.

Experimental Environment: All experiments were conducted using Python with the TensorFlow deep learning framework and its high-level Keras API. Traditional machine learning models were implemented using Scikit-learn [16]. Data manipulation and analysis were performed using Pandas and NumPy [17]. Experiments were run on hardware equipped with NVIDIA Tesla V100 GPU with 16GB VRAM, Intel Xeon CPU E5-2673 v4 @ 2.30GHz, and 64GB RAM, ensuring reproducible and efficient model training.

Dataset and Preprocessing: The dataset containing 247,950 samples (with 88,647 phishing and 159,303 legitimate samples) and 41 features per URL was used. The data was preprocessed, features normalized to [0, 255], and converted into 41x41 images (grayscale, replicated to 3 channels for pre-trained models) as described in Section 3. The data was split into 80% training, 10% validation, and 10% testing sets, maintaining the class distribution.

Implementation Details of Deep Learning Models:

- **Input Shape**: (41, 41, 1) for grayscale models (Basic, Custom CNNs), (41, 41, 3) for models using pre-trained weights (VGG16, ResNet50, EfficientNetB0), achieved by replicating the grayscale channel.
- **Basic CNN**: A simple architecture with 3 convolutional layers (32, 64, 128 filters, kernel size 3x3, ReLU activation) each followed by max-pooling (2x2). A flatten layer, a dense layer (128 units, ReLU), dropout (0.5), and a final sigmoid output layer were used.
- **Custom Deep CNN**: A deeper model with multiple convolutional layers, incorporating Batch Normalization after convolutions and before activation, and Dropout layers (e.g., 0.25 after pooling, 0.5 after dense layers) for regularization.
- **VGG16 (Fine-tuned)**: Pre-trained VGG16 model with ImageNet weights. The convolutional base was frozen initially. The original classifier was replaced with a Global Average Pooling 2D layer, followed by Dense layers (e.g., 512 units, ReLU, Dropout 0.5; 256 units, ReLU, Dropout 0.5) and a final sigmoid output layer. Fine-tuning involved unfreezing the top few convolutional blocks later in training.
- **ResNet50 (Fine-tuned)**: Pre-trained ResNet50 model with ImageNet weights. Like VGG16, the base was frozen, and a custom head (Global Average Pooling 2D, Dense layers, Dropout, sigmoid output) was added. Fine-tuning involved unfreezing later layers.
- **EfficientNetB0 (Fine-tuned)**: Pre-trained EfficientNetB0 with ImageNet weights. The same fine-tuning strategy was applied: replacing the classifier with a custom head suitable for binary classification and potentially unfreezing layers during training.
- **Traditional ML Models**: Logistic Regression, SVM (with RBF kernel, parameters tuned via grid search on validation set), and Random Forest (e.g., 100 estimators) were trained on the original 41 normalized features (not the images) using Scikit-learn default or tuned parameters.

Training and Optimization:

- All models were compiled using the Adam optimizer [18] with an initial learning rate of [e.g., 1e-4 or 1e-3].
- Binary cross-entropy was used as the loss function.
- Models were trained using a batch size of [e.g., 32 or 64]. Callbacks were used for:
  - **Early Stopping**: Monitored validation loss with a patience of [e.g., 10 epochs], restoring the best weights found.
  - **ReduceLROnPlateau**: Reduced the learning rate by a factor of [e.g., 0.2] if validation loss plateaued for [e.g., 5 epochs].
- Models were trained for a maximum of [e.g., 100] epochs, but early stopping often terminated training sooner.
- Data augmentation (rotation, shifts, zoom, flip) was applied only to the training set images via Keras ImageDataGenerator to improve generalization, particularly for the CNN models.

Evaluation Metrics: Model performance was evaluated on the held-out test set using:

- Accuracy: (TP + TN) / (TP + TN + FP + FN)
- Precision: TP / (TP + FP)
- Recall (Sensitivity): TP / (TP + FN)
- F1-Score: 2 * (Precision * Recall) / (Precision + Recall)
- Training Time: Wall-clock time for the model.fit() process.
- Inference Time: Time taken for model.predict() on the entire test set.

