SCIENTIFIC CULTURE
www.sci-cult.com

SCIENTIFIC CULTURE, Vol. 12, No. 4, (2026), pp. 7272-7289
Open Access. Online & Print

DOI: 10.5281/zenodo.12426738

DETECTING MALICIOUS URLS USING HILBERT SPACE DEEP NEURAL NETWORKS FEATURES COUPLED WITH REGION-BASED SUPPORT MAPPING TECHNIQUES

Maruti Patil [1*], Sangram Patil [1], Deepali Jadhav[1], Shobha Patil [1], Yogesh Bodhe[1], Shreeshail Chavan [2], Ishwari Nandargi [2]

[1] Lecturer, Department of Information Technology, Government Polytechnic Kolhapur, 416006, India. patilmaruti16@gmail.com,
[1] Professor Department of Computer Science & Engineering, D.Y. Patil Agriculture & Technical University, Talsande, Kolhapur, 416112, India sangrampatil@dyp-atu.org,
[1] Associate Professor, Department of Computer Science and Engineering, Kolhapur Institute of Technology's College of Engineering Kolhapur, Maharashtra 416234 deepkjadhav80@gmail.com
[1] Assistant Professor, Department of Computer Engineering and Engineering D. Y. Patil College of Engineering and Technology, Kolhapur,416006 shobhapatil.dypcet@dypgroup.edu.in
[1] Lecturer, Department of Information Technology, Government Polytechnic Pune, 411016, India. bodheyog@gmail.com
[2] Student, Department of Electronics and Telecommunications, Pune Institute of Computer Technology Pune, 411016, India. chavan.shreeshail.2005@gmail.com
[2] Student, Department of Computer Science & Engineering, Walchand College of Engineering, Sangli, 416415, India. ishwarinandargi@gmail.com.

Received: 23/12/2025
Accepted: 27/02/2026

Corresponding Author: Maruti Patil (patilmaruti16@gmail.com)

ABSTRACT

Malicious URLs often host unsolicited content and serve as conduits for cybercrimes. Detecting them promptly is crucial. Traditionally, this task relies on blacklists, which are not exhaustive and furthermore, they cannot identify newly generated malicious URLs. To overcome this limitation, recent years have seen numerous initiatives to utilize Machine Learning for malicious URL detection. This paper presents an innovative approach to detecting malicious activity using Region-Based Support Mapping. The process begins with pre-processing the input data, which involves tokenization, special character handling, and word embedding. Significant features are then extracted using a Hilbert Space-based Deep Neural Network (HS-DNN). Finally, these features are input into a Region-Based Support Mapping Long Short-Term Memory (RSM-LSTM) model to classify URLs as either normal or malicious. Our proposed model outperforms existing models in terms of accuracy, precision, recall, F1 score, ROC, computation time, AUC, true positive rate, and false positive rate.

KEYWORDS: Cybersecurity,Malicious URL Detection,Feature Extraction, Tokenization, Deep Neural Network (DNN), URL Classification, Hilbert space-based deep neural network(HS-DNN), Region-based support mapping-Long short term memory (RSM-LSTM).

1. Introduction

The daily surge in internet users is driven by a multitude of factors, including the growing demand for computing devices, the expansion of cloud storage, the proliferation of online businesses, and the increasing affordability and speed of internet access, even in rural regions. However, this heightened connectivity comes with an increased risk of cyber threats. Internet-connected systems are now more vulnerable to a range of attacks, such as financial fraud, malware injection, spam advertising, and data breaches. Hackers frequently exploit malicious websites as a primary attack vector, luring unsuspecting users into clicking on dangerous URLs. This escalating threat landscape has spurred the development of advanced tools designed to assess and verify the safety of unfamiliar or suspicious websites, offering users enhanced protection against online threats [1,2]

The most common type of social engineering and cyber-attack is phishing. In these scams, phishers prey on gullible internet users, tempting them into disclosing private information for illicit gain. Users need to keep an eye out for phishing websites and keep a blacklist of them that needs to be updated frequently in response to new threats to prevent phishing efforts [3]. In present days, the use of deep neural network techniques and machine learning, allows for the early identification of phishing websites. Online consumers are still tricked into disclosing private information on phishing websites despite these safeguards. Cybersecurity dangers fall into a different category called social engineering assaults, which take advantage of behavioural patterns and human error instead of software flaws. Social engineers are skilled at manipulating their victims to obtain sensitive information and important data. Techniques include tricking people into opening malicious attachments, clicking on spoof URLs, or inadvertently disclosing private information [4].

About one in ten websites contains dangerous information, according to a Symantec Internet security report. The perpetrators of these websites usually work by creating malicious URLs and then waiting for the victim to click. These URLs can initiate a drive-by-download when clicked, which gives the attacker access to install files on the victim's device that can carry out a variety of malicious tasks, like displaying intrusive advertisements (adware), stealing confidential data (spyware), or jeopardizing the victim's data integrity [5]. Phishing is another type of URL-based attack that doesn't require direct downloads to the user's device. Rather, it tricks users into disclosing private information that attackers can use against them in later attacks. In light of these dangers, a system that evaluates a website's legitimacy without requiring a visit is desperately needed to prevent possible attacks. This means that customers can receive proactive protection against various risks by implementing machine learning (ML) [6].

The importance of phishing URL detection is growing, which highlights the requirement for an AI-driven independent detection system. Researchers have proposed a number of methods for identifying spoofing Uniform Resource Locators (URLs). The simplest of these tactics involves adding URLs to a whitelist or blacklist [7]. The completeness of the URL list has a major impact on how effective this detection is. The system will either allow or deny access based on whether the URL is mentioned. Such lists must be updated frequently, either automatically or manually, to stop freshly produced phishing URLs from slipping past detection. Heuristics provide another approch that provides a broader detection strategy than blacklisting. This method works using present guidelines that are taken from lists of already-existing URLs. Nonetheless, it necessitates a thorough investigation into the various techniques used to generate phishing URLs [8, 9].

Researchers use models from Deep Learning (DL) and Machine Learning (ML) to optimize the process of using data to create rule-based systems. Machine learning algorithms are information-driven computer methods that can automatically acquire new skills and improve with additional training data. Deep learning is one aspect of machine learning that is especially interesting [10, 11]. Feature selection plays a critical role in machine learning in order to build robust models that function effectively on unknown input. The training data that is employed determines the quality of the machine learning insights. The main components of a dataset are its features; they are used as independent variables in machine learning models. Several studies have attempted to detect phishing URLs using different feature sets extracted from different datasets [12, 13]. Because lexical properties of phishing URLs are easy to use and safe, some researchers choose to use them. On the other hand, some people research the quality of internet content, which is risky as phishing sites occasionally have harmful viruses that can interfere with systems. Additionally, external elements like website ratings and DNS data are integrated. Some studies concentrate on visual similarity characteristics,

while many researchers select hybrid features that incorporate URL lexical data, webpage content features, external features, and others [14, 15]. The main contribution of the work is,

Problem Statement:
With few false positives, blacklist-based URL detection works well for quickly identifying known risks, but it needs to be updated frequently to stay effective. The issue of keeping protection up to date arises from the fact that cybercriminals regularly generate new URLs or employ dynamic methods to get around blacklists. Because blacklist techniques depend on a constantly updated database, they may overlook new threats, underscoring the necessity for more detection techniques.

Motivation:
The increasing demand for a proactive, scalable, and dependable malicious URL detection system that reduces user risk is what spurred our investigation. The need for machine learning/deep learning-based models that can intelligently assess URLs without depending on manual or rule-based procedures is evident given the ongoing rise in malicious instances, many of which take advantage of behavioral patterns rather than system vulnerabilities. This research aims to

- The URL dataset undergoes a comprehensive preprocessing phase, including word embedding, tokenization, and special character handling, to ensure the data is optimally prepared for analysis.
- Key features are then identified using a Hilbert Space Deep Neural Network (HS-DNN), which employs a range of convolutional window combinations to extract features at multiple levels of granularity. This approach ensures the extraction of a rich, high-quality feature set that is thoroughly optimized for subsequent analysis.
- In the final stage, the Region-Based Support Mapping - Long Short-Term Memory (RSM-LSTM) classifier is employed to accurately categorize URLs into normal or malicious classes, enhancing the system's detection capabilities

An overview of the paper's structure is provided below: Related work is discussed in Section 2. Section 3 presents a summary of the proposed techniques. Section 4 presents the experimental results, and Section 5 concludes the paper.

