VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2025, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 13, Number 4, October-December 2025 *pp:82-96*

Hybrid Deep Learning Approaches Enable Intrusion Detection System for Zero-Day Phishing Detection

Nimra Shaik[1], **Ambreen Hussain**[2], **Qurat-ul-ain Mastoi**[3], **Asif Aziz Memon**[1], **Atif Jamil**[1], **Abdullah Lakhan**[1*]

[1]Department of Cybersecurity, Dawood University of Engineering and Technology, Karachi, Sindh, 74800, Pakistan; [2]Department of Computer Science, Birmingham City University, Birmingham, B4 7BD, United Kingdom; [3]School of Computer Science and Creative Technologies, University of the West of England Bristol, BS16 1QY, United Kingdom.

Keywords:*Phishing, Zero Day Attacks, Intrusion Detection System, Deep Learning, CNN, LSTM and Auto-Encoder.*

Journal Info:
Submitted:
November 24, 2025
Accepted:
December 05, 2025
Published:
December 15, 2025

Abstract

These days, The website Uniform Resource Locator (URL) is widely used for accessing and navigating online information. However, the rise of AI-generated fake content, scams, counterfeit URLs, and other cyberattacks has significantly increased phishing-related threats. These fake URLs are difficult to identify because phishing links often resemble legitimate URLs. Consequently, both known and unknown (zero-day) phishing attacks remain difficult to detect in practice.This paper presents a hybrid deep learning–based intrusion detection system capable of detecting both known and zero-day phishing URLs. The objective is to provide users with absolute URLs while protecting them from fake ones. Another goal is to design an adaptive intrusion detection system (IDS) that combines static analysis, signature-based detection, and heuristic methods to identify phishing URLs. The proposed method, named the Zero-Phishing Convolutional Neural Network and Long Short-Term Memory (ZP-CNN-LSTM) algorithm, consists of three distinct schemes. For instance, static analysis rules are based on regular expressions, signatures for convolutional neural networks (CNNs), and zero-day detection using LSTM and autoencoders. We tested our project in the VR and AR research laboratory on 2 million testbed URLs, determined whether they were real or fake with respect to phishing, and predicted their phishing patterns. Results show that the proposed method achieves 98% higher accuracy than existing phishing detection methods in practice.

***Correspondence author email address:** abdullah.lakhan@duet.edu.pk
DOI: 10.21015/vtse.v13i4.2287

1 Introduction

Today, the internet supports a significant portion of critical activities across various sectors. Financial sectors, including retail and banking, increasingly provide services online due to the rise of digital transactions. Although this shift improves accessibility for organizations, it also introduces numerous security vulnerabilities. Because of extensive internet connectivity and open network architectures, the rate of cyberattacks continues to rise [1].

Existing cyberattack systems, such as machine learning, deep learning, and heuristics, are suggested [2–5]. These methods only address known and signature-based attacks in the network and largely ignore zero-day attacks. These zero-day threats can rapidly alter their behaviour multiple times in their early stages to evade network intrusion detection systems (NIDS). In fact, even well-designed, frequently updated signature-based NIDS cannot detect zero-day threats because they lack an adequate, adaptive signature database that can keep pace with the evolving threat landscape. The system requires training based on a malicious traffic dataset that not only contains known attacks but also, to some extent, reflects the characteristics of unknown, zero-day attacks [4].

Existing machine learning–based solutions [3, 5] face several key limitations:

Firstly, these models often exhibit a high false positive rate [3], [5] when dealing with a wide array of attacks; secondly, the models lack generalisability, as current studies typically use only a single dataset to report performance; thirdly, the models evaluated so far have not been exposed to the immense scale of modern network traffic; and finally, these solutions are required to accommodate the rapidly increasing volume, velocity, and dynamism of today's high-speed networks [3].

