Taylor & Francis
Taylor & Francis Group

Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/taut20

Real-time phishing URL detection framework using knowledge distilled ELECTRA

K. S. Jishnu & B. Arthi

To cite this article: K. S. Jishnu & B. Arthi (2024) Real-time phishing URL detection framework using knowledge distilled ELECTRA, Automatika, 65:4, 1621-1639, DOI: 10.1080/00051144.2024.2415797

To link to this article: https://doi.org/10.1080/00051144.2024.2415797

AUTOMATIKA
2024, VOL. 65, NO. 4, 1621–1639
https://doi.org/10.1080/00051144.2024.2415797

Real-time phishing URL detection framework using knowledge distilled ELECTRA

K. S. Jishnu and B. Arthi

Department of Computing Technologies, School of Computing, College of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur Campus, Chengalpattu, Tamilnadu, India

ABSTRACT

The rise of cyber threats, particularly URL-based phishing attacks, has tarnished the digital age despite its unparalleled access to information. These attacks often deceive users into disclosing confidential information by redirecting them to fraudulent websites. Existing browser-based methods, predominantly relying on blacklist approaches, have failed to effectively detect phishing attacks. To counteract this issue, we propose a novel system that integrates a deep learning model with a user-centric Chrome browser extension to detect and alert users about potential phishing URLs instantly. Our approach introduces a Knowledge Distilled ELECTRA model for URL detection and achieves remarkable performance metrics of 99.74% accuracy and a 99.43% F1-score on a diverse dataset of 450,176 URLs. Coupled with the browser extension, our system provides real-time feedback, empowering users to make informed decisions about the websites they visit. Additionally, we incorporate a user feedback loop for continuous model enhancement. This work sets a precedent by offering a seamless, robust, and efficient solution to mitigate phishing threats for internet users.

ARTICLE HISTORY
Received 3 May 2024
Accepted 19 September 2024

KEYWORDS
Distilled ELECTRA; phishing detection; URL classification; chrome extensions; real-time security

1. Introduction

In today's digitally connected world, where the internet underpins essential activities from social interactions to financial transactions, cyber security threats have become more pervasive than ever. Among these threats, phishing attacks stand out as particularly insidious, exploiting both technological vulnerabilities and human psychology to deceive unsuspecting users. Despite concerted efforts to combat these threats, the landscape of cyber crime continues to evolve rapidly, with phishing attacks reaching unprecedented levels of sophistication and prevalence. The year 2023 marked a grim milestone in the battle against phishing, as evidenced by the alarming statistics revealed in the Anti-Phishing Working Group (APWG) report, which documented nearly five million reported phishing attacks over the year, surpassing the previous record set in 2022. The APWG report for 2023 reveals the monthly count of unique phishing websites, as illustrated in Figure 1. Notably concerning is the peak value of 619,060 recorded in March, while even the lowest count, observed in June, remained above 300,000, specifically at 306,847. It is evident that traditional approaches to prevention and detection are falling short in the face of this escalating threat.

Despite ongoing advancements in cybersecurity, existing phishing mitigation strategies face critical limitations that leave users vulnerable to increasingly sophisticated attacks. User awareness campaigns, while essential, often fall short due to the constantly evolving tactics of cybercriminals. Similarly, traditional software-based detection systems, reliant on static blacklists and heuristic algorithms, struggle to keep pace with the dynamic and adaptive nature of phishing attacks. These gaps underscore the urgent need for more sophisticated, real-time detection solutions that can adapt to emerging threats. This is where the intersection of machine learning and cyber security holds tremendous promise. By harnessing the power of advanced machine learning techniques, such as deep learning, it becomes possible to develop dynamic and adaptive systems capable of identifying phishing URLs with unprecedented accuracy and speed [1].

Addressing these critical gaps, our research introduces a novel phishing URL detection system that leverages advanced deep learning methodologies integrated with user-friendly browser extensions. This approach not only enhances the accuracy of phishing detection but also adapts in real-time to new and evolving threats. By empowering users with an intuitive, proactive tool, our solution has the potential to significantly reduce the incidence of phishing attacks, thereby contributing to a safer online environment for individuals and organizations alike. The practical implications of our proposed model are far-reaching. In practice, this system can be seamlessly integrated into existing web browsers,

CONTACT B. Arthi arthib@srmist.edu.in Department of Computing Technologies, School of Computing, College of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur Campus, Chengalpattu, Tamilnadu 603203, India
This article has been corrected with minor changes. These changes do not impact the academic content of the article.

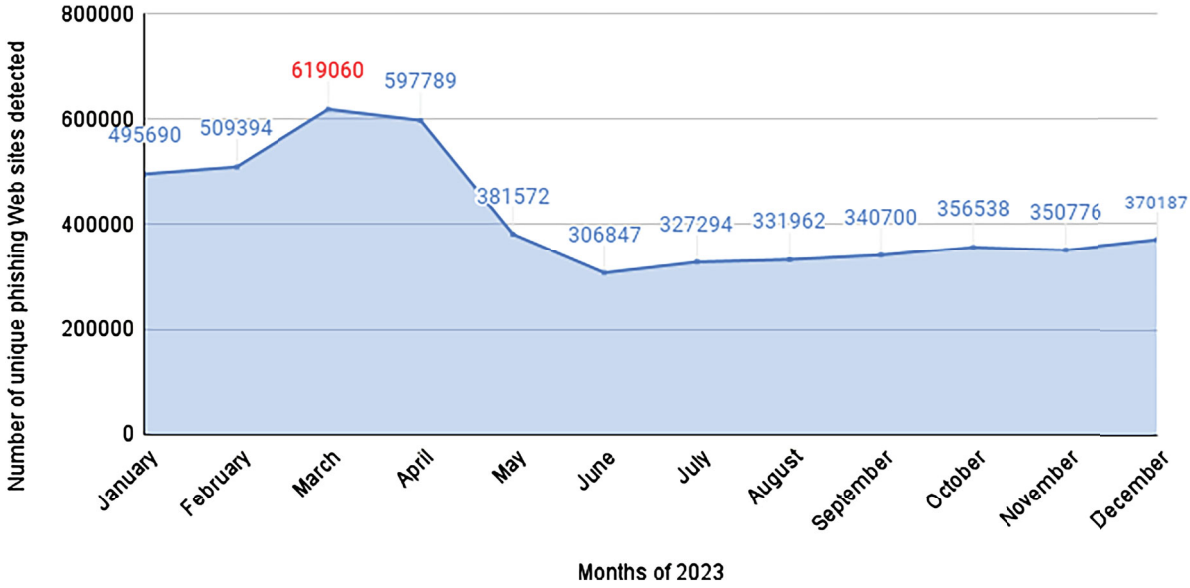

Figure 1. Number of unique phishing web sites detected in 2023.

offering real-time protection against phishing attacks to millions of users worldwide. Moreover, its adaptability ensures that it can be effectively deployed across critical sectors such as finance, healthcare, and e-commerce, where the stakes of phishing attacks are particularly high. The system's ability to learn from new phishing attempts in real-time ensures that it remains effective even as cyber threats continue to evolve.

At the heart of our method lies the Knowledge Distilled ELECTRA model, meticulously crafted and optimized for the task of URL classification. By leveraging the latest advancements in machine learning, we aim to revolutionize real-time phishing URL detection, empowering users to navigate the complexities of the online world with confidence and security. The contributions of this work are significant, revolutionizing the landscape of cybersecurity by:

- Introducing a pioneering method that merges deep learning with browser extensions to detect phishing URLs in real-time, marking a fundamental shift in prevention techniques.
- Leveraging the Knowledge Distilled ELECTRA model, meticulously crafted for URL classification, significantly enhances detection accuracy.
- Providing user-friendly browser extensions, our solution empowers individuals to make informed decisions and reduce the risk of falling prey to phishing attacks.
- Offering a comprehensive examination of our approach, including methodologies, empirical evaluations, and implications for cyber security research and practice, this paper advances knowledge in the field.

The paper unfolds as a comprehensive exploration of our groundbreaking approach, structured to provide readers with a deep understanding of the challenges, methodologies, results, and implications of our research. Section 2 delves into current phishing detection methodologies, highlighting their limitations and the necessity for novel solutions. Section 3 unveils the architecture of the Knowledge Distilled ELECTRA model and its integration with browser extensions. Section 4 presents empirical evaluations, showcasing system performance across diverse phishing URLs and examines the implications of the findings for cyber security research and practice. Finally, Section 5 concludes with reflections on the significance of our contributions and future directions for real-time phishing URL detection.

2. Related works

In phishing URL detection employing machine learning, a rich body of literature encompasses traditional and contemporary methodologies, addressing the evolving landscape of cyber threats [2]. This section overviews fundamental studies, categorizing them into Machine Learning-Based Techniques, Deep Learning-Based Techniques, and Web Browser Extension Approaches.

2.1. *Machine learning-based techniques*

A diverse array of studies has contributed to the dynamic landscape of phishing detection using machine learning techniques. Odeh et al. [3] explored machine learning techniques for detecting phishing domains by applying CatBoost, XGBoost, and LightGBM models. Their analysis, based on the UCI phishing domains dataset, revealed that the CatBoost model outperformed the others and exceeded the performance of earlier methods, utilizing 18 extracted features.

Qasem et al. [4] implemented Shallow Neural Networks (SNNs) and Decision Trees (DT) in their phishing URL detection system. They used two datasets consisting of 58,645 and 88,647 URLs, containing a total of 111 features. In another work, Qasem et al. [5] introduced a two-layer machine learning system to detect malicious URLs, first classifying them as benign or malicious, and then further categorizing them into five classes. The system, tested on the ISCX-URL2016 dataset, employed four ensemble methods: bagging trees, k-nearest neighbour, boosted decision trees, and subspace discriminator. Sheikhi et al. [6] proposed a model that extracts features and selects relevant ones using a firefly algorithm. This model then detects malicious websites by employing XG Boost, which has been optimized using Particle Swarm Optimization.

Hannousse and Yahiouche [7] adopted a holistic approach, integrating machine learning, visual similarity, and heuristics, achieving optimal accuracy. Rashid et al. [8] explored phishing detection using Support Vector Machine, achieving commendable accuracy. Basit et al. [9] employed various machine learning algorithms, detecting phishing attacks with 97.33% accuracy. Stobbs et al. [10] combined machine learning, heuristics, and list-based approaches, demonstrating optimum results with Random Forest. Abedin et al. [11], Alam et al. [12], Sindhu et al. [13], Kasim [14], and others [15–19] each brought distinctive approaches, achieving high accuracies but facing challenges such as limited datasets and algorithmic dependencies.