2. Related Work

Anwar Mustafa Hilal et al.'s Machine Learning (ML) and Deep Learning (DL) models have made it possible to create algorithms that accurately identify and classify harmful URLs [16]. For the Artificial Fish Swarm Algorithm (AFSA), the AFSADL-MURLC model is developed in this work. The AFSADL-MURLC model aims to differentiate between harmful and genuine URLs. The GloVe-based embedding of words method and data preprocessing are the first steps taken by the AFSADL-MURLC model to achieve this. Additionally, the developed vector model is subsequently subjected to the Gated Recurrent Unit (GRU) classification process in order to identify the harmful URLs. In order to improve the GRU model's efficiency, AFSA is finally applied to the proposed model. Experiments were conducted using a benchmark dataset from the Kaggle repository to validate the proposed AFSADL-MURLC approach.

Muhammad Waqas Shaukat et al. [17] used a large dataset of 20,000 internet URLs to generate a comprehensive dataset, from which 22 prominent features were extracted. Furthermore, a different dataset made up of web content is prepared for text analysis by natural language processing. Because content typically appears as images on phishing sites, the text is retrieved from those images to determine whether it is spam or real. The evaluation of the study showed how accurate and successful phishing detection was. The following techniques are used in the layered classification model: Support Vector Machine (SVM), Naïve Bayes, SVC, XGBoost, Random Forest, Multilayer Perceptron, and Linear Regression. The XGBoost algorithm surpassed other pertinent models with maximum precision and accuracy of 94% in training and 91% in testing, according to the results. During testing, the Multilayer Perceptron performed admirably as well, achieving 91% accuracy.

Shayan Abad et al. [18] are developing machine learning-based models for the efficient detection and categorization of malicious URLs in an effort to enhance cybersecurity. To effectively categorize URLs, the study combines Bayesian optimization with the use of k-nearest neighbors (KNNs), decision trees (DTs), Random Forest models (RFs), and Support Vector Machines (SVMs). Instance selection techniques including random selection, reduction of data based on locality-sensitive hashing (DRLSH), and border point extract based on BPLSH are used to increase computational efficiency. The outcomes demonstrate the effectiveness of RFs in terms of recall, precision, and high F1 scores. SVMs, on the other hand, perform competitively but require more training time.

Naya Nagy et al. [19] explored a variety of multitask and multithreading techniques in Python to efficiently train Machine Learning (ML) and Deep Learning (DL) models. Their dataset consisted of 12K records for testing and 54K records for training. The first experiment utilized sequential execution, while the subsequent four experiments investigated parallel execution strategies, including manual threading, using Python's parallel backend, multitasking with a parallel backend, and leveraging a parallel backend for handling multiple tasks and counting. The study tested four different models: Random Forest (RF), Naïve Bayes (NB), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN).

The study by Anil Kumar et al. [20] that is being presented here offers a thorough analysis of machine learning-based APT detection methods. The characteristics of APTs and the details of their attack models are described at the outset. It analyzes APTs in more detail by describing their attack strategies and tactics. This paper covers a wide range of APT attack detection methodologies, with a particular emphasis on machine learning techniques. Support Vector Machines (SVM), k-nearest neighbors (KNN), Deep Belief Networks (DBN), Decision Trees, and Convolutional Neural Networks (CNN) are taken into consideration in the context of APT detection. Each method's applicability and underlying assumptions for APT detection are assessed. The blacklist-based approach uses a large list of URLs that have already been marked as dangerous, either manually or through automated methods, to detect this issue. This technique has drawbacks even though it provides rapid detection with few false positives. On the other hand, it is essential to keep an updated database of blacklisted URLs; failing to do so could lead to the omission of newly created dangerous URLs. Furthermore, a lot of hackers don't use known malicious URLs in their subsequent attacks, which reduces the efficacy of this strategy.

Issa Qiqieh et al. [21] developed a smart system to detect online threats using machine learning and a method inspired by how hawks hunt, called Harris Hawks Optimization (HHO). They built a model called HHO-SVM, which combines this hawk-based method with a machine learning model called Support Vector Machine (SVM) to find different types of online dangers, like harmful web links (malicious URLs). Initially, the HHO method is used to pick out the most important parts of the data and improve the settings of the SVM model. This two-step process helps the system learn faster and make more accurate predictions. The model was tested on several public datasets, including one focused on harmful websites, and it performed very well. The results showed that the HHO-SVM model is reliable for real-world cybersecurity and works better than traditional machine learning methods at telling the difference between safe and dangerous websites.

Over 500,000 Facebook posts were examined by Saeed Abu-Nimeh et al. [22] to determine the prevalence of spam and harmful content on social media.They used a tool called Defensio, along with the ThreatSeeker Network and a Support Vector Machine (SVM)-based model, to check the safety of web links shared in these posts. Their results showed that about 9% of the posts were spam, and 0.3% had harmful (malicious) links. Even though the percentage of harmful links was small, they were spread out across many IP addresses and countries, showing how widespread the problem is. The system used in the study combined known patterns (signatures) and trust scores (reputation checks) to find risky content and alert users or remove it. The research highlighted that spam and harmful links are still serious problems on social networks, and showed how machine learning, especially SVM models, can help automatically detect and deal with such threats.

Omar A. Alzubi et al. [23] proposed a new hybrid model for classification called the Dynamic Programming-based Ensemble Design (DPED) algorithm.[23]. This model aims to build the best group of classifiers by using ideas from cooperative game theory and dynamic programming. The goal is to keep the model accurate while making it smaller and more efficient. First, the model uses a measure called Cohen's kappa to check how different the classifiers are from each other and to find which ones add the most variety to the group. Then, it applies a method called maximum sum increasing subsequence (MSIS) to choose the best set of classifiers. The researchers tested their model on 13 well-known datasets from the UCI repository and found that DPED performed much better than traditional ensemble methods. It not only kept or improved prediction accuracy but also used fewer classifiers, showing that it is both powerful and efficient.

In their thorough analysis of phishing website detection strategies, Rasha Zieni et al. [24] covered both conventional and more recent machine learning methods.They grouped the detection methods into three main types: blacklist-based, heuristic-based, and machine learning-based, noting a growing shift toward using smart, hybrid systems.

The survey also looked at the latest improvements in the field, such as deep learning models like CNNs, RNNs, and transformers, as well as ensemble methods and models that can resist adversarial attacks. These modern techniques have proven effective in identifying tricky or new (zero-day) phishing links. The authors also reviewed commonly used datasets, important features used in detection, and key performance measures like precision, recall, F1-score, and AUC. The paper highlights real-world challenges like phishing tricks, unbalanced data, and putting models into live use, offering valuable insights for future research and development of strong phishing detection systems.

Sanskriti Sanjay Kumar Singh et al. [25] introduced a novel approach using meta-learning to improve web security by detecting malicious URLs. Their approach combines several basic machine learning models with a meta-learner that adjusts based on how well the models perform on different types of URLs and threats. The system uses traditional techniques like Random Forest and Support Vector Machine (SVM), along with advanced ensemble methods that use optimization to boost accuracy and work well across different datasets. A key part of their method is understanding how the models relate to each other and adjusting the importance of features at the meta-level, especially to deal with imbalanced data and tricky, adversarial examples. Tests showed that their model performed better than individual classifiers, especially in detecting new (zero-day) attacks and hidden phishing links. This research shows that meta-learning is a powerful tool for creating smarter, more flexible systems to detect harmful URLs.

3. Proposed System

This paper introduces a powerful technique for detecting malicious URLs, combining Region-Based Support Mapping with Long Short-Term Memory (RSM-LSTM) and advanced feature extraction using Hilbert Space Deep Neural Networks (HS-DNN). The process begins with meticulous data pre-processing to prepare the URLs for model training, involving steps such as word embedding, tokenization, and handling of special characters. The feature extraction module leverages HS-DNN to capture abstract-level features from the URLs, employing diverse convolutional window combinations to extract features of varying granularity. This approach ensures the extraction of comprehensive, high-quality features, optimizing the feature set for robust performance. In the final stage, the proposed RSM-LSTM model accurately classifies URLs into malicious or benign categories. Figure 1 illustrates the block diagram of the proposed system.