This paper introduces a deep learning model designed for accurate and efficient phishing detection using website URLs. In contrast to prior work, this study also evaluates the feasibility of deploying the proposed model on resource-constrained devices. This work also employs a substantial volume of legitimate and malicious URLs to construct the training set and assess the effectiveness of the proposed model [6]. The paper makes the following contributions:

The paper proposes novel hybrid approaches for runtime, dynamic phishing detection across all applications.

- We present a robust and scalable pre-processing method, a multi-stage data pre-processing pipeline specifically engineered for URL analysis and open set recognition. This pipeline involves:
- Tokenise raw URLs to capture subtle adversarial lexical patterns;
- Padding and truncation for sequence uniformity, required for the
- A data for embedding, ensuring an optimally structured feature space for the subsequent autoencoder against zero-day attacks across a large-scale, high-velocity dataset.

The paper is organised as follows. Section 2 is about related work. Section 3 is about the proposed architecture. Section Methods. Section 5 evaluation.

2 Related Work

It has been widely investigated in the literature via traditional machine learning and, more recently, deep learning approaches. Most earlier research focused on blacklist-based, heuristic, and feature-engineered methods that rely on manually extracted URL features, such as domain length, suspicious characters, and WHOIS information. While these approaches achieved moderate accuracy, they often failed against zero-day phishing attacks and dynamic obfuscation techniques.

A CNN requires a large labelled dataset for training and has wide applications in image classification, phishing URL detection, and related domains. Its core modules are convolutional, max-pooling, and fully connected layers, which are primarily used for feature extraction and identifying important patterns in the data [1]. LSTM is a variant of RNN that learns dependencies in sequential data and determines whether the input is phishing. It comprises three gates: input, forget, and output; and with the addition of an autoencoder, it can leverage both known attacks and anomaly-based detection.

A real-time web page detection system uses deep learning to check URLs against blacklists and whitelists

[2]; one-dimensional CNNs have been applied for phishing URL detection using PhishTank and Alexa datasets [3]; hybrid DNN frameworks monitor network and host activity for known and unseen cyberattacks [4]; CNN-based approaches utilize network and host traffic for NIDS development [5]; GRU-RNNs detect complex patterns in intrusion detection systems [6]; lightweight CNN-based models extract data features for phishing URL detection [7]; DOC++ and DID frameworks enhance novelty-based detection of content attacks [8]; CNN models distinguish phishing from legitimate websites [9]; Deep Belief Networks (DBN) use web and interaction features for detection [10]; PhishHaven detects AI-generated and human-crafted URLs using lexical analysis [3]; systematic literature reviews (SLR) analyze current challenges in deep-learning-based phishing detection [11]; CNNs applied to large datasets like MUPD improve phishing URL detection [5]; surveys review deep learning algorithms for IDS [6]; hybrid models combining IRNN, LSTM, and CNN architectures identify malicious URLs [7]; LSTM-DNN hybrids use character embeddings and NLP features [12]; URLNet learns nonlinear URL embeddings and word representations [13]; URLDeepDetect analyzes semantic and lexical features for time-of-click phishing detection [14]; SVM, LSTM, and NLP algorithms detect email phishing attacks in cloud environments [15]; Siamese networks integrate twin LSTM sub-networks for robust URL feature representation [16]; convolutional recurrent neural networks (CRNN) build hybrid IDS for network-level detection [17].

literature reviews cover AI techniques including ML, DL, and hybrid approaches [18]; DURLD encodes raw URLs using character-level embeddings [19]; neural network-based systems leverage pattern recognition to detect subtle phishing indicators [20]; preprocessing, feature extraction, optimal selection, and classification frameworks improve detection efficiency using datasets like PhishStorm [21]; GANs generate URL-based phishing examples [22]; ResNet50 feature extraction combined with SVM classification distinguishes legitimate and phishing websites [23]; and ML-based approaches trained on large datasets identify distinguishing characteristics of phishing URLs, protecting sectors like healthcare [24].

Overall, the integration of CNN, LSTM, and autoencoder methods enhances phishing URL detection for both known and zero-day attacks, demonstrating the effectiveness of hybrid deep learning frameworks [25].