2.2. Deep learning-based techniques

In the pursuit of effective phishing website detection, recent research has explored innovative approaches, particularly leveraging the capabilities of deep learning techniques. Several studies have demonstrated promising results in this domain. Wang et al. [20] created a malicious URL detection model with a dynamic convolutional neural network (DCNN) that incorporates a new folding layer and k-max-pooling instead of traditional pooling. The model adjusts feature mapping and pooling parameters according to URL length and convolution depth to improve feature extraction. Manika and Shivani [21] developed the BiLSTM-GHA-CNN for phishing URL detection. The BiLSTM captures context, the CNN extracts key features, and the highway network enhances feature extraction and convergence speed. A gating mechanism balances the outputs from both the CNN and BiLSTM. Saad et al. [22] introduced a phishing detection model called PDGAN that relied solely on a website's URL. The model used a GAN with an LSTM as the generator and a CNN as the discriminator. Zaimi et al. [23] suggested a method that uses permutation importance to pick out important features and the SMOTE-Tomek link method to deal with imbalanced datasets. The system also uses the XGBoost classifier and four deep learning models to categorise URLs. Alshingiti et al. [24] proposed a method that utilizes LSTM, CNN, and a hybrid model (LSTM–CNN). They applied a SelectKBest feature selection model to select 30 features from an initial set of 80. Sahingoz et al. [25] implemented five deep learning-based methods and presented their results to showcase the impact of deep learning on phishing URL classification. All the models outperformed traditional machine learning models. Barath et al. [26] proposed a CNN-based phishing URL detection system called BaitNet, which achieved high accuracy when trained on a dataset of 100,000+ URLs. However, implementing this system as a real-time website for detecting phishing URLs is not feasible because it would require users to check each website through a separate platform, which is impractical and inefficient.

Bu and Cho [27] employed a recurrent neural network (RNN) in a heuristic framework, achieving enhanced sensitivity but acknowledging a focus on character-level features, suggesting potential improvements with a broader consideration of web address structure. Korkmaz et al. [28] utilized a convolutional neural network (CNN) for detecting phishing URLs, reporting an 88.90% accuracy rate. Feng and Yue [29] showcased the effectiveness of RNN models, achieving a detection accuracy of 99.50%, albeit with a limited exploration of algorithms and features. Meanwhile, Sirigineedi et al. [30] employed a combination of neural network models and machine learning classifiers, achieving a commendable accuracy of 96.60% but faced challenges with the utilization of external URL features. Anil Kumar Yamarthy and Ch Koteswararao [31] have proposed a highly accurate phishing attack detection model. They balance data using Adv-SyN, extract features with DSelSa, optimize feature selection using OpGoA, and employ classification via MDepthNet. Additionally, recent studies [29,32–45] have collectively underscored the ongoing advancements in employing deep learning for robust phishing detection. These endeavors highlight the potential of deep learning models to improve sensitivity and accuracy in detecting phishing attacks. However, they also reveal areas for further exploration and refinement in addressing real-time analysis, adapting to emerging threats, and improving model generalization to novel attack patterns.

2.3. Web browser extension approaches

In the domain of web browser extension-based solutions for phishing detection, innovative studies have emerged to enhance real-time protection. Atimorathanna et al. [46] presented an extensive anti-phishing protection system, integrating a browser extension, an email detection plug-in, and a machine

Table 1. Summary of recent phishing detection methods from 2019 to 2024.

Authors	Used Algorithm	Challenges/ Limitations
Odeh et al. [3]	CatBoost, XGBoost, and LightGBM	Bias can occur in manual feature selection.
Qasem et al. [4]	SNN, DT	Extracting 111 features from a real-time URL is not feasible.
Qasem et al. [5]	Bagging trees, KNN, Boosted decision trees, subspace discriminator	Used 59 features from a URL
Sheikhi et al. [6]	PSO-XG Boost	Only 36000 Urls used
Hannousse et al. [7]	SVM, DT, LR, RF, Naive Bayes	Manual feature selection used
Rashid et al. [8]	SVM	Small dataset used
Basit et al. [9]	RF, K-NN, DT, and ANN	Used an open source dataset
Stobbs et al. [10]		
Bu and Cho [27]	CNN	Used Character level features
Korkmaz et al. [28]	CNN	Low Accuracy
Feng and Yue [29]	RNN	They used 17 features for classification
Sirigineedi et al. [30]	Neural Networks and M L algorithms	Third party URL features used.
Anil Kumar et al. [31]	Multi Head Depth wise Tern integrated-LSTM	Lacks dataset adaptability
Wang et al. [20]	DCNN	In real-time, detection can impact network performance
Manika and Shivani [21]	BiLSTM-GHA-CNN	Heavy approach
Saad [22]	GAN-LSTM-CNN	Not Suitable for real-time detection
Zaimi et al. [23]	XG Boost, LSTM, CNN	Maximum accuracy is 97.82%
Alshingiti et al. [24]	LSTM, CNN	Used small dataset
Sahingoz et al. [25]	ANN, CNN, RNN and attention networks	
Barath et al. [26]	CNN	real-time website for detecting phishing URLs is not feasible

learning-based phishing detection server. Maurya et al. [47] introduced a real-time browser plugin, combining whitelist matching, blacklist filtering, and machine learning-based prediction. Shah et al. [48] developed a Chrome extension utilizing machine learning to identify phishing URLs, achieving an accuracy rate of 89.6%. Sundaram et al. [49], Abiodun et al. [50], Gowda et al. [51], Lizhen [52] and others explored browser extensions, each contributing unique features to enhance user security.

These studies collectively contribute to the ongoing discourse surrounding phishing detection, offering diverse methodologies, insights, and potential avenues for future research in the field. While each approach has demonstrated efficacy, challenges persist, such as dataset limitations, algorithmic dependencies, and the need for continuous adaptation to evolving phishing techniques (Table 1). The proposed system in this work builds upon these foundations, integrating advanced machine learning techniques with a user-centric browser extension, setting a precedent for seamless and efficient phishing threat mitigation in the digital landscape.

3. Proposed methodlology

The crux of our methodology lies in the sophistication of our deep learning classifier model. Its effectiveness and adaptability are pivotal in safeguarding users from phishing threats. This model introduces a novel prediction module, Knowledge Distilled ELECTRA for Phishing URL Detection, which employs advanced knowledge distillation techniques to boost the effectiveness of ELECTRA.

3.1. Knowledge distilled ELECTRA for phishing URL detection

The main goal is to develop a more streamlined and resource-efficient iteration of ELECTRA tailored specifically for the task of detecting phishing URLs. Figure 2 shows the knowledge distillation framework that was used on the ELECTRA model. The goal was to take knowledge from a more complicated teacher model and put it into a small, effective student model that could find phishing URLs. For our knowledge distillation framework, we opt for the ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) model as the primary architecture (Figure 3). It has proven to be a state-of-the-art transformer-based model with exceptional capabilities in various natural language processing tasks. ELECTRA's self-attention mechanisms and transformer architecture let it understand long-distance dependencies and semantic relationships within sequences. This makes it a great choice for difficult tasks like finding phishing URLs.

3.1.1. Tokenization and input encoding

The initial phase of Knowledge Distilled ELECTA's phishing URL classification pipeline hinges on the intricate process of tokenization. This critical step dissects input URLs into individual units, such as words or subwords, creating a granular representation for the model. ELECTRA leverages its own tokenizer, specifically tailored for this transformer-based architecture, ensuring optimal vocabulary handling. This specialized tokenizer adeptly addresses the presence of rare words and subwords, ensuring they are accurately captured for subsequent processing [53].

Following tokenization, special tokens are added to the beginning and end of the sequence, playing crucial roles in the model's understanding of the input. These include:

- **Classification token (CLS)**: The Classification token serves as a starting point for the model, allowing it

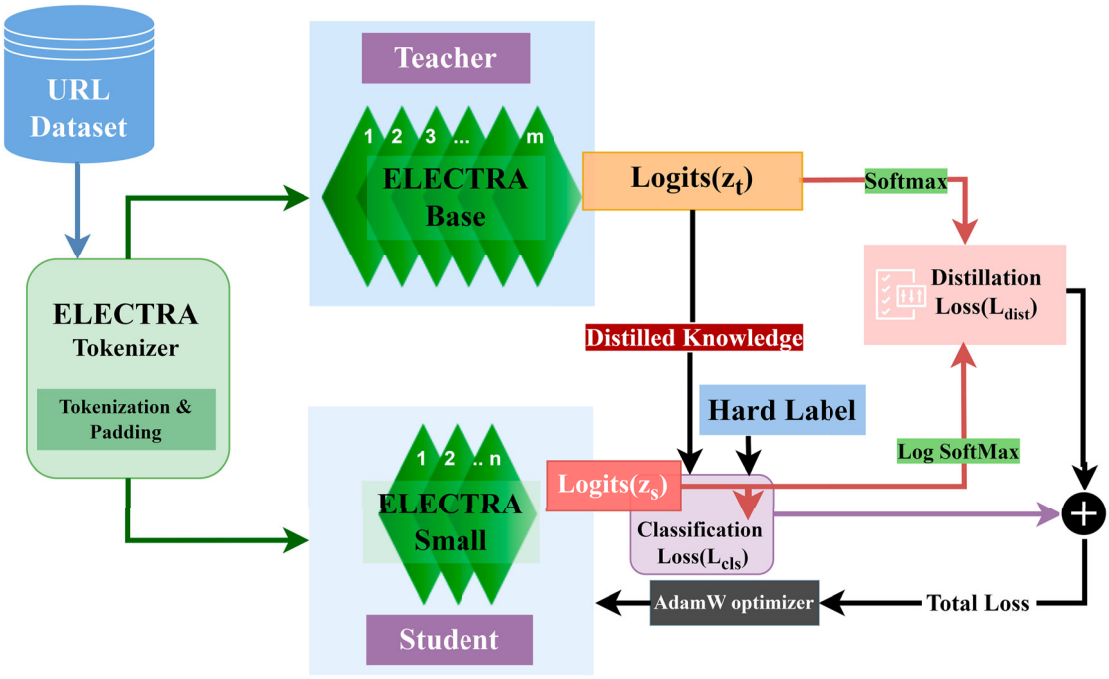

Figure 2. Knowledge distillation framework used on the ELECTRA model.

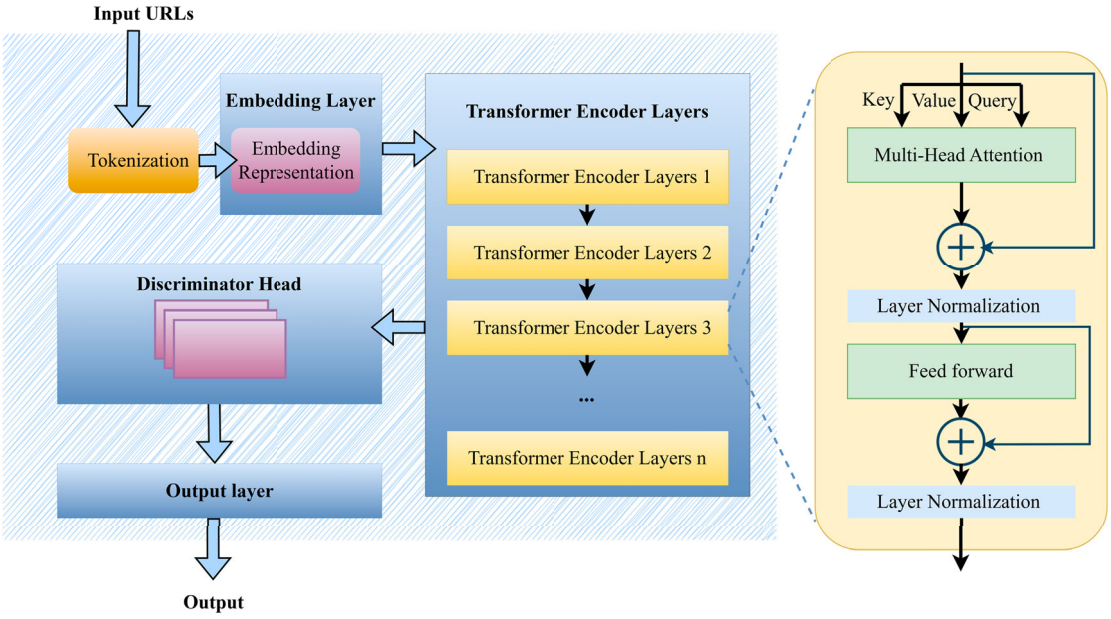

Figure 3. ELECTRA architecture.

to focus on the overall classification task rather than individual tokens.