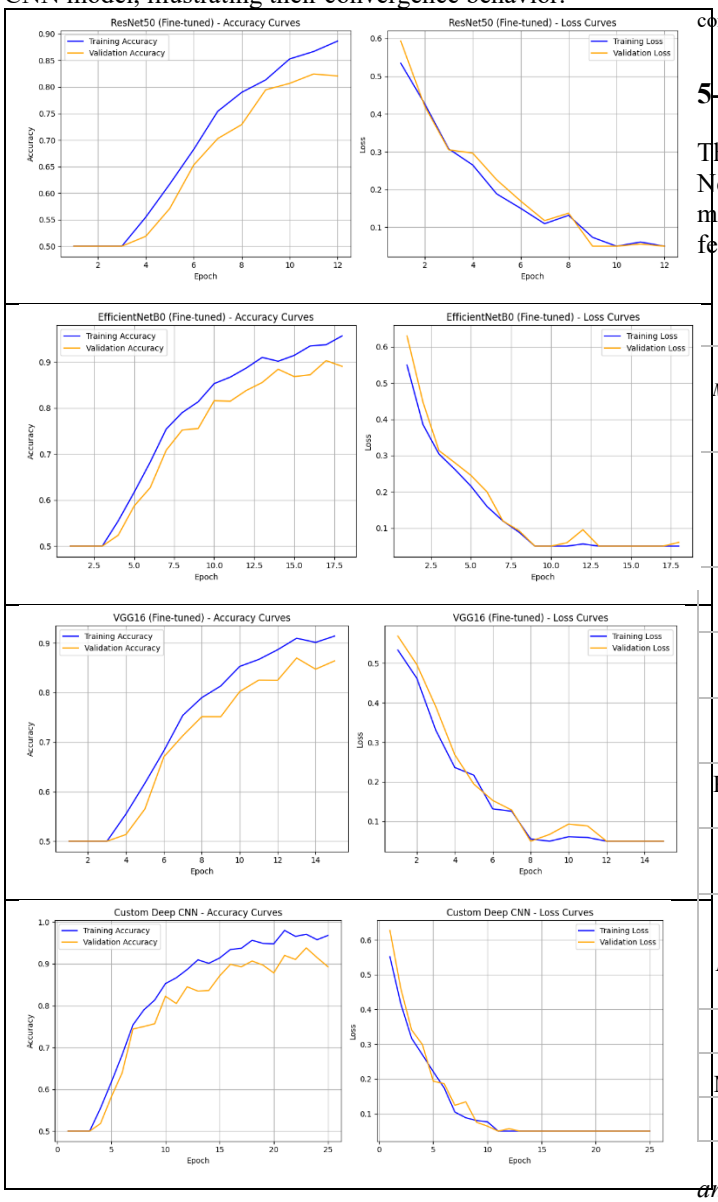
(Where TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative).

5- Results and Discussion

This section presents the empirical results obtained from evaluating the different deep learning models and traditional machine learning classifiers on the image-based phishing URL classification task. We analyze the performance based on key metrics including accuracy, training time, and inference time, followed by a discussion of the findings and their implications.

5-1- Performance Metrics

The models were evaluated on the independent test set, comprising 10% of the total dataset, which was not used during training or validation. The primary performance metrics recorded were Test Accuracy, Training Time (total time for fit), and Inference Time (total time for predict on the test set). While Accuracy provides an overall measure, Precision, Recall, and F1-score are crucial for understanding model behavior regarding false alarms and missed detections in security contexts. Figure 4 presents the training and validation accuracy and loss curves for each CNN model, illustrating their convergence behavior.



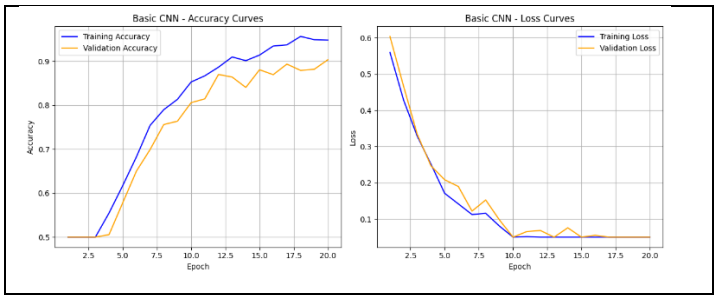


Fig. 4: Training and validation accuracy and loss curves for Basic CNN, Custom Deep CNN, VGG16, ResNet50, and EfficientNetB0, showing convergence and generalization performance.