In this work [26], a new intelligent method for phishing prediction using a processing pipeline is proposed, employing machine learning and deep learning to predict phishing websites with high accuracy. The model is developed using four machine learning algorithms—Decision Tree, Naive Bayes, Support Vector Machine (SVM), and Random Forest—and CNN as the deep learning component. Research works [27–35] analysed adaptive phishing attacks to understand runtime threats, though these works were conducted on fixed devices. Various works [36–46] proposed distributed nodes and security algorithms to mitigate runtime security issues, identified runtime security methods such as encryption and flags, and studied runtime vulnerabilities in use cases related to healthcare and transportation.

Recent advances in large language models have significantly improved the state of the art in phishing detection, particularly in URL analysis and multi-agent email classification systems. Cohen [47] presents a new client-side security analysis paradigm by demonstrating zero-shot LLM inference directly in the browser. This work shows that lightweight in-browser LLMs can analyze URLs for malicious patterns while offering greater privacy, lower latency, and real-time protection on resource-constrained devices. Complementing this, Xue et al. [48] present Multi-Phish-Guard, an LLM-based multi-agent framework for complex phishing email detection. Their system leverages coordinated agents that specialise in content interpretation, URL verification, and behavioural pattern analysis, thereby increasing detection accuracy. This work demonstrates that multi-agent reasoning and distributed model components outperform single-model baselines, particularly for sophisticated, unseen phishing attacks. Taken together, these works indicate a growing trend toward the fusion of zero-shot LLM reasoning, client-side deployment, and multi-agent collaboration to develop improved phishing defences

that provide scalable and generalisable solutions.

To the best of our knowledge, zero-day attacks on phishing URLs are widely ignored; this paper addresses this issue and proposes hybrid phishing approaches for phishing URLs.

3 Proposed Architecture

The proposed architecture consists of multiple components, as illustrated in Figure 1.

The first component involves raw URL data, which may appear in multiple formats, including HTML. The raw processing stage preprocesses input data such as URL strings or network traffic sequences. The data are first processed using a CNN with ReLU activation. The CNN extracts local patterns, such as character-level or token-level N-grams, that frequently appear in phishing URLs. ReLU (Rectified Linear Unit) introduces nonlinearity by mapping negative inputs to zero and leaving positive inputs unchanged, thereby enabling the model to learn complex relationships. The LSTM encoder then processes these extracted features sequentially to capture contextual and temporal dependencies. The combined output of the convolutional, ReLU, and LSTM layers forms a latent representation—a compressed vector that captures normal behavioural patterns learned from benign training data. The first part of the network (the CNN–LSTM encoder) maps the raw input into a latent feature representation.

The second part of the network, the decoder, reconstructs the input data from the latent representation. The decoder employs upsampling, convolutional layers, and ReLU activation to expand the latent features and reconstruct the sequential structure. The decoder's LSTM layers reconstruct the sequential order of the uses, both ML and DL-based approaches, and apply them in the cloud to detect email attack anomalies. In data, allowing it to learn an identity function with a bottleneck (latent representation). For benign inputs, the model reconstructs the sequence with low reconstruction error. For anomalous or phishing inputs, the model encounters previously unseen patterns, leading to poor reconstruction and a high reconstruction error. These higher error values are interpreted as potential phishing attempts or anomalies.

Overall, the architecture integrates CNN, LSTM, and autoencoder components to detect both known and zero-day phishing attacks through feature extraction, sequential modeling, and anomaly detection.

4 Mathematical Model for Phishing Detection

Let the input be a raw sequence X, which can represent a URL, HTML token sequence, or network traffic. Our goal is to identify known and unknown phishing instances.