- **Separator token (SEP)**: The Separator token denotes the end of one URL and the beginning of another, facilitating the model's processing of multiple URLs during batch training.

Subsequent to tokenization and special token insertion, the sequence undergoes a transformation known as input encoding. This process aims to convert the fragmented URLs into a format compatible with the neural network's inner workings. Two key outputs are produced during this step:

- **Input ids**: These represent numerical identifiers assigned to each individual token, enabling the model to efficiently decipher the sequence. Each token is represented by a unique index in the vocabulary. Let t_i be the ith token in the URL. The input ID for t_i is the index of t_i in the vocabulary:

$$\text{Input ID}(t_i) = \text{Index}(t_i) \tag{1}$$

- **Attention mask**: This binary mask serves a critical role in directing the model's focus. It assigns a value of 1 to tokens deemed relevant for analysis and 0 to those deemed irrelevant. This ensures the model

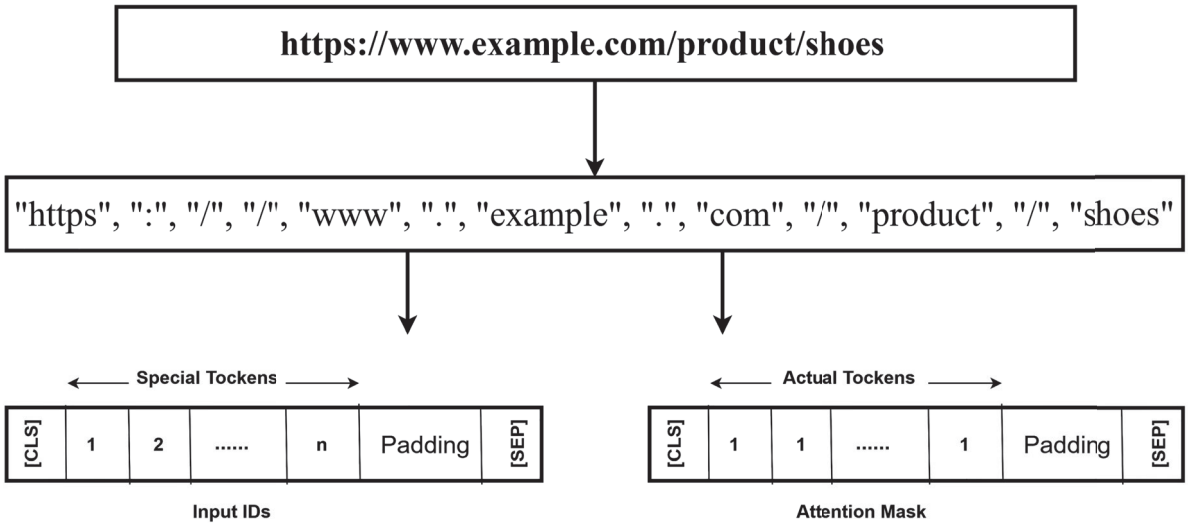

Figure 4. Tokenization and input encoding.

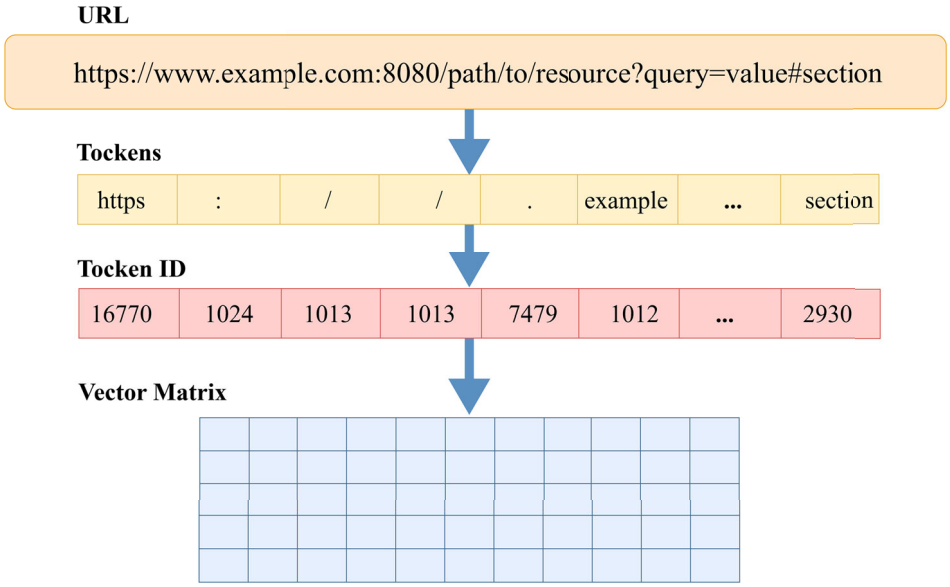

Figure 5. Embedding process.

allocates its computational resources efficiently by prioritizing key tokens for processing.

These meticulously encoded inputs, along with the strategically placed special tokens, form the bedrock upon which the subsequent prediction phase is built. The carefully crafted tokenization, special token insertion, and input encoding processes provide ELECTRA with the necessary foundation to perform its crucial task of phishing URL classification. The tokenization process is illustrated in Figure 4 through a comprehensive block diagram. For each token t_i in the URL, we represent it as a one-hot vector:

$$\text{Token One-Hot Encoding}(t_i) = [0, 0, \ldots, 1, \ldots, 0] \tag{2}$$

where the ith position is 1, indicating the position of the token in the vocabulary.

- **Embedding Process:** The embedding process involves representing each token with a continuous vector through the embedding matrix. The embedding matrix E is a trainable parameter in the model. It has dimensions (V, d), where V is the vocabulary size, and d is the dimensionality of the embedding vectors (Figure 5). The embedding process involves multiplying the one-hot encoded token vector with the embedding matrix. The embedding $\text{Embed}(t_i)$ for a token t_i is given by:

$$\text{Embed}(t_i) = \text{Token One-Hot Encoding}(t_i) \times E \tag{3}$$

This operation results in the continuous vector representation for the token t_i.

During training, the parameters of the embedding matrix E are updated using backpropagation and optimization algorithms to minimize a certain loss

function. The loss function involves the discrepancy between the predicted embeddings and the target embeddings, contributing to the learning of semantic representations. The obtained embeddings serve as the input to the subsequent Transformer Encoder Layers.

3.1.2. Transformer encoder layer with multihead attention, FFN, and layer normalization

Central to Distilled ELECTRA's prowess lies the Transformer Encoder Layer (Figure 3). This intricate structure orchestrates key operations, including Multihead Attention, Feedforward Neural Network (FFN), and Layer Normalization with Residual Connection. These mechanisms work in concert, enabling the Encoder Layer to capture and comprehend nuanced patterns within sequential data, serving as a vital cog in Distilled ELECTRA's representation learning engine.

(1) **Multihead Attention Mechanism**
The Transformer Block begins its operations with the Multihead Attention Mechanism. Given the input sequence X_i, the mechanism computes attention weights for each head (h) and forms a weighted sum of input vectors. The result, denoted as Multihead(X_i), is a concatenation of outputs from all heads, linearly transformed to produce the final output [54].

$$\text{Multihead}(X_i) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_H) \cdot W_O \tag{4}$$

Here, For each head h, the input sequence X_i undergoes linear transformations to create three sets of vectors: Q_h (Query), K_h (Key), and V_h (Value).

$$\begin{aligned} Q_h &= X_i \cdot W_{Qh}, \quad K_h = X_i \cdot W_{Kh}, \\ V_h &= X_i \cdot W_{Vh} \end{aligned} \tag{5}$$

Where head_h is computed as:

$$\text{head}_h = \text{Attention}(X_i \cdot W_{Qi}, X_i \cdot W_{Ki}, X_i \cdot W_{Vi}) \tag{6}$$

$$\text{Attention}(Q_h, K_h, V_h) = \text{softmax}\left(\frac{Q_h \cdot K_h^T}{\sqrt{d_h}}\right) \cdot V_h \tag{7}$$

Here, d_h is the dimensionality of the Key vectors in head h, and softmax applies the softmax function element-wise. W_{Qi}, W_{Ki}, and W_{Vi} are learnable linear transformations for query, key, and value projections, respectively. W_O is another learnable linear transformation for the output. The Attention function computes attention weights and the weighted sum.

(2) **Layer Normalization and Residual Connection (First Pass)**
Following the Multihead Attention Mechanism, the input sequence undergoes a process of Layer Normalization and Residual Connection. The initial normalization (LayerNorm_1) involves adding the original input to the output of the Multihead Attention Mechanism. This is followed by another layer of normalization (LayerNorm_2), further enhancing the model's ability to capture complex dependencies.

$$\text{LayerNorm}_1(X_i) = X_i + \text{Multihead}(X_i) \tag{8}$$

$$X'_i = \text{LayerNorm}_2(\text{LayerNorm}_1(X_i) + X_i) \tag{9}$$

(4) **Feedforward Neural Network (FFN)**
The output (X'_i) from the Layer Normalization and Residual Connection stage is then subjected to a Feedforward Neural Network (FFN). The FFN introduces non-linearity through the Rectified Linear Unit (ReLU) activation function. This stage involves two linear transformations and bias additions (Equation (10)), enhancing the model's capacity to extract hierarchical features.

$$X''_i = \text{ReLU}(X'_i \cdot W_{1i} + b_{1i}) \cdot W_{2i} + b_{2i} \tag{10}$$

Here, W_{1i}, b_{1i}, W_{2i}, and b_{2i} are learnable parameters specific to block i.

(5) **Layer Normalization and Residual Connection (Second Pass)**
Post-FFN, another round of Layer Normalization and Residual Connection is applied (LayerNorm_3). Similar to the first pass, the output is enriched by the addition of the initial input sequence, forming the final output of the Transformer Block.

$$\text{LayerNorm}_3(X'''_i) = X'''_i + X''_i \tag{11}$$

$$\text{Output}_i = \text{LayerNorm}_3(X'''_i) + X''_i \tag{12}$$

This orchestrated sequence of operations, involving Multihead Attention, FFN, and Layer Normalization with Residual Connection, is repeated across multiple Transformer Blocks in the model architecture. The layer-wise application of these operations enables the model to efficiently capture intricate patterns and dependencies within sequential data, forming the basis for its powerful representational learning capabilities. These outputs encapsulate contextualized representations of individual tokens within the input sequence. Let H be the matrix representation of these contextualized embeddings, where $H \in \mathbb{R}^{L \times D}$, with L representing the sequence length and D denoting the hidden size.

$$\text{Sequence embedding} = \frac{1}{L}\sum_{i=1}^{L} H_i \tag{13}$$

This operation computes the average of the embeddings along the sequence length dimension, resulting

in a condensed, fixed-size vector. Each H_i signifies the contextualized embedding of the ith token, capturing its significance within the broader sequence context.