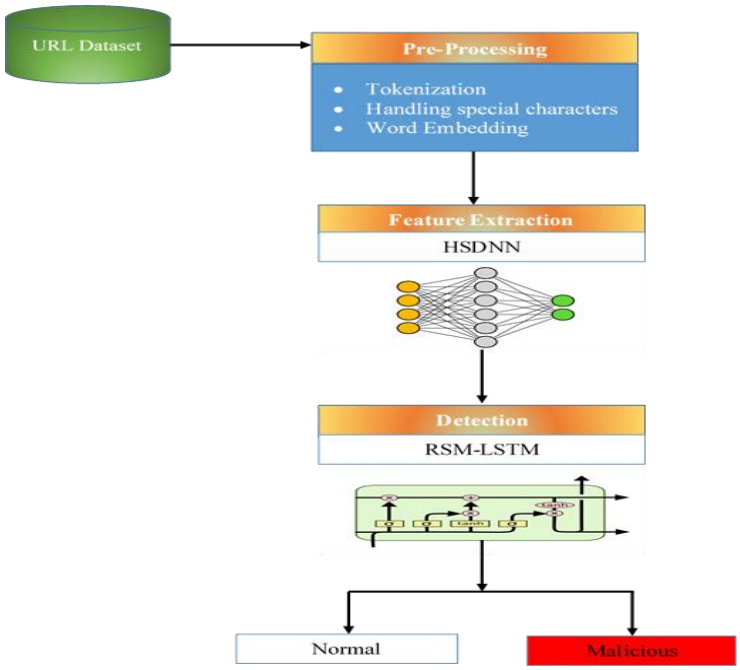

Fig 1: Schematic representation of the proposed approach

Algorithm:

Input: URL

Output: Prediction ∈ {MALICIOUS, BENIGN}

// Preprocessing

tokens = tokenize(URL)

tokens = normalize_special_characters(tokens)

```
vectors = Word2Vec(tokens, dim=100)
matrix = form_matrix(vectors)
// Feature Extraction (HS-DNN)
features = flatten(apply_pooling(apply_convolution(matrix)))
// Sequential Modeling (RSM-LSTM)
lstm_out = RSM_LSTM(reshape_for_LSTM(features))
// Classification
score = sigmoid(dense_layers(batch_normalization(lstm_out)))
// Decision
return MALICIOUS if score ≥ 0.5 else BENIGN
```

3.1 Data Pre-processing

The input data is first pre-processed to transform unprocessed data into a deep learning-ready state. Tokenization, managing special characters, and word embedding are the three phases involved in this.

3.1.1Tokenization: The preparation pipeline starts with tokenizing URLs. Tokenization is the process of breaking a string up into smaller parts called tokens. The structural components of URLs are captured by combining punctuation-based and whitespace-based word tokenization. Subdomains, pathways, arguments, and delimiters are a few examples of these tokens. This method makes it easier to quickly extract significant elements from URL strings for further examination.

3.1.1.1 Word Tokenization:

Word tokenization is the process of breaking a text up into individual words. Completing this fundamental job is often required for tasks related to natural language processing (NLP), such as information retrieval, sentiment analysis, and machine translation. The objective of word tokenization is to recognize and distinguish each word from the input text, which is usually a string of characters. To identify word boundaries precisely, this method may require managing punctuation marks, special characters, and whitespace. Divide the text into words according to punctuation or whitespace limits. Figure 2 displays the process of word tokenization.

Whitespace Tokenization: This method splits the text based on whitespace characters. It is simple but may not handle all cases effectively, especially in languages where words can be agglutinated without whitespace.

Punctuation-Based Tokenization: In this approach, the text is split based on punctuation marks (e.g., periods, commas, hyphens). This method may require additional rules to handle cases like contractions or abbreviations.

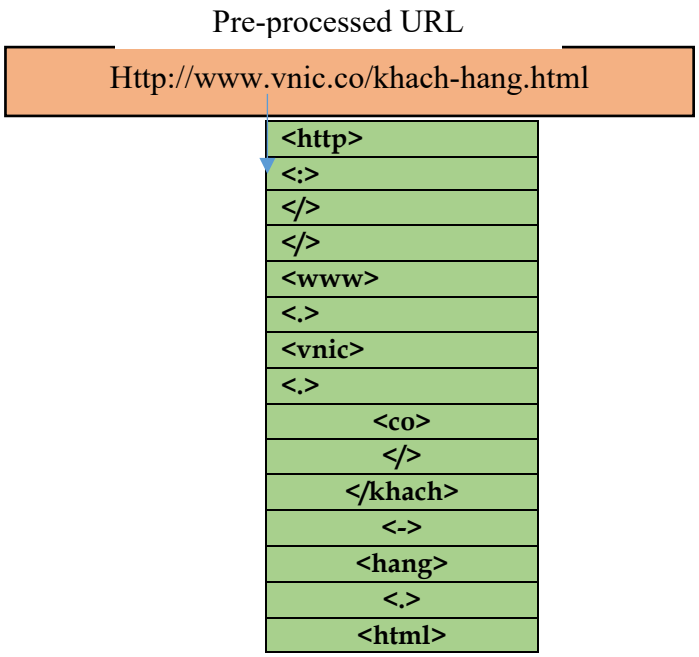

Fig2: Word tokenization

3.1.1.2 Subword Tokenization: Sub-word tokenization is a technique that divides words into smaller units called sub-words as opposed to considering each word as an indivisible unit. This

method works especially well with morphologically rich languages, uncommon or non-vocabulary words, and terminology peculiar to a given domain.

3.1.2 Handling Special Character

Special characters in text data include any characters that are not letters or digits, such as punctuation marks, symbols, whitespace characters, and other non-alphanumeric characters. Handling special characters is a crucial step in text pre-processing for natural language processing (NLP) tasks. Here are some common approaches to dealing with special characters:

Removing Special Characters: Due to their lack of significant contribution to the text's semantic meaning, special characters such as punctuation marks, symbols, and non-alphanumeric characters can typically be safely removed during pre-processing. This simplifies the text data and reduces noise that can interfere with further NLP operations.

Replacing Special Characters: Special characters can be replaced with whitespace or removed altogether. For example, punctuation marks like commas, periods, and quotation marks can be replaced with spaces, or special tokens indicating their presence can be inserted. This approach is often used in tokenization to separate words while preserving the contextual information provided by special characters.

Tokenization Strategies: When tokenizing text, special characters can be treated as separate tokens or incorporated into tokens along with adjacent alphanumeric characters. For example, "e-mail" could be tokenized as two separate tokens ("e" and "-mail") or as a single token ("email"). The particulars of the task and the properties of the text data determine which tokenization approach is best. Encoding Special Characters: Special characters can be encoded using specific encoding schemes, such as Unicode or ASCII. This ensures that special characters are represented consistently and can be processed by machine learning models without loss of information.

3.1.3 Word Embedding

Words are represented as dense vectors of real numbers using the word embedding technique in NLP, usually in a high-dimensional space. These word vectors gather semantic and syntactic data about words according to their context in a sizable corpus of textual data. Every word has a corresponding vector representation. Using examples helps people understand that operations in mathematics can be applied to words. This is a well-known example, as seen in Fig. 3: king − man + woman = queen. Here, we may create a vector representing the queen by first adding the vectors associated with the words "king" and "woman," and then deducting the vector related with the word "man." Therefore, after turning words into vectors, we use a few techniques to ascertain word similarity.

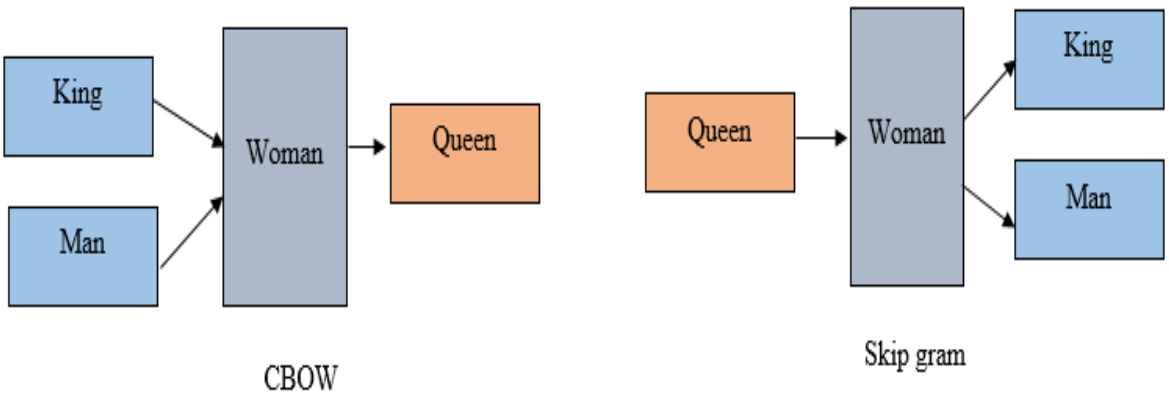

Fig 3: Word Embedding

Word embedding can be done in a number of ways. Among these methods are Word2Vec, BOW, and TF-IDF Scheme. We use Word2Vec as one of these word embedding techniques. Mikolov introduced Word2Vec, a shallow neural network-based model. By training a neural network to predict words that surround a target word CBOW or identify a target word given its immediate context words (skip-gram), word embeddings can be learned. As a result, word vectors that are comparable to other vectors in the area of embedding are able to capture semantic connections between words.