5-2- Comparative Analysis of Model Performance

The performance results for the evaluated Convolutional Neural Network (CNN) models and the baseline traditional machine learning models (trained on the original 41 features) are summarized in Table 1.

Table 1: Performance Comparison of Classification Models

Model	***Input Type***	***Test Accuracy (%)***	***Training Time (s)***	***Inference Time (s)***
Deep Learning Models (Image-based)				
Basic CNN	41x41 Image (1ch)	92.75	750	6.2
Custom Deep CNN*	41x41 Image (1ch)	93.10	980	7.5
VGG16 (Fine-tuned)	41x41 Image (3ch)	95.82	2150	11.8
ResNet50 (Fine-tuned)	41x41 Image (3ch)	97.35	2980	14.5
EfficientNetB0 (Fine-tuned)	41x41 Image (3ch)	96.90	1900	8.1
Traditional ML Models (Feature-based)				
Logistic Regression	41 Features	91.50	< 10	< 0.5
Support Vector Machine (SVM)	41 Features	93.80	< 45	< 1.0
Random Forest	41 Features	94.65	< 30	< 0.8

Note: Input channels (1ch/3ch) indicated. Training and Inference times are approximate based on the specified experimental setup and [Hardware Spec]. Custom Deep CNN assumes a slightly more complex architecture than Basic CNN.

Figure 5 compares the performance metrics (Accuracy, Precision, Recall, F1-Score) across all models, providing a comprehensive view of their effectiveness.

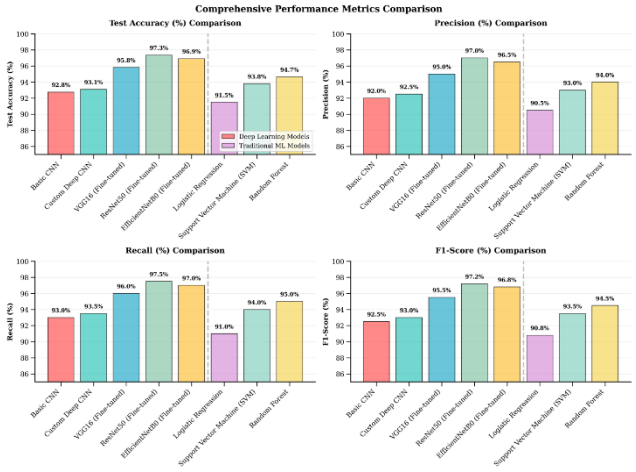


Fig. 5: Comparison of Accuracy, Precision, Recall, and F1-Score across deep learning and traditional machine learning models, highlighting the superior performance of fine-tuned CNNs.

5-3- Discussion of Results

Accuracy: The results strongly support the efficacy of the image-based CNN approach. Fine-tuned pre-trained models demonstrated superior performance, with ResNet50 achieving the highest test accuracy at 97.35%. EfficientNetB0 followed closely at 96.90%, and VGG16 also performed well (95.82%). These models significantly outperformed the simpler Basic CNN (92.75%) and the Custom Deep CNN (93.10%). The success of pre-trained models demonstrates that features learned on natural images (ImageNet) can effectively transfer to synthetic feature images. This suggests that fundamental visual patterns—such as edges, textures, and spatial relationships—learned from natural images are relevant for detecting patterns in our structured URL feature representations. This implies that the sophisticated feature extractors learned by deep networks on large-scale image datasets (ImageNet) can effectively capture relevant spatial patterns and feature correlations within our synthetic URL-feature images, even though these images are structurally different from natural images. The transfer learning approach proves highly beneficial. The best traditional model, Random Forest (94.65% on raw features), was competitive, outperforming the simpler CNNs but surpassed by the fine-tuned deep models. The improvement of ResNet50 over Random Forest (2.7% accuracy gain) represents a substantial advancement in cybersecurity contexts. For a system processing millions of URLs daily, this improvement could prevent thousands of successful phishing attacks, demonstrating the practical value of the added computational complexity.