This sequence embedding effectively captures essential information from the entire sequence, providing a contextually rich representation. This vector, derived from mean pooling, serves as a compact input for subsequent layers, including the sequence classification head.

3.1.3. Discriminator head

The Discriminator Head operates by transforming the high-dimensional Sequence embedding representation using a linear layer. This transformation involves learning a set of weights and biases that map the pooled output to a one-dimensional vector. Let SE denote the Sequence embedding, and let W and b represent the weights and biases of the linear layer, respectively. The operation of the Sequence Classification Head can be expressed as follows:

$$\text{Logits(z)} = SE \cdot W^T + b \tag{14}$$

Here, W is the weight matrix of dimensions $(D, 1)$, transposed for the matrix multiplication and, b is a bias term, a scalar. The output, denoted as Logits, represents the model's raw predictions before applying any activation function. After applying the sigmoid activation function to the raw predictions (Logits), the output can be denoted as Probabilities. Mathematically, this transformation is expressed as:

$$\text{Probabilities} = \sigma(\text{Logits}) \tag{15}$$

The resulting Probabilities vector represents the likelihood of the corresponding input sequence belonging to the positive class. The sigmoid function squashes the raw scores into the range [0, 1], providing a probability distribution.

3.1.4. Knowledge distillation

Knowledge distillation is a machine learning technique that involves transferring knowledge from a more complex teacher model to a simpler student model. This process leverages soft labels, which are probability distributions over classes provided by the teacher model, in addition to traditional hard labels. The smoothness of these distributions is controlled by a temperature parameter, allowing for more nuanced learning for the student. The training objective combines a standard classification loss with a distillation loss, where the latter measures the divergence between the student's predictions and the softened predictions of the teacher. The distillation loss in knowledge distillation is calculated using Kullback–Leibler (KL) divergence between the softmax distributions of the student and teacher logits. The overall loss is a weighted sum of these components, with the weighting factor (λ_{distill}) determining the trade-off between classification and distillation objectives. The equation for the Total loss is as follows:

$$\text{Classification Loss}(L_{\text{cls}}) = \text{CrossEntropy}(z_s, \text{labels}) \tag{16}$$

$$\text{Distillation Loss}(L_{\text{dist}}) = \text{KL}\left(\log_{\text{softmax}}\left(\frac{z_s}{T}\right), \text{softmax}\left(\frac{z_t}{T}\right)\right) \tag{17}$$

$$L_{\text{total}} = L_{\text{cls}} + \lambda_{\text{distill}} \times L_{\text{dist}} \tag{18}$$

Here, z_s represents the logits from the student model, z_t represents the logits from the teacher model, and T represents the temperature parameter. The KL divergence measures the information lost when the student's predictions are compared to those of the teacher, encouraging the student to mimic the softer knowledge encoded in the teacher's logits.

After calculating the total loss, the next steps involve backpropagation and optimization. Gradients are computed to understand parameter influences, and the AdamW optimization algorithm iteratively adjusts student model parameters to minimize the total loss. This process is repeated over multiple epochs, refining the student model to capture both ground truth labels and knowledge from the teacher model. Algorithm 1 describes the knowledge distillation process used to train a student model based on a teacher model.

3.2. System model

The proposed model is a multifaceted approach that encompasses various components, each meticulously designed to ensure the robustness and effectiveness of our AI-powered web browser extension for real-time phishing URL detection. The workflow, illustrated in Figure 6, smoothly progresses through a succinct series of steps meticulously explained.

(1) **User Installs and Activates Chrome Extension:** Users install and activate the Chrome extension, integrating it into their browser for real-time phishing detection.
(2) **URL Monitoring by the Extension:** The extension actively monitors URLs accessed by the user during their browsing sessions, capturing real-time data.
(3) **URL Preprocessing (Client Side):** Before sending a URL to the server, the client-side preprocessing module cleans the URL. This step optimizes data transfer and reduces server load.
(4) **Secure Data Transmission (Client Side):** The pre-processed URL features are securely transmitted to the server for analysis. Encrypted communication, such as HTTPS, is employed to

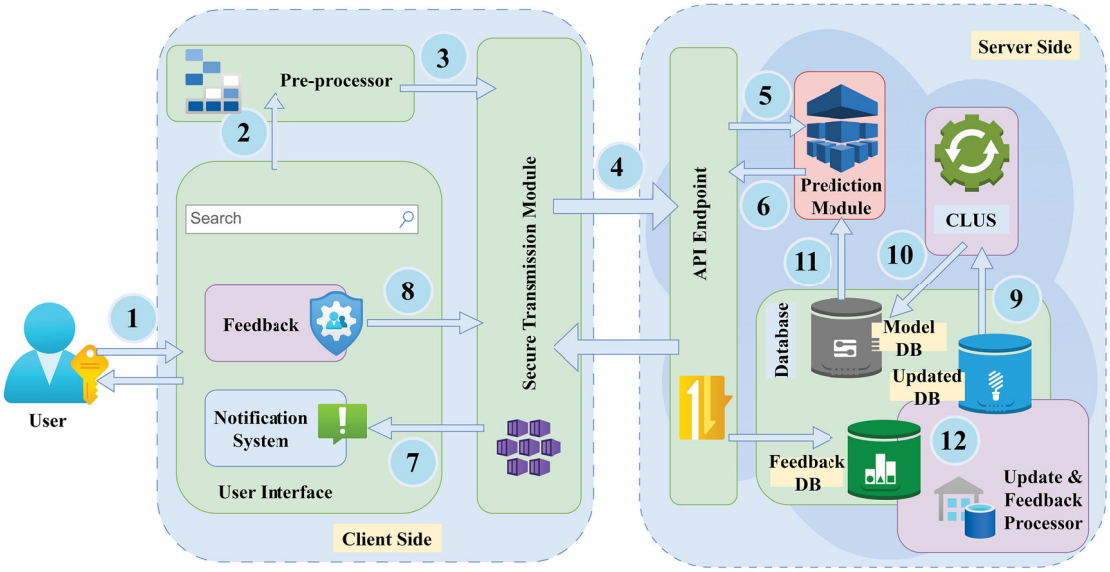

Figure 6. Architecture of the deep learning-based browser extension for real-time phishing URL detection.

Algorithm 1 Knowledge Distillation process of ELECTRA

Require: Teacher model with logits z_t, Student model with logits z_s, Temperature parameter T, Distillation loss weighting factor λ_{distill}, etc.

Ensure: Trained student model parameters Θ_S

Training Process:
Initialize student model parameters Θ_S
for each epoch t **do**
  Set teacher model to evaluation mode
  Set student model to training mode
  for each batch B_i in training data **do**
    $z_t = \text{Teacher}(x_i)$
    $z_s = \text{Student}(x_i)$
    $L_{\text{cls}} = \text{CrossEntropy}(z_s, \text{labels})$
    $L_{\text{dist}} = \text{KL}(\text{log\_softmax}(\frac{z_s}{T}), \text{softmax}(\frac{z_t}{T}))$
    $L_{\text{total}} = L_{\text{cls}} + \lambda_{\text{distill}} \times L_{\text{dist}}$
    Backpropagate gradients: $\nabla_{\Theta_S} L_{\text{total}}$
    Update student model parameters using optimizer
  end for
  Compute average training loss for epoch
end for
Evaluation:
Evaluate student model on validation/test data
Calculate evaluation metrics: accuracy, F1-score, precision, recall, FPR, FNR
Repeat:
while not converged **do**
  Continue training for next epoch
end while

preserve user privacy during data transmission. The server-side API endpoint receives the preprocessed URL features from the client, acting as the central hub for communication.

(5) **Knowledge Distilled ELECTRA Model Prediction (Server Side):** The Distilled ELECTRA-based prediction module processes the received URL and classifies the URL as safe or suspicious based on learned patterns.

(6) **Results Sent Back to Chrome Extension:** The prediction results are sent back to the Chrome extension through the API endpoint, providing real-time feedback on the safety status of the accessed URL

(7) **User Notifications (Client Side):** The Chrome extension displays real-time notifications to the user based on the prediction results, indicating whether the accessed URL is Safe, Suspicious, or Dangerous.

(8) **User Feedback Loop (Client Side):** The extension provides a user interface allowing users to report false positives/negatives, actively involving users in improving the system's accuracy.

(9) **Continuous Learning and Model Update (Server Side):** The server-side continuous learning system periodically re-trains the KD-ELECTRA model using new data, incorporating user feedback into the training dataset for ongoing improvement.

(10) **Continuous Learning from Updated Dataset:** The server periodically acquires an updated dataset that combines new labelled data with existing information.This dataset includes the latest insights into evolving phishing patterns, user behaviour, and feedback. The KD-ELECTRA model undergoes a retraining process using this updated dataset, enhancing its ability to adapt to emerging threats.

(11) **Saving the New Model to Model Database:** After completion of the retraining process, the newly updated KD ELECTRA model is saved in the model database.This database serves as a

repository for storing different versions of the model, allowing the system to maintain a history of improvements and changes over time. Each model version is timestamped or versioned to track the evolution of the model.

(12) **Loading the Updated Model to the Prediction Module:** The server ensures a seamless transition by loading the newly updated KD ELECTRA model into the prediction module. This step involves replacing the previous model with the most recent version, enabling the system to make predictions based on the latest knowledge.

(13) **Update and Feedback Processing (Server Side):** The update and feedback processor regularly checks the feedback database for new entries, using insights from user feedback to update the list of known phishing URLs and patterns in the update database.

4. Result and analysis

Evaluating the Knowledge-Distilled ELECTRA model has yielded insightful results, providing a comprehensive understanding of its effectiveness in detecting phishing attacks.

4.1. Experimental setup

The knowledge distillation experiments for training the ELECTRA model were conducted on a high-performance computing system. The system was equipped with an Intel Xeon w5-2465X processor running at 3.10 GHz, providing substantial computational power. The operating system utilized for the experiments was Microsoft Windows 11, ensuring a user-friendly environment for seamless execution. Accelerating the training process, an NVIDIA RTX A6000 graphics card was employed, leveraging GPU parallelism. The system boasted a remarkable 512 GB of installed RAM, facilitating the handling of large datasets and complex model architectures. This powerful hardware and software configuration created an optimal environment for efficient and effective model training.

4.1.1. Dataset collection & data cleaning

During the data collection phase, we acquired Several distinct datasets from Kaggle and PhishStorm. Upon acquisition of the dataset, a comprehensive data cleansing process was executed, encompassing several stages:

(1) **Elimination of Duplicates**: Systematic scanning and removal of duplicate URLs were performed to ensure the uniqueness of each entry and prevent redundancy.