Tokens are converted into numerical representations using the Word2Vec embedding

technique. In particular, the skip-gram model trained on the URL dataset is used to generate 100-dimensional vectors for every token. By capturing the syntactic and semantic links between URL components, these embeddings improve the model's capacity to differentiate between benign and harmful patterns. The following analogy is frequently used to illustrate word embedding semantics: vector('king') − vector('man') + vector('woman') ≈ vector('queen'). Figure 3 illustrates this approch.

3.2 Feature Extraction Using HS-DNN

After the URL data has been tokenized and embedded, a Hilbert Space-based Deep Neural Network (HS-DNN) is used to extract features. Convolutional layers for local pattern extraction, pooling layers for dimensionality reduction, fully linked layers for high-level abstraction, and a Hilbert environment Layer for data projection onto a high-dimensional environment are all components of this design. By improving the separability of features related to harmful and benign URLs, this layered design optimizes the feature set for classification.

After pre-processing, feature extraction is performed using a deep neural network that is based on Hilbert space. In this work, characteristics from the data may be extracted using HS-DNN. A simple CNN consists of the input layer, hidden layer, and output layers, which comprise the Hilbert space layer, pooling layer, and fully connected layers. Figure 4 displays the HS-DNN.

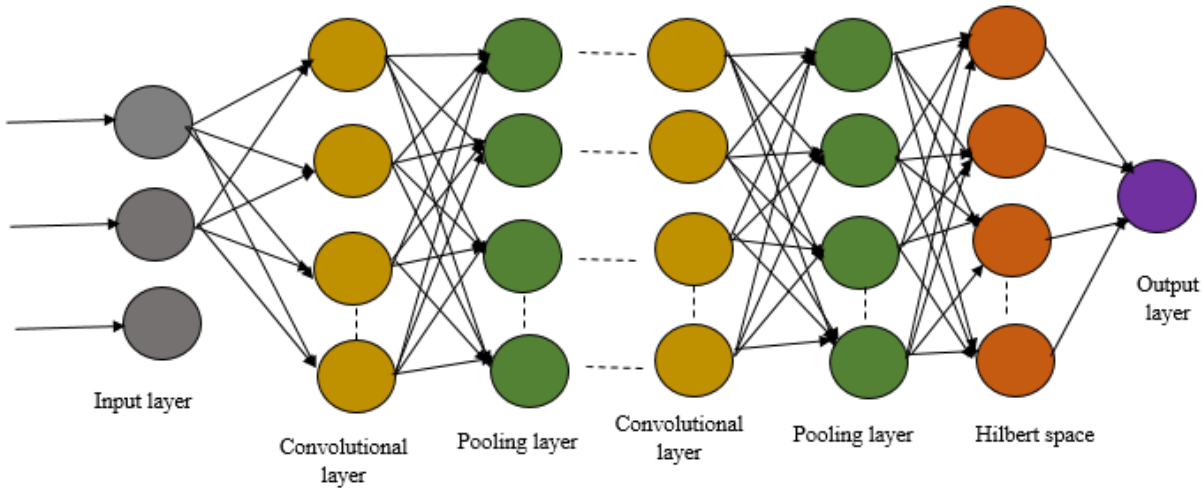

Fig 4: HS-DNN

The final CNN resolution is determined by the biases and weights of the system's previous layers. Consequently, when criteria (1) and (2) are taken into account independently, the model make sense for every layer.

$$\Delta W_q = -\frac{x\lambda}{r} W_c - \frac{x}{N_t}\frac{\partial R}{\partial W_c} + \Delta W_c(t) \quad (3)$$

$$\Delta B_s = -\frac{x}{C}\frac{\partial R}{\partial B_s} + \Delta B_s(t) \quad (4)$$

In this case, x stands for the learning level, the updating step, the rate function, W_n the weight, B_n the bias, n the layer quantity, λ the regularization factor, and N_t the total number of training trials. The following are some of the several types of layers that CNN includes the following:

Input Layer: The input word X is fed into the input layer.

Convolutional Layer: Convolutional layers perform convolution operations on the input data within the context of the Hilbert space. The output feature maps $X^{(c)}$ after convolution are computed as,

$$X^{(c)} = f_{conv}\left(X * W^{(c)} + b^{(c)}\right) \quad (5)$$

Where f_{conv} represents the activation function, $*$ denotes the convolution operation, $W^{(c)}$ are the convolutional filters, $b^{(c)}$ are the bias, and $X^{(c)}$ output of the convolutional layer.

Pooling Layer: Information is gathered from nearby areas of the input feature maps by pooling layers. The pooling feature maps of the output $X^{(p)}$ are calculated as

$$X^{(p)} = f_{pool}\left(X^{(c)}\right) \quad (6)$$

Where f_{pool} represent pooling operation, $X^{(p)}$ output of the pooling layer.

Fully Connected Layer: A fully connected layer comprises all the neurons in the layer of neurons above that are connected to each other. The output's activation $X^{(fc)}$ is computed as

$$X^{(fc)} = f_{fc}\left(X^{(p)} \cdot W^{(fc)} + b^{(fc)}\right) \quad (7)$$

Where $\cdot$ indicates the matrix multiplication operation, $W^{(fc)}$ are the weight and $b^{(fc)}$ are the bias, and f_{fc} the activation function $X^{(fc)}$ is represented by the fully linked layer's output.

Hilbert Space Layer: The Hilbert space layer operates on the output activation from the fully connected layer $X^{(fc)}$ within the Hilbert space. The layer may involve mathematical operations specific to Hilbert spaces, such as inner products, norms, or projections. The output of the Hilbert space layer is $X^{(h)}$.

Output layer: The output layer generates the final forecasts or representations. The prediction for the output $\hat{Y}$ is calculated as

$$\hat{Y} = f_{out}\left(X^{(h)} \cdot W^{(out)} + b^{(out)}\right) \quad (8)$$

Where $W^{(out)}$ stand for the output layer's weight, f_{out} represent the activation function, $b^{(out)}$ represent bias, and $\hat{Y}$ are the output predictions.

3.3. Malicious Detection Using RSM-Based LSTM

After feature extraction, malicious identification using Region-based Support Mapping (LSTM) takes place. Region-based support mapping, or RSM, is a method used in machine learning and artificial intelligence for tasks like object detection. LSTM networks are among the techniques it employs to extract long-range connections from sequential data. The proposed RSM-based LSTM can be applied, for example, to feature extraction sequence analysis from picture areas for object detection applications. First, take features out of the input's regions of interest. The extracted feature from the region i at the time step t as $x_{i,t}$. The extracted region features are given to Support mapping. This support mapping is calculated by the equation (9).

$$y_i = \hat{Y}.f\left(x_{i,t}, \theta\right) \quad (9)$$

Where, y_i is the output or prediction for region i, $x_{i,t}$ is the feature vector extracted from region i, f is the function representing the support mapping, θ represents the parameters of the support mapping function. Because of their gates, which control the flow of information, LSTMs are able to remember or forget information for any amount of time. The input gate, forget gate, and output gate, all three of which are governed by a set of weights, are typically the three main components of an LSTM cell. Figure 5 shows the LSTM network based on RSM.