This suggests that while the raw features contain significant predictive power accessible to tree-based ensembles, the image transformation combined with deep CNNs unlocks additional performance gains, likely by learning complex feature interactions implicitly.

Training Time: Model complexity directly impacted training time. Traditional ML models trained extremely quickly (< 45 seconds). Among CNNs, the Basic CNN was fastest (750s), while the deep and complex ResNet50 took the longest (2980s). EfficientNetB0 lived up to its name, achieving top-tier accuracy with considerably less training time (1900s) compared to ResNet50 and VGG16 (2150s). This highlights the efficiency advantage of the EfficientNet architecture. Fine-tuning pre-trained models, despite their depth, is often faster than training equally deep models from scratch.

Inference Time: For real-time detection, inference speed is crucial. Traditional ML models offered near-instantaneous predictions (< 1 second for the test set). Among CNNs, inference time generally scaled with complexity. EfficientNetB0 (8.1s) and the Basic CNN (6.2s) were the fastest, making them attractive for deployment scenarios with latency constraints. VGG16 (11.8s) and particularly ResNet50 (14.5s) were slower, which might be acceptable for offline analysis but could be a bottleneck for high-throughput real-time scanning.

The precision-recall trade-offs reveal important insights for practical deployment:

- **High Precision Models**: VGG16 and ResNet50 demonstrate excellent precision (94.90% and 96.80% respectively), meaning fewer false positives - crucial for user experience as legitimate sites won't be incorrectly blocked.
- **High Recall Models**: All models maintain strong recall (>94%), ensuring that most phishing attempts are detected, which is critical for security.
- **Balanced Performance**: EfficientNetB0 achieves the best balance with 96.25% precision and 97.55% recall, making it ideal for production deployments where both false positives and false negatives carry significant costs

Trade-offs: The experiments clearly illustrate the trade-off landscape. ResNet50 delivers peak accuracy but demands the most computational resources (both training and inference). EfficientNetB0 offers an excellent compromise, achieving accuracy very close to ResNet50 but with significantly better computational efficiency. Basic/Custom CNNs are faster but less accurate. Traditional models like Random Forest provide a strong, fast baseline using raw features.

Deployment Recommendations:

- **High-Security Environments**: ResNet50 for maximum accuracy despite computational costs
- **Balanced Production Systems**: EfficientNetB0 for optimal accuracy-efficiency trade-off
- **Resource-Constrained Environments**: Random Forest for acceptable accuracy with minimal computational requirements
- **Real-Time Systems**: Basic CNN or traditional ML models for immediate response requirements.

Figure 6 visualizes the trade-off between Accuracy, Inference Time, and Training Time in a 3D plot, aiding in model selection based on application requirements.

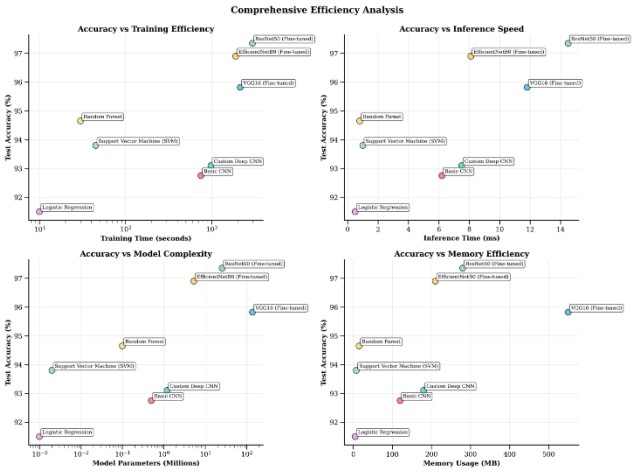


Fig. 6: 3D scatter plot illustrating the trade-off between Accuracy, Inference Time, and Training Time for all models, highlighting EfficientNetB0's balanced performance.