(2) **Addressing Missing Values**: The dataset underwent meticulous examination for instances of missing or null values. Depending on the extent of missing data, rows with such values were either purged or methods like mean imputation or interpolation were applied.

(3) **Exclusion of Irrelevant Columns**: Columns deemed irrelevant for analysis, such as timestamps or extraneous metadata, were excised to streamline the dataset.

(4) **Ensuring Label Cohesion**: Rigorous validation was conducted to ensure the consistency and accuracy of labels denoting whether each URL indicated phishing or legitimate activity. This validation process involved cross-referencing the labels with the URLs themselves to affirm correctness.

(5) **Dataset Segmentation**: Subsequently, the cleansed dataset was partitioned into distinct subsets for training, validation, and testing purposes, facilitating subsequent tasks such as model training and performance evaluation.

Table 2. Dataset details.

Dataset	Sources	Count	URLs
A	Kaggle	11430	Phishing: 5717 Legitimate: 5713
B	Phishstorm [55]	95913	Phishing: 47904 Legitimate: 48009
C	Kaggle, Phish Tank, and Majestic million	450176	Phishing: 104438 Legitimate: 345738

Through these meticulous procedures, the dataset was thoroughly cleansed and primed for subsequent analysis and modelling endeavors, adhering to principles of academic integrity and originality. Specific details of the dataset can be found in the accompanying Table 2. The dataset comprises two columns, namely "url" and "type," where the "type" column encompasses two categories: "legitimate" and "phishing".

4.1.2. Hyper parameter details

The training of the Knowledge distilled Electra model involves key hyperparameters, summarized in Table 3. The number of epochs is set to 10, with a distillation weight of 0.5. The AdamW optimizer is employed with a learning rate of 0.000005. The model architecture comprises a teacher model ("google/electra-base-discriminator") and a student model ("google/electra-small-discriminator"). Tokenization parameters include a maximum sequence length of 512, with padding and truncation. The dataset is split into training (80%), validation (10%), and test (10%) sets. A batch size of 16 is used during training and evaluation. These hyperparameters collectively define the training configuration, influencing the model's performance in classifying URLs.

Table 3. Hyperparameter details for electra model training.

Category	Hyperparameter	Description	Values or Setting
Training Parameters	num_epochs	Number of training epochs	10
	lambda_distill	Weight assigned to the distillation loss	0.5
Optimizer	Learning Rate	Learning rate for the AdamW optimizer	5×10^{-5}
Data Processing	max_length	Maximum length of input sequences during tokenization	512
	padding	Padding applied during tokenization	Applied
	truncation	Sequences longer than max_length truncated	Applied
Data Split	Training Set Ratio	Percentage of the dataset used for training	80%
	Validation Set Ratio	Percentage of the dataset used for validation	10%
	Test Set Ratio	Percentage of the dataset used for testing	10%
Data Loader	batch_size	Number of samples in each batch during training/evaluation	16

4.1.3. Evaluation metrics

In the context of phishing URL classification, the performance of the model can be evaluated using several metrics. These metrics provide insights into how well the model is performing in distinguishing between phishing and legitimate URLs.

- **Accuracy**: Accuracy is a metric that evaluates the overall correctness of the model's predictions by determining the proportion of accurately classified URLs, encompassing both phishing and legitimate, relative to the total number of URLs assessed.

$$\text{Accuracy} = \frac{P_c + L_c}{P_c + L_c + P_{inc} + L_{inc}} \tag{19}$$

  Where:
  (1) P_c is the number of phishing URLs correctly classified as phishing.
  (2) L_c is the number of legitimate URLs correctly classified as legitimate.
  (3) P_{inc} is the number of legitimate URLs incorrectly classified as phishing.
  (4) L_{inc} is the number of phishing URLs incorrectly classified as legitimate.
- **Precision**: Precision assesses the accuracy of positive predictions generated by the model. Specifically in the domain of phishing URL classification, it gauges the proportion of accurately classified phishing URLs among all URLs that were predicted as phishing. Mathematically, precision is computed as the ratio of the number of phishing URLs correctly identified as phishing (P_c) to the sum of P_c and the number of legitimate URLs mistakenly classified as phishing (P_{inc}).

$$\text{Precision} = \frac{P_c}{P_c + P_{inc}} \tag{20}$$

- **Recall (Sensitivity)**: Recall, also known as sensitivity, assesses the model's ability to capture all positive instances. It measures the proportion of correctly classified phishing URLs among all actual phishing URLs.

$$\text{Recall} = \frac{P_c}{P_c + L_{inc}} \tag{21}$$

- **False Positive Rate (FPR)**: The False Positive Rate (FPR) measures the proportion of legitimate URLs that are incorrectly classified as phishing among all actual legitimate URLs.

$$\text{FPR} = \frac{P_{inc}}{P_{inc} + L_c} \tag{22}$$

- **False Negative Rate (FNR)**: The False Negative Rate (FNR) measures the proportion of phishing URLs that are incorrectly classified as legitimate among all actual phishing URLs.

$$\text{FNR} = \frac{L_{inc}}{P_c + L_{inc}} \tag{23}$$

- **F1-score**: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It considers both false positives and false negatives.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

The layout of the confusion matrix is depicted in Figure 7.These metrics collectively provide a comprehensive evaluation of the model's performance in phishing URL classification.

4.2. Evaluation of KD-ELECTRA with different datasets

The performance of KD-ELECTRA was thoroughly evaluated across three distinct datasets (A, B, and C) using various essential evaluation metrics over different training epochs. The results are presented in Table 4, offering a comprehensive overview of KD-ELECTRA's behaviour across diverse datasets and training iterations. Each dataset is examined at different training epochs, showcasing key metrics such as Accuracy, F1 Score, Precision, Recall, False Positive Rate (FPR), and False Negative Rate (FNR). These metrics allow for a detailed analysis of the model's performance under varying conditions.

Among the three datasets, KD-ELECTRA exhibits the most remarkable performance on Dataset C, where it achieves exceptional accuracy and precision. This

Table 4. Performance metrics for different epochs on each dataset.

Dataset	Epochs	Accuracy	F1 Score	Precision	Recall	FPR	FNR
	5	0.9466	0.9461	0.9657	0.9272	0.0336	0.0728
A	**20**	**0.9545**	**0.9534**	**0.9744**	**0.9333**	**0.0244**	**0.0667**
	50	0.9414	0.9403	0.9514	0.9296	0.0470	0.0704
	75	0.9379	0.9352	0.9715	0.9014	0.0261	0.0986
	5	0.9853	0.9855	0.9799	0.9911	0.0089	0.0205
B	**20**	**0.9868**	**0.9866**	**0.9915**	**0.9817**	**0.0083**	**0.0183**
	50	0.9732	0.9730	0.9760	0.9700	0.0235	0.0299
	100	0.9704	0.9702	0.9720	0.9684	0.0315	0.0315
	5	**0.9974**	**0.9943**	**0.9962**	**0.9925**	**0.0012**	**0.0075**
C	10	0.9966	0.9926	0.9997	0.9855	0.0001	0.0145
	15	0.9960	0.9960	0.9983	0.9938	0.0017	0.0062
	20	0.9955	0.9956	0.9979	0.9933	0.0021	0.0067

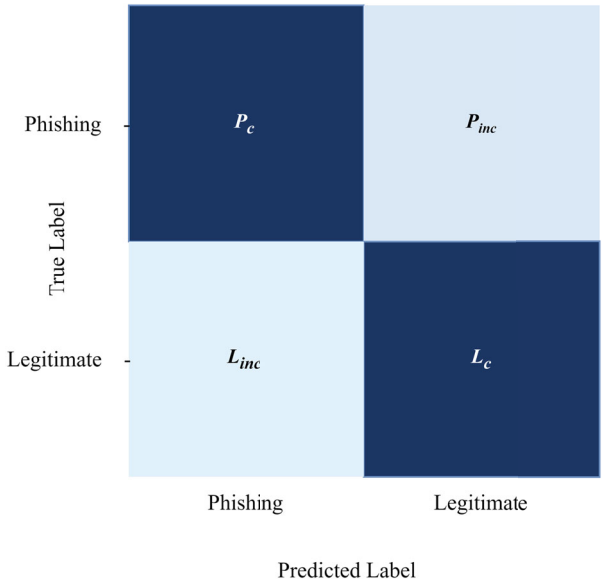

Figure 7. Layout of a confusion matrix.

dataset showcases the model's capability to deliver highly precise results, making it particularly well-suited for tasks where precision is of utmost importance. Following closely behind, Dataset B demonstrates strong performance metrics, especially in terms of F1 Score and Precision, indicating its reliability for various classification tasks. On the other hand, Dataset A, while still showing commendable performance, exhibits a slight decrease in its metrics compared to the other two datasets, particularly as the number of training epochs increases. In summary, KD-ELECTRA displays promising and robust performance across all datasets, with its standout performance on Dataset C, closely trailed by Dataset B. These insights offer valuable guidance for further optimizing and employing KD-ELECTRA in specific tasks, considering the desired balance between accuracy, precision, and recall requirements.

The Figure 8 depicts the False Positive Rate (FPR) and False Negative Rate (FNR) across three datasets (A, B, and C). It reveals a clear distinction in performance between Dataset C and Datasets A and B. While both FPR and FNR for Dataset C are significantly lower, hovering around 0.0012 and 0.0075 respectively, Datasets A and B show higher error rates.

The Figure 9 depicts the performance metrics across three datasets (A, B, and C). It reveals a significant improvement in performance in Dataset C. Dataset C achieves the highest values in all metrics, reaching near-perfect performance with an Accuracy of 0.9974, F1 Score of 0.9943, Precision of 0.9962, and Recall of 0.9925. Conversely, Dataset A exhibits the lowest performance, particularly in Recall (0.9333) which indicates a higher chance of missing true positive cases. Dataset B falls between the two, demonstrating a moderate improvement over Dataset A in all metrics. The Figure 10 provides a visual representation illustrating the F1-Recall comparison of KD-ELECTRA across three datasets, namely A, B, and C. Finally, Figure 11 presents the confusion matrix for the test dataset.