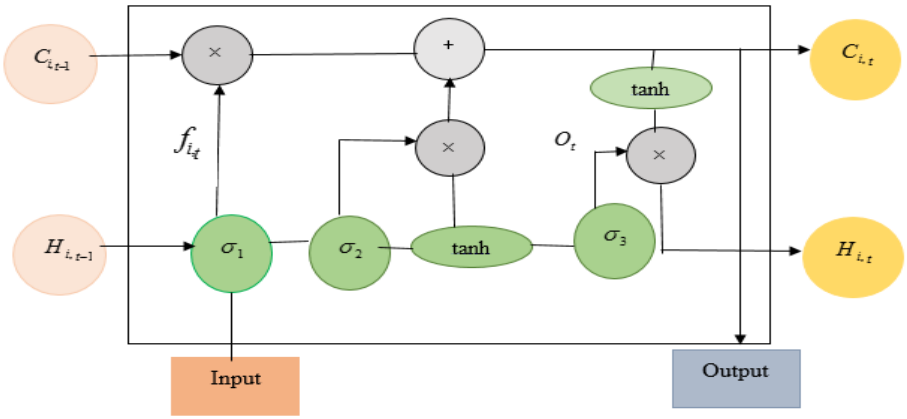

Fig: 5 RSM-based LSTM

The forget gate is defined by,

$$f_{i,t} = \sigma_1\left(W_{if}.\left[h_{i,t-1}, x_{i,t}\right] + b_f\right) \quad (10)$$

Where $h_{i,t-1}$ in the state $t-1$ is the output located, containing and expressing the forget gate's bias b_f and W_f weight matrices. Prior to being stored in the cell state, new information is processed at the second step. In this stage, a hyperbolic tangent (tanh) layer simultaneously generates a vector of

candidate values $\widetilde{C}_{i,t}$ and the sigmoid layer σ_2 determines the updated values in the cell state, represented as. Next, the new data and the previous cell state are changed into the new cell state C_t by

$$C_{i,t} = f_{i,t} * C_{i,t-1} + i_t * \widetilde{C}_{i,t} \quad (11)$$

i_t Which d $\widetilde{C}_{i,t}$ are defined by,

$$i_t = \sigma_2\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \quad (12)$$

$$\widetilde{C}_t = \tanh\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \quad (13)$$

Where $\left(W_i, b_i\right)$and $\left(W_c, b_c\right)$are weight matrices and the biases of input gate and memory cell state, respectively. From this updated state, the last step is to output values of an LSTM cell. The cell state C_t is put into a tanh layer to scale down the vector state to number between -1 and 1 while the output gate O_t determines part of the cell state being outputted through the number within [0, 1] computing from the sigmoid layer σ_3. The value h_t is then regulated and filtered before outputting by multiplying these two numbers as

$$h_{i,t} = O_{i,t} * \tanh\left(C_{i,t}\right) \quad (14)$$

In which $O_{i,t}$ is defined by

$$O_{i,t} = \sigma_3\left(W_o.\left[h_{i,t-1}, x_{it}\right] + b_o\right) \quad (15)$$

Where W_o and b_o are the weight matrix and the biases of output gate. Finally, Region-based Support Mapping - LSTM significantly classifies the URL whether it is normal or malicious.

4. Result and Discussion

In this section, it presents the analysis of results and compares them with those from other methodologies. We utilized a URL dataset to evaluate the research outcomes, benchmarking our proposed approach against techniques such as SVM, LSTM [21], DNN, and CNN. The experiments were conducted using Intel Core™ i7 processors running at 1.6 GHz on the Python platform. This comparison highlights the effectiveness of our method in relation to established models.

4.1 Dataset Description

URL Dataset:

The dataset that are use in this study is formed by joining multiple publicly available sources to make a collection of benign, phishing and malware URLs which are comprehensive and balanced URLs. Aim of this study is to assure in URL structure and content, while searching class imbalance, which is general issue in cybersecurity datasets.

The following data sources were used to construct the dataset:

•ISCX-URL2016: The dataset which is from Canadian Institute for Cybersecurity used in great extent which is made up of labeled benign and malicious URLs. It represent as the foundation of dataset and supplies trustworthy baseline for the research classification of URL .

•Malware Domain Blacklist: This source was used to increase the depiction of malware URLs. This is an open-source collection of noted malicious and malware-hosting domains.

•PhishTank and PhishStorm:This is community-led and publicly available phishing URL databases. These were incorporated to broaden the phishing URLs, verifying a broad and renewed set of phishing attack vectors

•Faizan GitHub Repository:Added to the benign class to avoid imbalace in counter class created due to phsishing and malware URL.

For final dataset compilation all sources's URLs were collected and blend into a unique structured DataFrame maintaining only the URL and its respective label (benign or malicious). Duplicate entries were removed, and labels were standardized. This multi-source integration gives a rich dataset that promotes the robust training and evaluation of the proposed deep learning model. Table 1 presents the composition of the dataset.

Table: 1 Dataset Composition

Source	Type	Number of URLs	Percentage (%)
ISCX-URL2016	Benign & Malicious	140,000	35%
Malware Domain Blacklist	Malicious	60,000	15%
PhishTank	Phishing	80,000	20%
PhishStorm	Phishing	40,000	10%
Faizan GitHub Repository	Benign	80,000	20%
Total	**–**	**400,000**	**100%**

4.2 Performance Metrics

In terms of summary, the numerical metrics of Accuracy, Precision, Recall, F-Score, AUC, ROC, TPR, FPR, Computation Time, and Confusion Matrix can be expressed. We utilize the statistical indicators provided in this part to examine the efficacy of our suggested work. Performance metrics are computed for True Positive (TP), True Negative (TN), False Positive (FN), and False Negative (TP) (FN). Table 2 displays the Performance Measures Formula.

Table: 2 Performance measures formula
TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

Metric	Formula	Description
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Proportion of total correct predictions
Precision	TP / (TP + FP)	Proportion of predicted malicious URLs that are correct
Recall	TP / (TP + FN)	Proportion of actual malicious URLs correctly detected
F1-Score	2 × (Precision × Recall) / (Precision + Recall)	Harmonic mean of precision and recall
Specificity	TN / (TN + FP)	Proportion of benign URLs correctly identified
AUC-ROC	Area under the ROC curve	Model's ability to distinguish between classes

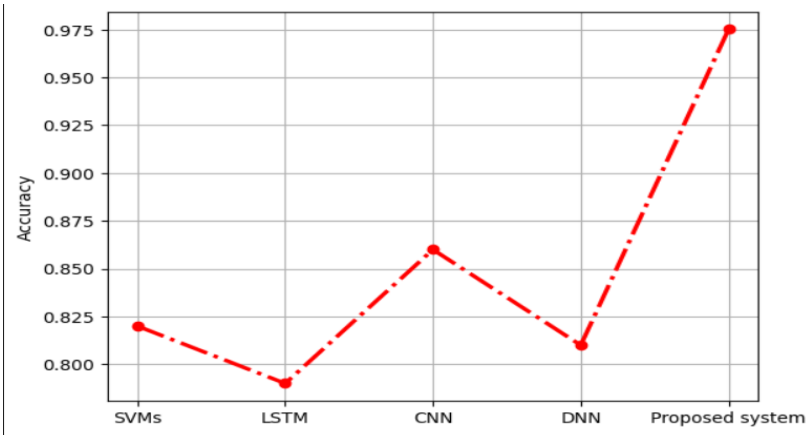

Fig: 6 Performance and Comparative Analysis of Accuracy

Fig. 6 presents a comparison between the accuracy of the proposed system and current approaches for the ISCX-URL2016 dataset. With a precision of 98.2%, the suggested system outperforms the state-of-the-art techniques like SVM (82%), LSTM (79%), CNN (86%), and DNN (81%).

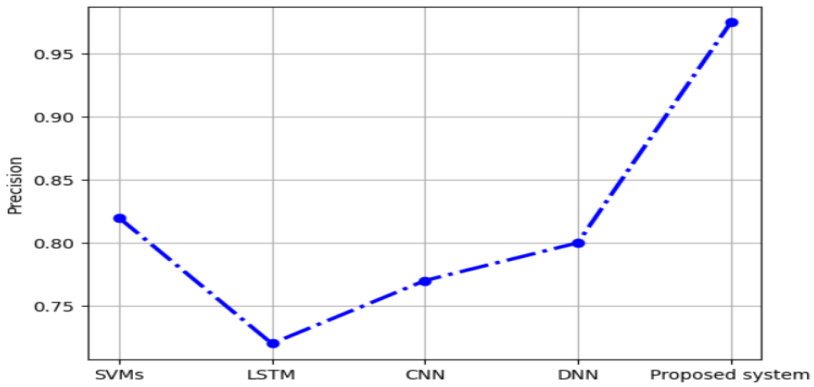

Fig: 7 Performance and Comparative Analysis of Precision

Fig 7 depicts a comparison of precision between the proposed system and existing methods on the ISCX-URL2016 dataset. Notably, the proposed system exhibits superior precision, achieving a perfect score of 98.2% outperforming established methods like SVM (82%), LSTM (72%), CNN (77%), and DNN (80%).