4.1 Step 1: CNN Feature Extraction

The CNN module extracts hierarchical features from the input sequence:

$$X = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathcal{V} \tag{1}$$

$$E = \text{Embed}(X) \in \mathbb{R}^{n \times d} \quad \text{(embedding matrix)} \tag{2}$$

$$C_k = \sigma\Big(\text{Norm}(\text{Conv1D}(E; k, f_k))\Big), \quad k \in \mathcal{K} \tag{3}$$

$$P_k = \text{Pool}(C_k) \quad \text{(max or average pooling)} \tag{4}$$

$$F_{\text{raw}} = \text{Concat}(P_{k_1}, \ldots, P_{k_m}) \tag{5}$$

$$F = \text{Align}(F_{\text{raw}}, T) = \{f_t\}_{t=1}^{T} \tag{6}$$

where:

- $\mathcal{K} = \{k_1, \ldots, k_m\}$ are kernel sizes, $\mathcal{F} = \{f_1, \ldots, f_m\}$ are filter counts.
- $\sigma(\cdot) = \text{ReLU}(\cdot)$ is the activation.
- $\text{Norm}(\cdot)$ denotes batch or layer normalization.
- $\text{Align}(\cdot)$ optionally upsamples or interpolates features to length T.

4.2 Step 2: LSTM Sequence Encoding with Adaptive Contrastive Loss

Each feature sequence F is encoded using an LSTM, projected into a latent embedding space, and optimized using a contrastive loss:

$$h_i = \text{LSTM}_{\theta_{\text{enc}}}(F_i) \tag{7}$$

$$c_i = \text{FC}(h_i) \tag{8}$$

$$z_i = g_\phi(c_i), \quad z_i \in \mathbb{R}^{d_z} \tag{9}$$

$$\hat{z}_i = \frac{z_i}{\|z_i\|_2} \quad \text{(unit-normalized embedding)} \tag{10}$$

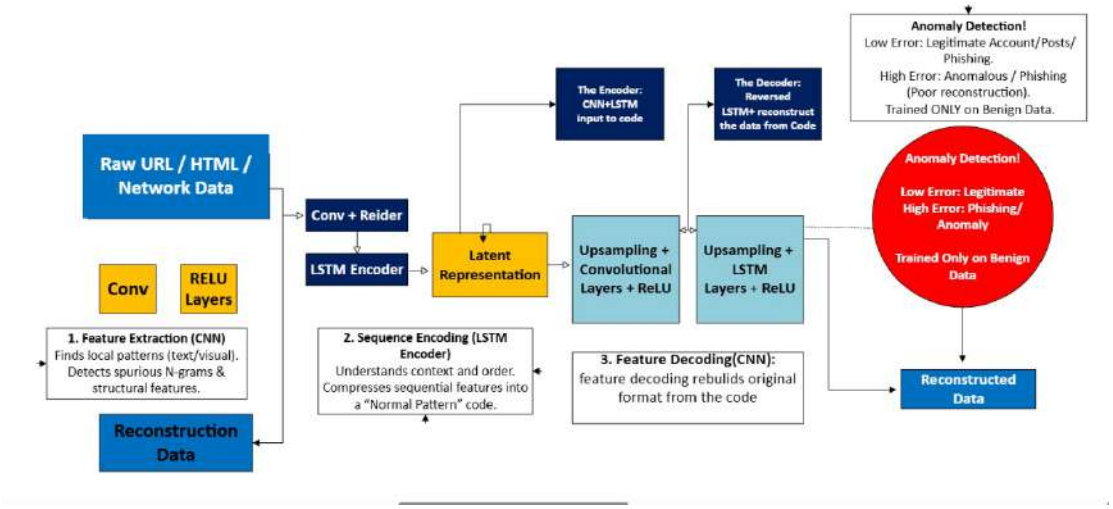

Figure 1. A Novel Deep Learning-Enabled Phishing Attack Detection Architecture

For a mini-batch of pairs $\mathcal{B} = \{(F_i, F_j, y_{ij})\}$, define cosine similarity:

$$s_{ij} = z_i^\top z_j \tag{11}$$

Adaptive margin contrastive loss:

$$\mathcal{L}_{AM} = \frac{1}{|\mathcal{B}|} \sum_{(i,j)\in\mathcal{B}} \begin{cases} 1 - s_{ij}, & y_{ij} = 1 \\ \max(0, s_{ij} - (\alpha + \delta(1 - s_{ij}))), & y_{ij} = 0 \end{cases} \tag{12}$$