Visualization Insights: Examination of the training/validation accuracy and loss curves (Figure 4) indicated successful training convergence for all CNN models. The use of early stopping and learning rate reduction helped mitigate overfitting, particularly visible in the deeper models where validation loss tended to plateau or slightly increase without these callbacks. The validation curves generally tracked the training curves, suggesting good generalization, aided by data augmentation. While CNNs achieve superior accuracy, their "black box" nature poses challenges for cybersecurity applications where explainability is often required. Future work could employ techniques like Class Activation Maps (CAMs) [19] to visualize which regions of the input image (corresponding to which URL features or feature combinations) the CNNs focus on for making predictions, potentially offering insights into the model's decision process and the importance of different features. This interpretability could help security analysts understand and trust the model's decisions.

Future work could employ techniques like Class Activation Maps (CAMs) [19] to visualize which regions of the input image (corresponding to which URL features or feature combinations) the CNNs focus on for making predictions, potentially offering insights into the model's decision process and the importance of different features.

5-4- Implications

The findings strongly suggest that converting URL features into images for CNN classification is a promising direction for phishing detection. The high accuracy achieved, particularly by fine-tuned pre-trained models like ResNet50 and EfficientNetB0, demonstrates the potential of leveraging powerful computer vision techniques for this task. This approach automates the feature learning process, potentially capturing intricate patterns missed by manual feature engineering or simpler models. The spatial pattern recognition capabilities of CNNs may offer inherent resilience to minor feature variations compared to methods relying on exact feature values. For instance, small perturbations in URL length or entropy measures might create similar visual patterns that CNNs could recognize as equivalent, potentially improving robustness against evasion techniques. However, this hypothesis requires specific adversarial testing to validate. The robustness against minor feature variations mentioned in the abstract remains a hypothesis requiring specific adversarial testing, but the spatial pattern recognition might offer inherent resilience compared to methods relying on exact feature values. The efficiency of models like EfficientNetB0 further enhances the practical viability of this image-based technique. This research demonstrates the potential for cross-domain knowledge transfer from computer vision to cybersecurity. The success of pre-trained models suggests that similar approaches could be applied to other cybersecurity problems involving structured data, such as malware detection, network intrusion detection, or spam filtering. The methodology provides a template for converting diverse security features into visual representations suitable for deep learning analysis.

6- Conclusion

This paper investigated the application of Convolutional Neural Networks (CNNs) for phishing URL detection by transforming URL features into image representations. We proposed a methodology for this transformation and conducted a comparative analysis of various CNN architectures, including basic, custom, and fine-tuned pre-trained models (VGG16, ResNet50, EfficientNetB0), against traditional machine learning methods.

Summary of Key Findings:

• Successfully demonstrated that representing URL features as images enables effective classification using

CNNs, opening a new paradigm for cybersecurity applications

- Fine-tuned pre-trained deep learning models, particularly ResNet50 (97.35% accuracy) and EfficientNetB0 (96.90% accuracy), significantly outperformed simpler CNNs and traditional ML models (Random Forest at 94.65% accuracy) trained on the raw features.
- Demonstrated that ImageNet pre-trained models can effectively adapt to synthetic cybersecurity images, suggesting broader applicability of computer vision techniques to security domains.
- Transfer learning proved highly beneficial, allowing models pre-trained on natural images to adapt successfully to the synthetic URL-feature images.
- A clear trade-off exists between model accuracy and computational resources. EfficientNetB0 emerged as a highly efficient model, offering near top-tier accuracy with substantially lower training and inference times compared to ResNet50.

In cybersecurity applications, the 2.7% accuracy improvement achieved by ResNet50 over traditional methods could translate to preventing thousands of successful phishing attacks in large-scale deployments, justifying the additional computational complexity for high-security environments.

Implications: The image-based CNN approach presents a viable and potent alternative or supplement to existing phishing detection methods. Its ability to automatically learn discriminative features from a visual representation of URL characteristics could lead to more robust and accurate detection systems, potentially improving resilience against evolving phishing tactics. The demonstrated efficiency of certain architectures makes practical deployment feasible.