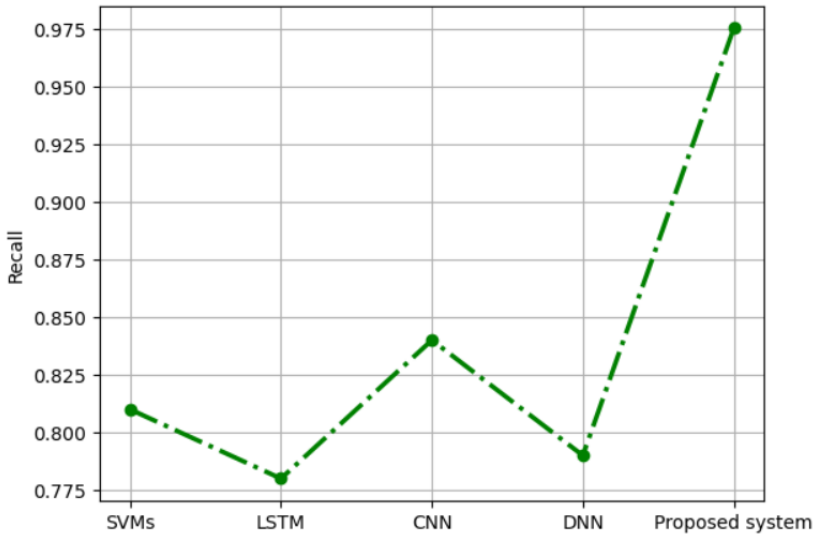

Fig: 8 Performance and Comparative Analysis of Recall

Fig 8 compares the recall of the proposed system for the ISCX-URL2016 dataset with existing methods. The proposed system demonstrates higher recall, reaching 98.2%, compared to existing methods such as SVM (81%), LSTM (78%), CNN (84%), and DNN (79%).

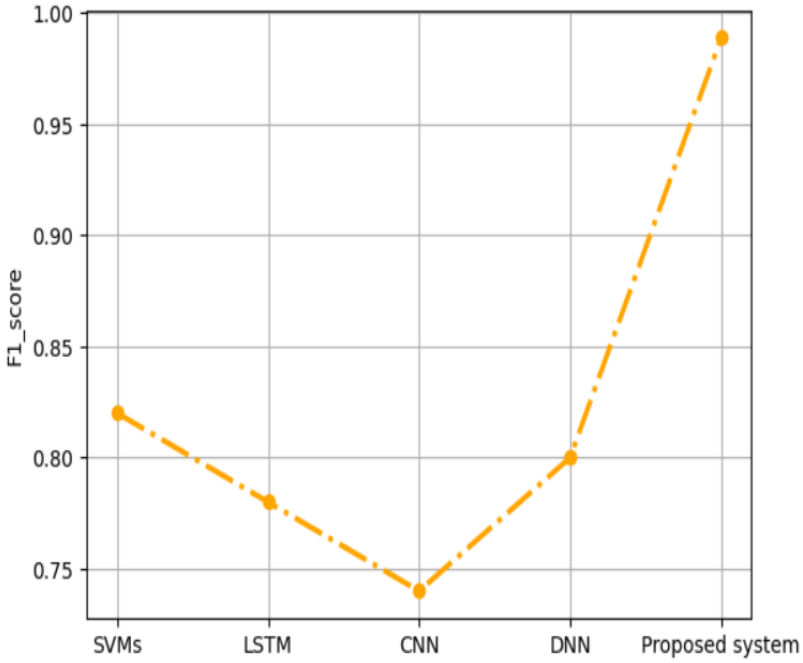

Fig: 9 Performance and comparative analysis of F1-score

Fig 9 presents a comparison of the F1-score achieved by the proposed system on the ISCX-URL2016 dataset against existing methods. Notably, the proposed system exhibits superior performance with an F1 score of 98.9%, surpassing conventional methods like SVM (82%), LSTM (78%), CNN (74%), and DNN (80%).

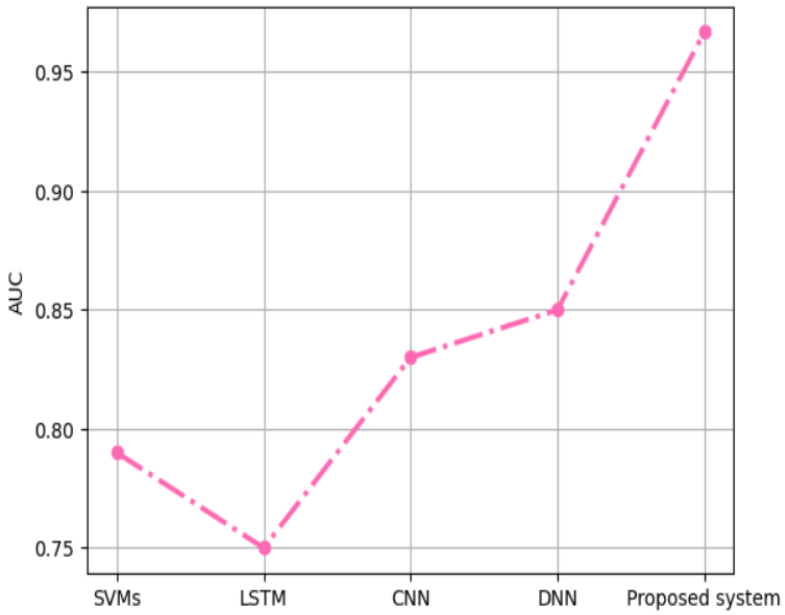

Fig 10: Performance and Comparative Analysis of Area Under the Curve

Fig 10 illustrates a comparison of the Area Under the Curve (AUC) achieved by the proposed system on the ISCX-URL2016 dataset against established methods. Notably, the proposed system showcases superior performance with an AUC of 97%, outperforming conventional methods such as SVM (79%), LSTM (75%), CNN (83%), and DNN (85%).

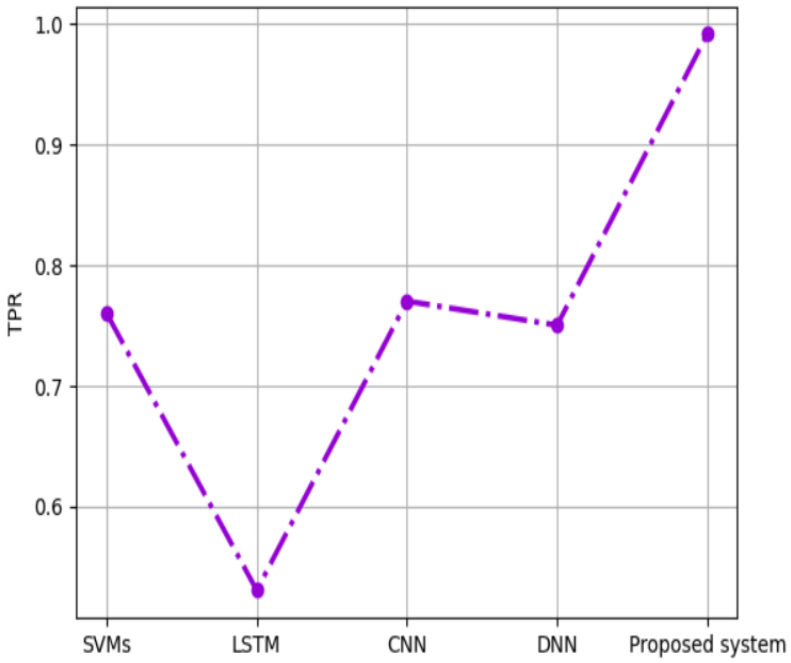

Fig 11: True Positive Rate

Fig 11 displays the rate of True Positive Rate (TPR) attained by the proposed system utilizing the ISCX-URL2016 dataset and the current techniques. Comparisons are made between the proposed and existing methods including SVM, LSTM, DNN, and CNN. The proposed system has demonstrated superior performance with a high TPR of 100%, surpassing existing methods like SVM (0.24%), LSTM (0.47%), CNN (0.23%), and DNN (0.25%).

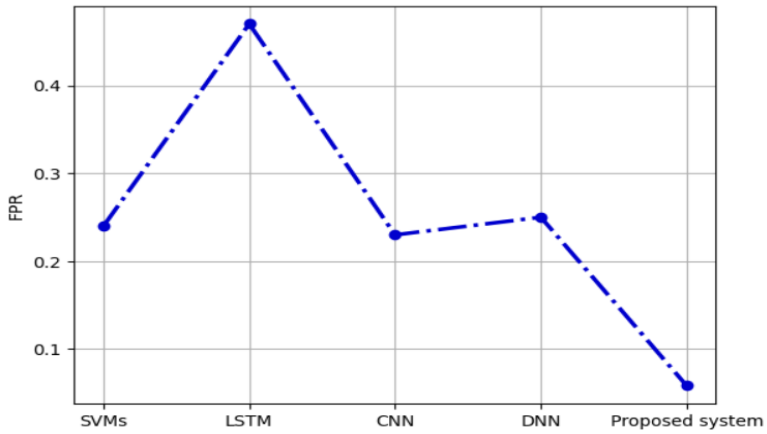

Fig: 12 False Positive Rate

Fig 12 displays the rate of True Positive Rate (TPR) attained by the proposed system utilizing the ISCX-URL2016 dataset and the current techniques. Comparisons are made between the proposed and existing methods including SVM, LSTM, DNN, and CNN. The proposed system has demonstrated superior performance with a low FPR of 0.058%, surpassing existing methods like SVM (76%), LSTM (53%), CNN (77%), and DNN (75%).