4.3. Prototype implementation

The system architecture of the phishing detection prototype is meticulously designed to seamlessly integrate the Google Chrome extension with a Flask-based server. This client-server model ensures efficient communication and real-time processing of URLs. Our implementation leverages a carefully selected set of technologies to ensure the robustness and efficiency of the phishing detection system. JavaScript, as the primary scripting language, enables dynamic user interaction and real-time URL monitoring. Chrome Extension APIs provide access to essential browser features for smooth integration. HTML and CSS are employed to structure and style the user interface, offering a visually appealing and user-friendly extension. On the server side, Flask serves as the backend framework, efficiently handling API requests. The KD ELECTRA model acts as the prediction module. An SQLite database stores critical information, including the latest model version, user feedback, and the existing URL database. A dedicated processor component is included for user feedback integration. Security measures include HTTPS for encrypted communication and URL anonymization to preserve user privacy. Figure 12 illustrates how the

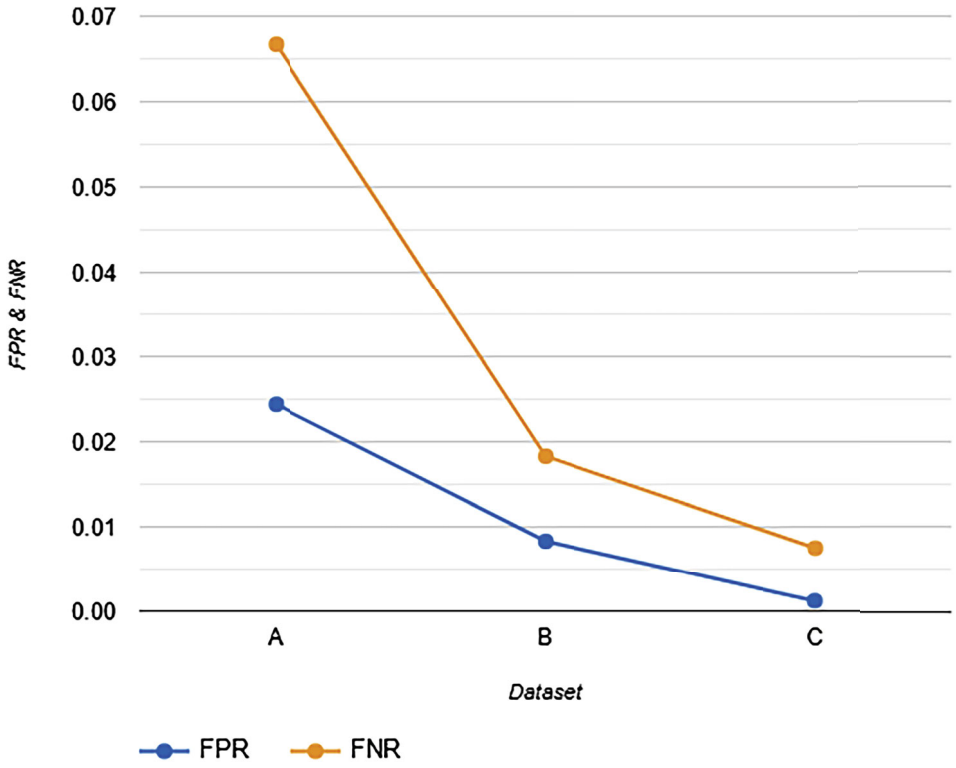

Figure 8. The FPR and FNR values of the KD ELECTRA model using different datasets.

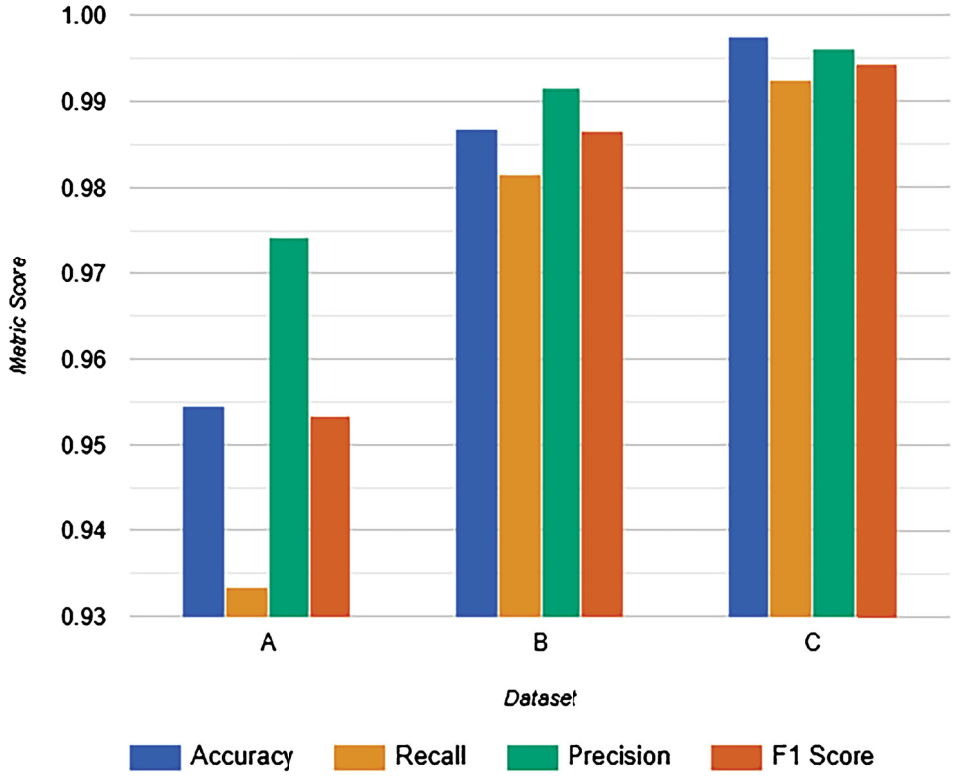

Figure 9. The best performance metrics of the KD ELECTRA model using different datasets.

extension displays a legitimate URL, while Figure 13 demonstrates its representation of a phishing URL.

4.4. Comparison with baseline models

We compare the performance of our proposed KD-ELECTRA model with the baseline models across all datasets. Table 5 presents the performance metrics of different models on each dataset. For Dataset A, KD-ELECTRA achieves an accuracy of 95.45%, surpassing all other models including LSTM, GRU, CNN, and Distil-BERT. Additionally, KD-ELECTRA demonstrates high precision and recall values, indicating its effectiveness in accurately detecting phishing

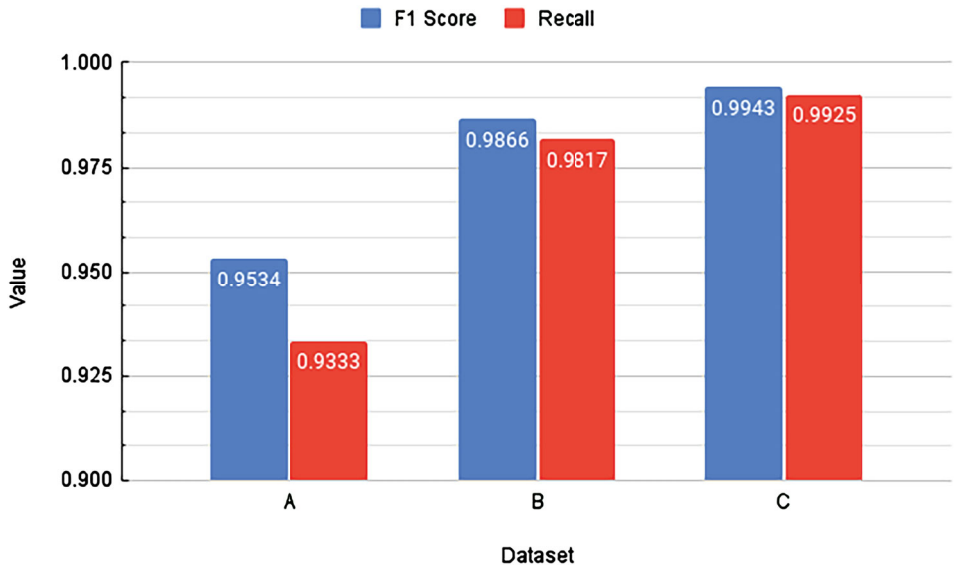

Figure 10. The F1-Recall comparison of the KD ELECTRA model using different datasets.

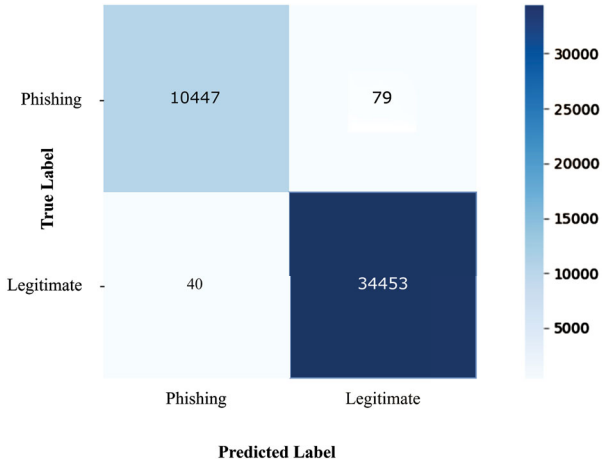

Figure 11. Confusion matrix of KD-ELECTRA on Dataset C.

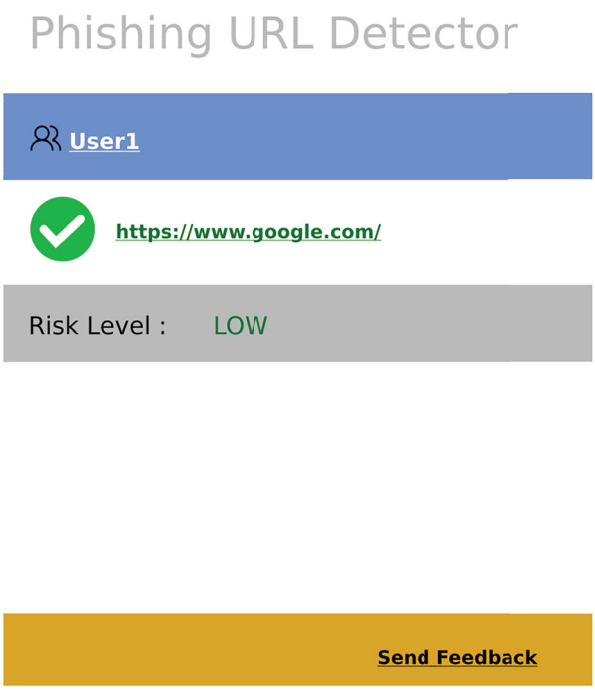

Figure 12. Model detecting legitimate URL.

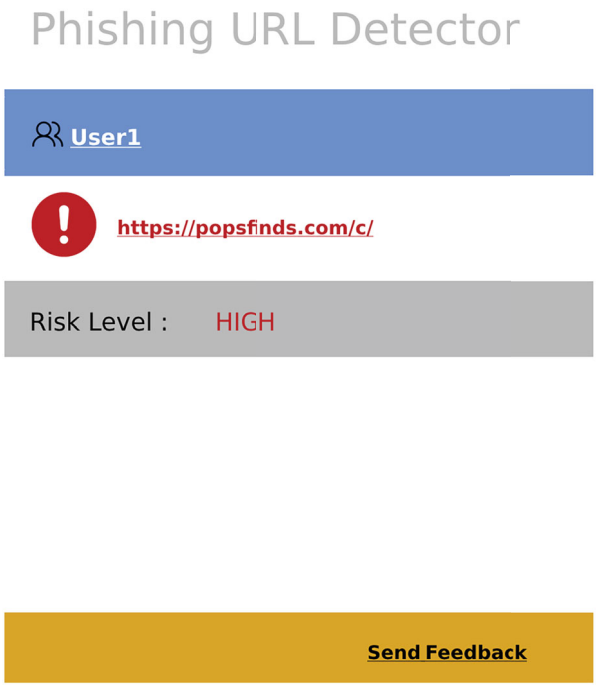

Figure 13. Model detecting phishing URL URL.