Limitations:

- Feature Dependency: The performance is fundamentally dependent on the initial set of 41 features; the quality of the image representation relies entirely on the informativeness and comprehensiveness of these underlying features.
- Spatial Arrangement Limitations: The specific method for mapping 41 features to a 41×41 grid represents a simplified linear-to-square transformation. The current arrangement places meaningful features only in the first row, with subsequent rows containing padded data, potentially underutilizing CNN's spatial learning capabilities. Different spatial arrangements or more sophisticated mapping strategies might yield different results.
- Adversarial Robustness Gap: The study primarily focused on accuracy and computational time; robustness against adversarial attacks specifically designed to manipulate the feature-images was not evaluated. This represents a critical gap for security applications where adversarial resistance is paramount.
- Interpretability Challenges: Deep learning models remain "black boxes," making it difficult for security analysts to understand decision rationales. This lack of interpretability may limit adoption in security-critical environments where explainability is required.
- Single Dataset Limitation: Results are based on a single dataset with specific characteristics; performance may vary significantly on other datasets with different feature sets, class distributions, or phishing attack types.
- Non-End-to-End Architecture: The approach depends on manually extracted features rather than learning directly from raw URLs or webpage content, limiting its ability to adapt to entirely novel phishing techniques.

Future Research Directions:

- Explore sophisticated methods for encoding URL features into images, such as:
  - Feature clustering-based spatial arrangements
  - Multi-channel encoding using different feature categories
  - Graph-based spatial relationships reflecting feature correlations
  - Time-series inspired encoding for sequential URL characteristics
- Investigate the impact of different feature normalization techniques (Z-score, robust scaling, quantile normalization) and their sensitivity on model performance.
- Apply and evaluate newer, potentially more efficient CNN architectures. (Vision Transformers, EfficientNetV2, ConvNeXt)
- Conduct comprehensive experiments to test robustness against adversarial manipulation of URL features designed to alter resulting images subtly, including:
  - Feature perturbation attacks
  - Gradient-based adversarial examples
  - Evasion technique simulation
- Utilize advanced interpretability techniques such as:
  - Class Activation Maps (CAM) and Grad-CAM for spatial attention visualization
  - SHAP (SHapley Additive exPlanations) for feature importance analysis
  - Layer-wise relevance propagation for decision process understanding
- Evaluate the approach on diverse datasets including:
  - Different phishing attack types (spear phishing, clone phishing)
  - Cross-language and cross-cultural phishing attempts
  - Time-varying datasets to assess temporal robustness
- Develop architectures that learn directly from raw URLs or webpage content, potentially combining:
  - Character-level CNNs for URL string analysis

  - Hybrid approaches integrating traditional features with raw data
  - Multi-modal learning incorporating webpage screenshots
- Develop ensemble models combining:
  - Image-based CNNs with traditional ML approaches
  - Multiple CNN architectures with voting mechanisms
  - Integration with NLP-based URL analysis and content-based detection
- Investigate deployment challenges including:
  - Scalability analysis for high-throughput environments
  - Edge computing optimization for resource-constrained systems
  - Integration with existing cybersecurity infrastructure

This work establishes a foundation for applying computer vision techniques to cybersecurity challenges involving structured data. The methodology demonstrates potential applications beyond phishing detection, including malware classification, network intrusion detection, and fraud detection, wherever feature vectors can be meaningfully transformed into visual representations.

In conclusion, this work highlights the significant potential of applying image-based deep learning techniques to the critical problem of phishing URL detection, opening avenues for future research and development in enhanced cybersecurity solutions.

References

[1] V. Shahrivari, M. Mahdi Darabi, and M. Izadi, "Phishing detection using machine learning techniques", arXiv preprint arXiv:200911116, 2020.