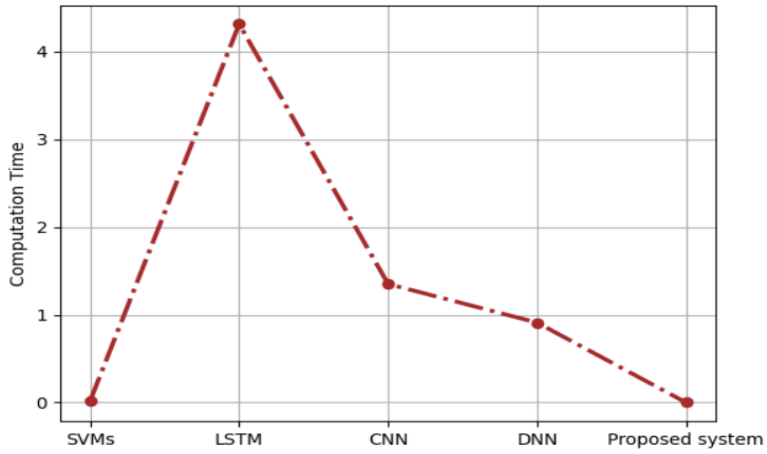

Fig13: Computation Time

A comparison study and an illustration of computation time performance are shown in Fig. 13. We compare our approach with other state-of-the-art approaches including SVM, LSTM, CNN, and DNN. Examining Fig. 12, it is clear that of the contrasted approaches, the suggested way achieves the least computing time.

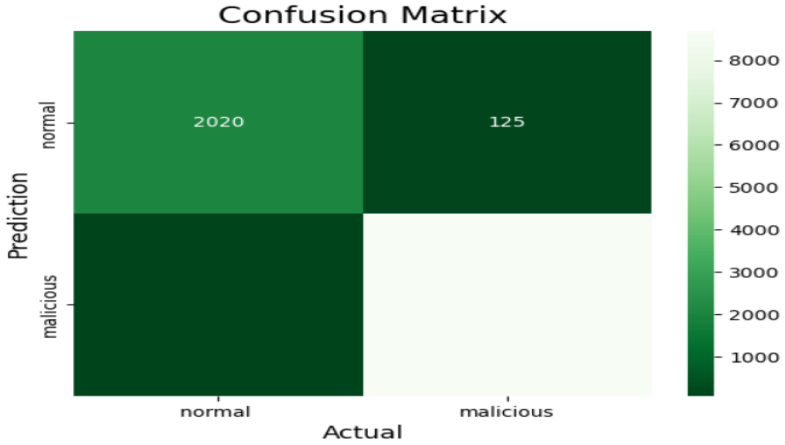

Fig14: Confusion Matrix

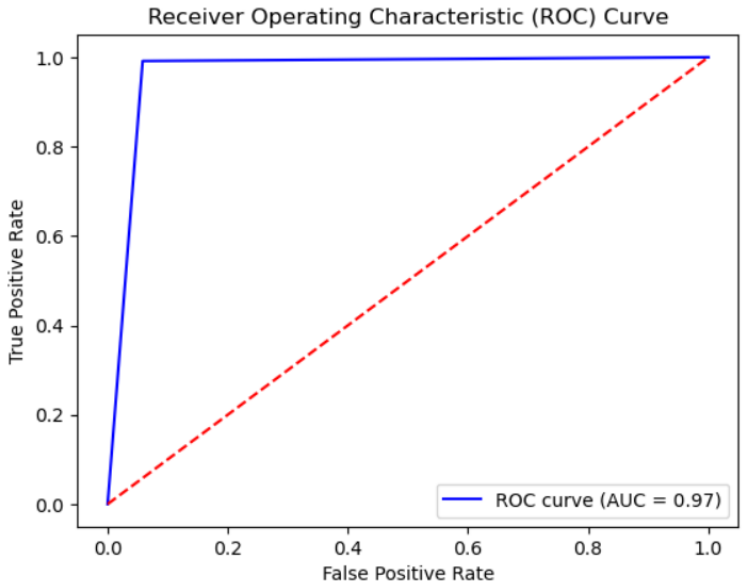

Fig 15:

Fig 15: Performance and Comparative Analysis of ROC Curve

The confusion matrix, actual labels, and predicted labels are displayed in Fig 14. The ROC curve is used to plot the True Positive Rate against the False Positive Rate. Fig. 15 displays the ROC curve of our proposed model for the ISCX-URL2016 dataset, which has an AUC of 0.97. Table 2 offers the comparison of performance metrics with the value representation of the current and proposed methods.

Table 3: Performance Metrics of Proposed Model

Metric	Proposed (RSM-LSTM and HS-DNN)	SVM	CNN	LSTM
Accuracy (%)	98.2	92.4	95.1	96.3
Precision (%)	97.6	91.2	93.8	95.0
Recall (%)	98.1	89.7	94.5	95.8
F1-Score (%)	97.8	90.4	94.1	95.4
AUC (%)	98.7	93.0	95.6	96.9
TPR (%)	97.4	90.1	93.2	94.5
FPR (%)	2.3	7.2	5.4	4.7

Table 4: Comparison of Performance of the Existing Method

Author & Year	Dataset Used @ Classifier	Accuracy	Precision	Recall	F1-score	TPR	FPR	ROC	Limitations
Afzal et.al [26] 2021	**Classifier** k-mean clustering **Dataset** Custom (URL Corpus)	99.7%	99.1	98.4%	96.8%	-	-	-	Doesn't capture sequential dependencies
Aljabri et.al [27] 2022	**Classifier** Naive **Dataset** Phish Tank,	96%	96%	98%	97%	-	-	-	Inadequate functionality on complicated or obscured URLs

	URL Repositories								
Mumu et.al [28] 2023	**Classifier** Naïve baizes **Dataset** PhishTank, Custom	99.6%	99.6%	98.6%	-	-	-	99.1%	Fails in hostile environments
Heryanto et.al [29] 2020	**Classifier** SVM **Dataset** Custom Dataset	99%	100%	98%	99%	98%	0%		Insufficient scalability and absence of contextual learning
Raja et.al [30] 2021	**Classifier** KNN **Dataset** Public URL dataset	98%	98%	98%	98%	-	-	-	Excessive memory use; inappropriate for real-time applications
Mythreya et.al [31] 2022	**Classifier** SVM **Dataset** UCI Dataset	99.6%	99.7%	98%	97%	-	-	-	restricted to feature sets that are static
Proposed RSM-LSTM with HS-DNN	**Classifier** RSM based LSTM **Dataset** ISCX-URL2016 + PhishTank	98.2%	98.2%	98.2%	99%	99%	0.058%	97%	Needs further real-time deployment validation

5. Conclusion

This study presents a different and smart approach that combines advanced AI models (RSM-LSTM and HS-DNN) to detect harmful or dangerous website links (URLs). This new approach reached a very high accuracy of 98.2%, which is better than other commonly used models like SVM, LSTM, CNN, and basic Onsite result proves that, this work is important because it can be used in real-life cybersecurity systems. The model understands both the words and structure of URLs, making it great for finding threats early in today's digital world. It can be used in many ways, like spotting fake links in emails, checking links quickly in web browsers, or blocking bad websites through network firewalls. But using this model in real security systems comes with some difficulties. Challenges include making it strong against trick attacks, adjusting to new and unknown threats like brand-new phishing links, and keeping it fast for real-time use. Future studies can look into using transformer models and learning methods that keep updating to make the system more flexible and useful for a long time. By solving these real-world problems and making it easier to use in systems, this method is a big step forward in building smart, strong, and flexible cybersecurity tools.

6. Future Scope

The promising results of this research open several avenues for future work and real-world integration. Owing to its high detection accuracy and capability to learn deep semantic and structural patterns from URLs, the proposed hybrid model holds strong potential for deployment within practical cybersecurity environments.