URLs. In Dataset B, KD-ELECTRA continues to outperform the baseline models with an accuracy of 98.68%. This superior performance is evident across all metrics, highlighting the robustness of KD-ELECTRA in phishing detection tasks. For Dataset C, KD-ELECTRA achieves exceptional performance with an accuracy of 99.74%. Notably, KD-ELECTRA exhibits the lowest false positive rate (FPR) and false negative rate (FNR) among all models, underscoring its reliability in distinguishing phishing URLs from legitimate ones. Furthermore, Figure 14 displays a column diagram showing accuracy, F-score, recall, and precision metrics across various models. Figure 15 presents the recall-F1 graph, revealing the balance between recall and F1 score. Additionally, Figure 16

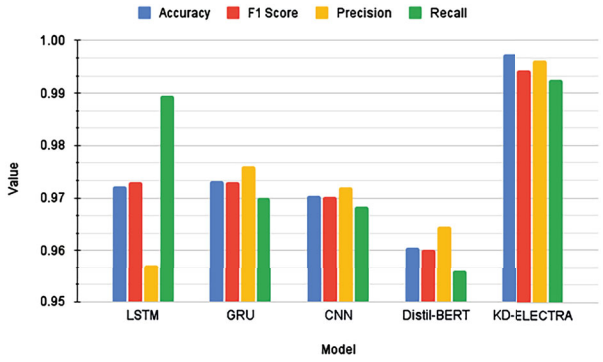

Figure 14. Comparison of baseline models with KD ELECTRA.

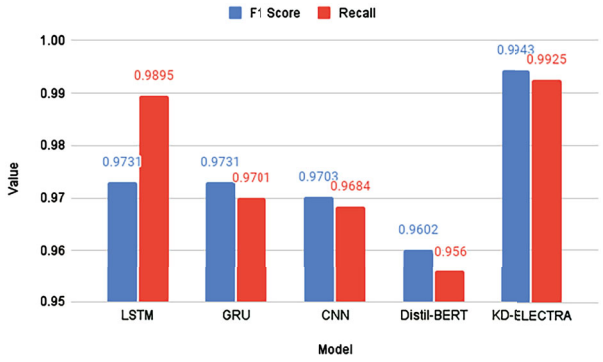

Figure 15. Comparison of baseline model's F1 score and recall with KD ELECTRA.

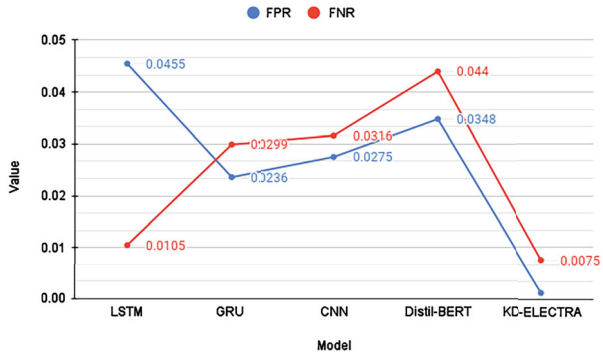

Figure 16. Comparison of baseline model's FNR and FPR with KD ELECTRA.

shows the FNR-FPR graph, illustrating the relationship between false negative rate (FNR) and false positive rate (FPR) across models. These visualizations offer valuable insights for comprehensive analysis and comparison of model performance. Overall, these comparison results demonstrate the superior performance of KD-ELECTRA compared to the baseline models, reaffirming its potential for real-world applications in phishing detection.

Table 6 present a comparison of proposed KD-ELECTRA powered browser extension with other baseline models. Our model achieves a remarkable accuracy of 99.74%, surpassing the accuracies reported by the models referenced in [51] (94.63%) and [47] (98.10%). Notably, while the baseline models rely on traditional machine learning algorithms such as Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression (LR), our approach leverages the advanced capabilities of KD-ELECTRA. One key advantage of our model is its support for real-time phishing detection, aligning with the needs of browser extensions where timely identification of phishing URLs is essential. Furthermore, our browser extension incorporates user feedback, a feature absent in the baseline models, which enhances its adaptability to evolving phishing tactics. Additionally, the continuous update mechanism embedded in our model ensures its effectiveness against emerging threats, a crucial aspect for maintaining security in dynamic online environments. Overall, our KD-ELECTRA powered browser extension stands out as a robust solution for real-time phishing detection, offering superior accuracy and advanced features to safeguard users against cyber threats.

4.5. Discussion

The proliferation of URL-based phishing attacks has outpaced the capabilities of traditional detection methods, particularly those relying on blacklist-based approaches. These conventional techniques suffer from a lag in response time and are often incapable of identifying newly created malicious URLs until they are reported by users or other sources. Our proposed system, integrating a Knowledge Distilled ELECTRA model, aims to overcome these limitations by leveraging deep learning techniques for accurate and timely phishing detection.

Proposed approach is distinguished by its dual components: a sophisticated URL classification model and a user-centric browser extension. The Knowledge Distilled ELECTRA model is particularly effective due to its ability to understand the semantic and syntactic patterns in URLs, thus offering a significant advancement over traditional heuristic and signature-based methods. The model's impressive performance, with a 99.74% accuracy and a 99.43% F1-score, highlights its potential to significantly reduce false positives and negatives in phishing detection. Furthermore, the integration of this model into a Chrome browser extension provides users with instant feedback on the legitimacy of URLs, enhancing their ability to avoid phishing attacks. This real-time alert mechanism is crucial in empowering users to make informed decisions and reduce the risk of falling victim to phishing scams.

4.5.1. Real-world application and user experience

The user feedback loop integrated into our system is a novel feature that allows for continuous improvement of the model's accuracy and relevance. By analyzing the feedback provided by users on false positives and negatives, the model can be retrained and fine-tuned

Table 5. Performance metrics of different models on each dataset.

Model	Dataset	Accuracy	F1 Score	Precision	Recall	FPR	FNR
	A	0.9309	0.9273	0.9692	0.8889	0.0278	0.1111
LSTM	B	0.9484	0.9467	0.9704	0.9242	0.0278	0.0758
	C	0.9723	0.9731	0.9572	0.9895	0.0455	0.0105
	A	0.9370	0.9284	0.9478	0.9380	0.0739	0.0521
GRU	B	0.9528	0.9522	0.9556	0.9489	0.0434	0.0511
	C	0.9733	0.9731	0.9761	0.9701	0.0236	0.0299
	A	0.9283	0.9263	0.9450	0.9083	0.0521	0.0917
CNN	B	0.9510	0.9513	0.9399	0.9630	0.0609	0.0370
	C	0.9705	0.9703	0.9721	0.9684	0.0275	0.0316
	A	0.9326	0.9304	0.9555	0.9067	0.0417	0.0933
Distil-BERT	B	0.9414	0.9403	0.9514	0.9296	0.0470	0.0704
	C	0.9606	0.9602	0.9645	0.9560	0.0348	0.0440
	A	0.9545	0.9534	0.9744	0.9333	0.0244	0.0667
KD-ELECTRA	B	0.9868	0.9866	0.9915	0.9817	0.0083	0.0183
	C	**0.9974**	**0.9943**	**0.9962**	**0.9925**	**0.0012**	**0.0075**

Table 6. Comparison of proposed model with existing models.

Model	Accuracy	Algorithm Used	Real-time	User Feedback	Continuous Update
[51]	0.9463	RF	✓	×	×
[47]	0.9810	RF,SVM & LR	✓	×	×
Proposed Model	0.9974	KD-ELECTRA	✓	✓	✓

to adapt to evolving phishing tactics. This iterative process ensures that the model remains robust and effective over time, keeping pace with the dynamic nature of cyber threats. The implementation of our system as a Chrome browser extension ensures a seamless user experience, as it operates unobtrusively in the background and alerts users only when a potential threat is detected. This design choice minimizes user fatigue and enhances the likelihood of users heeding the warnings provided by the system. Compared to existing solutions in the market, our system offers several advantages. While many anti-phishing tools rely on static databases of known malicious URLs, our system dynamically analyzes URLs in real-time, providing a proactive defense mechanism. Additionally, the use of deep learning models allows for a more nuanced understanding of phishing techniques, which are often designed to evade traditional detection methods.

Despite the strengths of our proposed system, there are potential challenges that need to be addressed. One such challenge is the continuous evolution of phishing techniques, which may involve more sophisticated obfuscation methods or the use of legitimate-looking URLs to deceive users. To counter these tactics, future work will focus on incorporating additional features and data sources into the model, such as domain age, hosting information, and content analysis.

4.5.2. Future scope

To build on the findings of this study, future research could focus on several areas. One potential direction is the integration of additional features, such as user behaviour metrics and website content analysis, to enhance the model's predictive capabilities. Incorporating more diverse datasets could also improve the model's robustness and ensure its applicability across various scenarios. Additionally, exploring the application of KD-ELECTRA to other cyber security tasks, such as malware detection and intrusion detection, could provide valuable insights into its broader utility. Research could also focus on developing strategies to mitigate the potential biases introduced during model training and evaluate the model's performance in different real-world environments. Additionally, investigating and reporting detection overhead will be essential for assessing the model's practical efficiency and computational impact.

5. Conclusion

In conclusion, our innovative approach to combating URL-based phishing attacks presents a significant leap forward in cyber security. By combining a cutting-edge deep learning model KD-ELECTRA with a user-friendly Chrome browser extension, we have successfully developed a robust system capable of instantly detecting and alerting users to potential threats with unparalleled accuracy. Our system's outstanding performance, boasting a remarkable 99.74% accuracy and a 99.43% F1-score on a diverse dataset of 450,176 URLs, underscores its efficacy in safeguarding users' digital security. Through real-time feedback mechanisms and continuous model enhancement via user input, we ensure our solution remains adaptive and responsive to emerging threats. This work not only sets a new standard for phishing detection but also empowers users to navigate the online landscape with confidence and informed decision-making. By prioritizing user safety and seamlessly integrating advanced technology, we pave the way for a more secure digital future.

Acknowledgments

I would like to express my gratitude to Dr. Arthi B, Associate Professor at SRM IST, for her invaluable guidance and unwavering support throughout the course of this research.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

ORCID

K. S. Jishnu http://orcid.org/0000-0003-2667-4858
B. Arthi http://orcid.org/0000-0001-6676-8993

References

[1] Naqvi B, Perova K, Farooq A, et al. Mitigation strategies against the phishing attacks: a systematic literature review. Comput Secur. 2023;132:103387. doi: 10.1016/j.cose.2023.103387

[2] Goenka R, Chawla M, Tiwari N. A comprehensive survey of phishing: mediums, intended targets, attack and defence techniques, and a novel taxonomy. Int J Inf Secur. 2023;23:819–848. doi: 10.1007/s10207-023-00768-x.