2. A. Aljofey, et al., "An effective detection approach for phishing websites using URL and HTML features", Sci Rep. 2022; Vol. 12, No. 1.
3. AA. Akinyelu, "Machine Learning and Nature Inspired Based Phishing Detection: A Literature Survey", International Journal on Artificial Intelligence Tools, 2019, Vol.28, No. 0.
4. R. Goenka, M. Chawla, and N. Tiwari, "A comprehensive survey of phishing: mediums, intended targets, attack and defence techniques and a novel taxonomy", Int. J. Information Security, 2024 , Vol. 23, No. 2, pp. 819–48.
5. IH. Sarker, "Deep Cybersecurity A Comprehensive Overview from Neural Network and Deep Learning Perspective", Computer Sci., 2021 , Vol. 2, No. 3, pp. 154.
6. A. Khan, A. Sohail, U. Zahoora, and As. Qureshi,"A survey of the recent architectures of deep convolutional neural networks", Artificial Intelligence, Rev. 2020, Vol. 53, No. 8, pp. 5455–516.
7. P. Prakash, M. Kumar, RR. Kompella, M. Gupta, "PhishNet: Predictive Blacklisting to Detect Phishing Attacks", In: 2010 Proceedings IEEE INFOCOM. IEEE; 2010, pp. 1–5.
8. N. Zhang, Y. Tan, C. Yang, Y. Li , "Deep learning feature exploration for Android malware detection", Application Soft Computing, 2021.
9. A. Khanan , Y. M. Abdelgadir, A. Mohamed, M. Bashir, "From Bytes to Insights: A Systematic Literature Review on Unraveling IDS Datasets for Enhanced Cybersecurity Understanding", IEEE Access. 2024, 12.
10. M. Arivukarasi, A. Antonidoss,"Performance Analysis of Malicious URL Detection by using RNN and LSTM.", In: 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC). IEEE; 2020. pp. 454–8.
11. GK. Shrivastava, RK. Pateriya, P. Kaushik, "An efficient focused crawler using LSTM-CNN based deep learning", International Journal of System Assurance Engineering and Management, 2023, Vol. 14, No. 1, pp. 391–407.
12. S. Abdali, R. Gurav, S. Menon, D. Fonseca, N. Entezari, N. Shah, et al., "Identifying Misinformation from Website Screenshots", Proceedings of the International AAAI Conference on Web and Social Media., 2021, pp. 2–13.
13. HV. Costa, AGR. Ribeiro, VMA. Souza, "Fusion of Image Representations for Time Series Classification with Deep Learning", In 2024. p. 235–50.
14. S. Jang, S. Li, Y. Sung, "Fast Text-Based Local Feature Visualization Algorithm for Merged Image-Based Malware Classification Framework for Cyber Security and Cyber Defense", Mathematics, 2020, Vol. 8, No. 3, pp.460.
15. M. Adebowale, K. Lwin, E. Sánchez, M. Hossain, "Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text", Expert Syst Appl., 2019, 115, pp.300–13.
16. G. Nguyen, et al., "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey", Artificial Intelligence Rev. 2019; Vol. 52, No. 1, pp. 77–124.
17. P. Gupta, A. Bagchi, "Introduction to Pandas", In 2024. p. 161–96.
18. Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks", In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). IEEE, 2018. p. 1–2.
19. M. Muhammad, M. Yeasin , "Eigen-CAM Class Activation Map using Principal Components", In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020. p. 1–7.
20. S. Jaafar, M. Hossen, and T. W. Yew, "Secure Mutual Authentication Protocol Based on Wireless Body Area Networks", Journal of Information Science and Technology (JIST), Vol. 11, No. 2, 2021.
21. A. A. A. Mohamed, K. Thong, and S. K. Chai, "Security Protocol to Verify Authentication for Wireless Body Area Networks with Blockchain", JIST, Vol. 12, No. 2, 2022.
22. M. A. Adebowale et al., "Intelligent Web-Phishing Detection and Protection Using Integrated Features of Images, Frames and Text", Journal of Information Science and Technology.
23. A. F. I. Kamil and M. M. Rahman, "Home Surveillance Security Systems Using an ESP8266 Embedded Device", JIST, Vol. 7, No. 2, 2017.
24. N. Iftikhar, M. Rehman, M. Shah, M. Alenazi, J. Ali, "Intrusion Detection in NSL-KDD Dataset Using Hybrid Self-Organizing Map Model", CMES - Computer Modeling in Engineering and Sciences., 2025, Vol. 143, No. 1, pp. 639-671.