A key direction for future exploration involves integrating the model into real-time security infrastructures, such as:

- Firewall-based web filters, enabling proactive blocking of malicious URLs at the network level
- Browser extensions, providing real-time phishing warnings during user navigation
- Email security gateways, for detecting and filtering malicious or phishing links in incoming communications
- Cloud-based detection services, delivered through REST APIs to support scalable enterprise-level applications

Data Availability and material

The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Acknowledgments:

The authors extend their appreciation to Taif University, Saudi Arabia, for supporting this work through project number (TU-DSPP-2024-56)

References:

1. Kumar, M. Ajay, Santhi Chebiyyam, and G. Archana Devi. "ENCOUNTERING AND NOTICING OF MALICIOUS URLS USING DEEP LEARNING METHODS." *intelligence* 52, no. 4 (2023).
2. Ul Hassan, Ietezaz, Raja Hashim Ali, Zai n Ul Abideen, Talha Ali Khan, and Rand Kouatly. "Significance of machine learning for detection of malicious websites on an unbalanced dataset." *Digital* 2, no. 4 (2022): 501-519.
3. Velpula, Koteswara Rao, Kataru Gayathri Priya, Kushwanth Kumar Jammula, Krishna Sruthi Velaga, and Praveen Kumar Kongara. "Malicious URL Detection Using Machine Learning."
4. Almousa, May, and Mohd Anwar. "A URL-Based Social Semantic Attacks Detection With Character-Aware Language Model." *IEEE Access* 11 (2023): 10654-10663.
5. Srinivasan, Sriram, R. Vinayakumar, Ajay Arunachalam, Mamoun Alazab, and K. P. Soman. "DURLD: Malicious URL detection using deep learning-based character level representations." *Malware analysis using artificial intelligence and deep learning* (2021): 535-554.
6. Kumar, Vivek, Diego Reforgiato Recupero, Daniele Riboni, and Rim Helaoui. "Ensembling classical machine learning and deep learning approaches for morbidity identification from clinical notes." *IEEE Access* 9 (2020): 7107-7126.
7. Raja, A.S.; Madhubala, R.; Rajesh, N.; Shaheetha, L.; Arulkumar, N. Survey on Malicious URL Detection Techniques. In Proceedings of the 6th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 28–30 April 2022; pp. 778–781.
8. Karim, Abdul, Mobeen Shahroz, Khabib Mustofa, Samir Brahim Belhaouari, and S. Ramana Kumar Joga. "Phishing Detection System Through Hybrid Machine Learning Based on URL." *IEEE Access* 11 (2023): 36805-36822.
9. Raja, A.S.; Pradeepa, G.; Arulkumar, N. Mudhr: Malicious URL detection using heuristic rules-based approach. In AIP Conference Proceedings; AIP Publishing LLC:Woodbury, NY, USA, 2022; Volume 2393.
10. Raja, A.S.; Vinodini, R.; Kavitha, A. Lexical features based malicious URL detection using machine learning techniques. Mater.Today Proc. 2021, 47, 163–166.
11. Raja, A.S.; Sundarvadivazhagan, B.; Vijayarangan, R.; Veeramani, S. MaliciousWebpage Classification Based onWeb ContentFeatures Using Machine Learning and Deep Learning. In Proceedings of the International Conference on Green Energy, Computing and Sustainable Technology (GECOST) 2022, Virtual, 26–28 October 2022.
12. Awasthi, A.; Goel, N. Phishing website prediction using base and ensemble classifier techniques with cross-validation. Cybersecurity2022, 5, 22.
13. Abu Al-Haija, Qasem, and Mustafa Al-Fayoumi. "An intelligent identification and classification system for malicious uniform resource locators (URLs)." *Neural Computing and Applications* (2023): 1-17.
14. Lamrabti, Omar, Abdellatif MezriOui, and Abdelhamid Belmekki. "URL_trigger: Real time solution for Detection Malicious URL using Deep Learning." In *E-Learning and Smart Engineering Systems (ELSES 2023)*, pp. 328-334. Atlantis Press, 2024.
15. Tsinganos, Nikolaos, Panagiotis Fouliras, Ioannis Mavridis, and Dimitrios Gritzalis. "CSE-ARS: Deep learning-based late fusion of multimodal information for chat-based social engineering attack recognition." *IEEE Access* (2024).
16. Hilal, Anwer Mustafa, Aisha Hassan Abdalla Hashim, Heba G. Mohamed, Mohamed K. Nour, Mashael M. Asiri, Ali M. Al-Sharafi, Mahmoud Othman, and Abdelwahed Motwakel. "Malicious URL Classification Using Artificial Fish Swarm Optimization and Deep Learning." *Computers, Materials & Continua* 74, no. 1 (2023).
17. Nikam, Umesh, and Vaishali M. Deshmukh. "Hybrid Feature Selection Technique to classify Malicious Applications using Machine Learning approach." *Journal of Integrated Science and Technology* 12, no. 1 (2024): 702-702.

18. Shaukat, Muhammad Waqas, Rashid Amin, Muhana Magboul Ali Muslam, Asma Hassan Alshehri, and Jiang Xie. "A hybrid approach for alluring ads phishing attack detection using machine learning." *Sensors* 23, no. 19 (2023): 8070.
19. Nagy, Naya, Malak Aljabri, Afrah Shaahid, Amnah Albin Ahmed, Fatima Alnasser, Linda Almakramy, Manar Alhadab, and Shahad Alfaddagh. "Phishing URLs Detection Using Sequential and Parallel ML Techniques: Comparative Analysis." *Sensors* 23, no. 7 (2023): 3467.
20. Kumar, Anil, Amandeep Noliya, Ritu Makani, Pardeep Kumar, and Jagsir Singh. "Advanced Persistent Threat Detection Performance Analysis Based on Machine Learning Models." *International Journal of Intelligent Systems and Applications in Engineering* 12, no. 2 (2024): 741-757.
21. Qiqieh, Issa, Omar Alzubi, Jafar Alzubi, K.C. Sreedhar, and Ala M. Al-Zoubi. "An intelligent cyber threat detection: A swarm-optimized machine learning approach." *Alexandria Engineering Journal* 83 (2024): Article ID 107878. https://doi.org/10.1016/j.aej.2024.12.039.
22. Afzal, Sara, Muhammad Asim, Abdul Rehman Javed, Mirza Omer Beg, and Thar Baker. "Urldeepdetect: A deep learning approach for detecting malicious URLs using semantic vector models." Journal of Network and Systems Management 29 (2021): 1–27.
23. Alzubi, Omar A., Jafar A. Alzubi, Mohammed Alweshah, Issa Qiqieh, Sara Al-Shami, and Manikandan Ramachandran. "An optimal pruning algorithm of classifier ensembles: dynamic programming approach." Neural Computing and Applications 32, no. 20 (2020): 16005–16028. https://doi.org/10.1007/s00521-020-04761-6.
24. Zieni, Rasha, Luisa Massari, and Maria Carla Calzarossa. "Phishing or not phishing? A survey on the detection of phishing websites." IEEE Access 11 (2023): 18499-18519
25. Singh, Sanskriti Sanjay Kumar, Vishnu Menon, S. A. Sajidha, V. M. Nisha, M. Nivedita, and Aakif Mairaj. "Meta Learning for Enhanced Web Security Against Malicious URLs." (2023).
26. Afzal, Sara, Muhammad Asim, Abdul Rehman Javed, Mirza Omer Beg, and Thar Baker. "Urldeepdetect: A deep learning approach for detecting malicious urls using semantic vector models." Journal of Network and Systems Management 29 (2021): 1-27.
27. Mumu, Mahmuda Haque, and Tanzina Aishy. "Malicious URL detection using machine learning and deep learning algorithms." PhD diss., East West University, 2023.
28. Aljabri, Malak, Fahd Alhaidari, Rami Mustafa A. Mohammad, Samiha Mirza, Dina H. Alhamed, Hanan S. Altamimi, and Sara Mhd Chrouf. "An assessment of lexical, network, and content-based features for detecting malicious urls using machine learning and deep learning models." Computational Intelligence and Neuroscience 2022 (2022).
29. Heryanto, Ahmad, Mohd Faizal Ab Razak, Anis Farihan Mat Raffei, Danakorn Nincarean Eh Phon, Shahreen Kasim, and Tole Sutikno. "A malicious URLs detection system using optimization and machine learning classifiers." Indonesian Journal of Electrical Engineering and Computer Science 17, no. 3 (2020): 1210-1214.
30. Raja, A. Saleem, R. Vinodini, and A. Kavitha. "Lexical features based malicious URL detection using machine learning techniques." Materials Today: Proceedings 47 (2021): 163-166.
31. Mythreya, S., A. Sampath Dakshina Murthy, K. Saikumar, and V. Rajesh. "Prediction and prevention of malicious URL using ML and LR techniques for network security: machine learning." In Handbook of Research on Technologies and Systems for E-Collaboration During Global Crises, pp. 302-315. IGI Global, 2022.