[3] Odeh A, Al-Haija QA, Aref A, et al. Comparative study of catboost, xgboost, and lightgbm for enhanced URL phishing detection: a performance assessment. J Internet Serv Inf Secur. 2023;13:1–11. doi: 10.58346/JISIS.2023.I4.001

[4] Al-Haija QA, Badawi AA. URL-based phishing websites detection via machine learning. In: 2021 International Conference on Data Analytics for Business and Industry (ICDABI), Sakheer, Bahrain. 2021. p. 644–649. doi: 10.1109/ICDABI53623.2021.9655851

[5] Al-Haija QA, Al-Fayoumi M. An intelligent identification and classification system for malicious uniform resource locators (URLs). Neural Comput Appl. 2023;35:16995–17011. doi: 10.1007/s00521-023-08592-z

[6] Sheikhi S, Kostakos P. Safeguarding cyberspace: enhancing malicious website detection with psooptimized xgboost and firefly-based feature selection. Comput Secur. 2024;142:103885. doi: 10.1016/j.cose.2024.103885

[7] Hannousse A, Yahiouche S. Towards benchmark datasets for machine learning based website phishing detection: an experimental study. Eng Appl Artif Intell. 2021;104:104347. doi: 10.1016/j.engappai.2021.104347

[8] Rashid J, Mahmood T, Nisar MW, et al. Phishing detection using machine learning technique. In: 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia. 2020. p. 43–46. doi: 10.1109/SMART-TECH49988.2020.00026

[9] Basit A, Zafar M, Javed AR, et al. A novel ensemble machine learning method to detect phishing attack. In: 2020 IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan. 2020. p. 1–5. doi: 10.1109/INMIC50486.2020.9318210

[10] Stobbs J, Issac B, Jacob SM. Phishing web page detection using optimised machine learning. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China. 2020. p. 483–490. doi: 10.1109/TrustCom50675.2020.00072

[11] Abedin NF, Bawm R, Sarwar T, et al. Phishing attack detection using machine learning classification techniques. In: 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India. 2020. p. 1125–1130. doi: 10.1109/ICISS49785.2020.9315895

[12] Alam MN, Sarma D, Lima FF, et al. Phishing attacks detection using machine learning approach. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India. 2020. p. 1173–1179. doi: 10.1109/ICSSIT48917.2020.9214225

[13] Sindhu S, Patil SP, Sreevalsan A, et al. Phishing detection using random forest, SVM and neural network with backpropagation. In: 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India. 2020. p. 391–394. doi: 10.1109/ICSTCEE49637.2020.9277256

[14] Kasim Ö. Automatic detection of phishing pages with event-based request processing, deep-hybrid feature extraction, and light gradient boosted machine model. Telecommun Syst. 2021;78:103–115. doi: 10.1007/s11235-021-00799-6

[15] Kumar M, Kondaiah C, Pais AR, et al. Machine learning models for phishing detection from TLS traffic. Cluster Comput. 2023;26:3263–3277. doi: 10.1007/s10586-023-04042-6

[16] Awasthi A, Goel N. Phishing website prediction using base and ensemble classifier techniques with cross-validation. Cybersecurity. 2022;5:22. doi: 10.1186/s42400-022-00126-9

[17] Li Y, Yang Z, Chen X, et al. A stacking model using URL and html features for phishing webpage detection. Future Gener Comput Syst. 2019;94:27–39. doi: 10.1016/j.future.2018.11.004

[18] Jha AK, Muthalagu R, Pawar PM. Intelligent phishing website detection using machine learning. Multimed Tools Appl. 2023;82:29431–29456. doi: 10.1007/s11042-023-14731-4

[19] Lansley M, Mouton F, Kapetanakis S, et al. Seader++: social engineering attack detection in online environments using machine learning. J Inf Telecommun. 2020;4(3):346–362. doi: 10.1080/24751839.2020.1747001

[20] Wang Z, Ren X, Li S, et al. A malicious URL detection model based on convolutional neural network. Secur Commun Netw. 2021;2021(1):5518528. doi: 10.1155/2021/5518528

[21] Nanda M, Goel S. URL based phishing attack detection using BiLSTM-gated highway attention block convolutional neural network. Multimed Tools Appl. 2024;83:69345–69375. doi: 10.1007/s11042-023-17993-0

[22] Al-Ahmadi S, Alotaibi A, Alsaleh O. PDGAN: phishing detection with generative adversarial networks. IEEE Access. 2022;10:42459-42468. doi: 10.1109/ACCESS.2022.3168235

[23] Zaimi R, Hafidi M, Lamia M. A deep learning mechanism to detect phishing URLs using the permutation importance method and SMOTE-Tomek link. J Supercomput. 2024;80:17159–17191. doi: 10.1007/s11227-024-06124-7

[24] Alshingiti Z, Alaqel R, Al-Muhtadi J, et al. A deep learning-based phishing detection system using CNN,

LSTM, and LSTM-CNN. Electronics. 2023;12(1):232. doi: 10.3390/electronics12010232
[25] Sahingoz OK, Buber E, Kugu E. Dephides: deep learning based phishing detection system. IEEE Access. 2024;12:8052–8070. doi: 10.1109/ACCESS.2024.3352629
[26] Barath S, Jaikrishnan J, Divas AS, et al. Baitnet: a deep learning approach for phishing detection. In: 2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, India. 2023. p. 1–8. doi: 10.1109/ICMNWC60182.2023.10436016
[27] Bu S-J, Cho S-B. Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing URL detection. Electronics. 2021;10(12):1492. doi: 10.3390/electronics10121492
[28] Korkmaz M, Kocyigit E, Sahingoz OK, et al. Phishing web page detection using N-gram features extracted from URLs. In: 2021 3rd International Congress on Human–Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey. 2021. IEEE. p. 1–6. doi: 10.1109/HORA52670.2021.9461378
[29] Feng T, Yue C. Visualizing and interpreting RNN models in URL-based phishing detection. In: Proceedings of the 25th ACM Symposium on Access Control Models and Technologies. SACMAT '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 13–24. doi: 10.1145/3381991.3395602
[30] Sirigineedi SS, Soni J, Upadhyay H. Learning-based models to detect runtime phishing activities using URLs. In: Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis. ICCDA 2020. New York, NY, USA: Association for Computing Machinery; 2020. p. 102–106.doi: 10.1145/3388142.3388170
[31] Yamarthy AK, Koteswararao CM. MDepthNet based phishing attack detection using integrated deep learning methodologies for cyber security enhancement. Cluster Comput. 2024;27:6377–6395. doi:10.1007/s10586-024-04313-w.
[32] Hussain M, Cheng C, Xu R, et al. CNN-fusion: an effective and lightweight phishing detection method based on multi-variant convnet. Inf Sci. 2023;631:328–345. doi: 10.1016/j.ins.2023.02.039
[33] Bozkir AS, Dalgic FC, Aydos M. Grambeddings: a new neural network for URL based identification of phishing web pages through N-gram embeddings. Comput Secur. 2023;124:102964. doi: 10.1016/j.cose.2022.102964
[34] Rendón-Segador FJ, Álvarez-García JA, Varela-Vaca AJ. Paying attention to cyber-attacks: a multi-layer perceptron with self-attention mechanism. Comput Secur. 2023;132:103318. doi: 10.1016/j.cose.2023.103318
[35] Singh S, Singh MP, Pandey R. Phishing detection from URLs using deep learning approach. In: 2020 5th International Conference on Computing, Communication and Security (ICCCS), Patna, India. 2020. p. 1–4. doi: 10.1109/ICCCS49678.2020.9277459
[36] Feng J, Zou L, Ye O, et al. Web2vec: phishing webpage detection method based on multidimensional features driven by deep learning. IEEE Access. 2020;8:221214–221224. doi: 10.1109/ACCESS.2020.3043188
[37] Opara C, Wei B, Chen Y. Htmlphish: enabling phishing web page detection by applying deep learning techniques on html analysis. In: 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK. 2020. p. 1–8. doi: 10.1109/IJCNN48605.2020.9207707
[38] Wei W, Ke Q, Nowak J, et al. Accurate and fast URL phishing detector: a convolutional neural network approach. Comput Netw. 2020;178:107275. doi: 10.1016/j.comnet.2020.107275
[39] AlEroud A, Karabatis G. Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In: Proceedings of the Sixth International Workshop on Security and Privacy Analytics. IWSPA '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 53–60. doi: 10.1145/3375708.3380315
[40] Abdelnabi S, Krombholz K, Fritz M. Visualphishnet: zero-day phishing website detection by visual similarity. In: CCS '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 1681–1698. doi: 10.1145/3372297.3417233
[41] Saha I, Sarma D, Chakma RJ, et al. Phishing attacks detection using deep learning approach. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India. 2020. p. 1180–1185. doi: 10.1109/ICSSIT48917.2020.9214132
[42] Al-Ahmadi S. A deep learning technique for web phishing detection combined URL features and visual similarity. Int J Comput Netw Commun (IJCNC). 2020;12(5). doi:10.5121/ijcnc.2020.12503.
[43] Mughaid A, AlZu'bi S, Hnaif A, et al. An intelligent cyber security phishing detection system using deep learning techniques. Cluster Comput. 2022;25:3819–3828. doi: 10.1007/s10586-022-03604-4
[44] Aslam S, Aslam H, Manzoor A, et al. AntiPhishStack: LSTM-based stacked generalization model for optimized phishing URL detection. Symmetry. 2024;16(2):248. doi: 10.3390/sym16020248
[45] Mandapati KS, Meesala S, Maddela D, et al. A hybrid transformer ensemble approach for phishing website detection. In: 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India. 2023. p. 1–8. doi: 10.1109/ICSSAS57918.2023.10331880
[46] Niroshan Atimorathanna D, Shehan Ranaweera T, Devdunie Pabasara RAH, et al. NoFish; total anti-phishing protection system. In: 2020 2nd International Conference on Advancements in Computing (ICAC), Malabe, Sri Lanka. Vol. 1. 2020. p. 470–475. doi: 10.1109/ICAC51239.2020.9357145
[47] Maurya S, Saini HS, Jain A. Browser extension based hybrid anti-phishing framework using feature selection. Int J Adv Comput Sci Appl. 2019;10(11):579–588. doi:10.14569/IJACSA.2019.0101178.
[48] Shah B, Dharamshi K, Patel MB, et al. Chrome extension for detecting phishing websites. 2020. Available from: https://api.semanticscholar.org/CorpusID:230125176.
[49] Sundaram KM, Sasikumar R, Meghana AS, et al. Detecting phishing websites using an efficient feature-based machine learning framework. Rev Geintec-Gestao Inovacao Tecnol. 2021;11(2):2106–2112.
[50] Abiodun O, Sodiya A, Kareem S, et al. Linkcalculator–an efficient link-based phishing detection tool. Acta Inf Malays. 2020;4(2):37–44. doi: 10.26480/aim.02.2020.37.44
[51] Mohith Gowda HR, Adithya MV, Gunesh Prasad S, et al. Development of anti-phishing browser based on random forest and rule of extraction framework.

Cybersecurity. 2020;3:20. doi: 10.1186/s42400-020-00059-1

[52] Tang L, Mahmoud QH. A deep learning-based framework for phishing website detection. IEEE Access. 2022;10:1509–1521. doi: 10.1109/ACCESS.2021.3137636

[53] Clark K, Luong M-T, Le QV, et al. Electra: pre-training text encoders as discriminators rather than generators. 2020. arXiv preprint arXiv:2003.10555. doi: 10.48550/arXiv.2003.10555

[54] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Adv Neural Inf Process Syst. 2017;30:5998–6008.

[55] Marchal S. PhishStorm – phishing/legitimate URL dataset. Aalto University. urlset(v.zip). 2014. doi: 10.24342/f49465b2-c68a-4182-9171-075f0ed797d5