4.3 Step 3: Autoencoder-Based Anomaly Detection

The autoencoder learns to reconstruct benign features F_b:

$$z_b = \mathcal{E}(F_b) \tag{13}$$

$$\hat{F}_b = \mathcal{D}(z_b) \tag{14}$$

$$\hat{X}_b = \text{ReconstructToInputFormat}(\hat{F}_b) \tag{15}$$

$$\mathcal{L}_{\text{rec}} = \frac{1}{B} \sum_{b=1}^{B} \ell_{\text{rec}}(X_b, \hat{X}_b) \tag{16}$$

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{R}(\theta_E, \theta_D) \tag{17}$$

Where ℓ_{rec} is typically mean squared error (MSE) or cross-entropy, and $\mathcal{R}(\theta)$ is a regularization term.

4.4 Step 4: Anomaly Detection Criterion

Define the reconstruction error for a test sample X_{test}:

$$F = \text{CNN\_Feature\_Extractor}(X_{\text{test}}) \tag{18}$$

$$z = \mathcal{E}(F) \tag{19}$$

$$\hat{F} = \mathcal{D}(z) \tag{20}$$

$$\hat{X}_{\text{test}} = \text{ReconstructToInputFormat}(\hat{F}) \tag{21}$$

$$e_{\text{test}} = \ell_{\text{rec}}(X_{\text{test}}, \hat{X}_{\text{test}}) \tag{22}$$

Decision function:

$$\mathcal{A}(X_{\text{test}}) = \begin{cases} \text{ANOMALY / PHISHING}, & e_{\text{test}} > \tau_{\text{err}} \\ \text{BENIGN / LEGITIMATE}, & e_{\text{test}} \leq \tau_{\text{err}} \end{cases} \tag{23}$$

The threshold τ_{err} is computed from validation data, e.g., the p-th percentile of benign reconstruction errors.

4.5 Step 5: Integrated Objective

The total learning objective combines the LSTM contrastive loss and the autoencoder reconstruction loss:

$$\min_{\theta_{\text{enc}}, \phi, \theta_E, \theta_D} \mathcal{L}_{\text{total}} = \mathcal{L}_{AM} + \beta \mathcal{L}_{\text{rec}} \tag{24}$$

where β balances embedding similarity and reconstruction fidelity.

This unified mathematical model captures feature extraction, representation learning, and anomaly detection for identifying known and unknown phishing URLs.

5 Proposed Method

The proposed method, named the Zero-Phishing Convolutional Neural Network and Long Short-Term Memory (ZP-CNN-LSTM) algorithm, consists of three distinct schemes. ZP-CNN-LSTM has many methods for dataset usage and processing. The proposed Zero-Phishing Convolutional Neural Network and Long Short-Term Memory (ZP-CNN-LSTM) method is a hybrid deep-learning framework designed to detect both known and unknown phishing attacks from raw sequential data, such as URLs, HTML tokens, or network traffic streams.

Algorithm 1. CNN Feature Extraction (Conv + ReLU + Pooling)

Require: Raw input sequence X (URL/HTML tokens/network sequence)
Require: Kernel sizes $\mathcal{K} = \{k_1, \ldots, k_m\}$ and filter counts $\mathcal{F} = \{f_1, \ldots, f_m\}$
Require: Pooling strategy (max/avg), activation $\sigma(\cdot)$ = ReLU, normalization method
Ensure: Feature map sequence $F = \{f_t\}_{t=1}^{T}$ (time / token aligned)
Step 1: Preprocess input
Tokenize input: $X \to \{x_1, \ldots, x_n\}$
Embed tokens: produce embedding matrix $E \in \mathbb{R}^{n \times d}$
Optionally: apply input normalization/dropout
Step 2: Multi-kernel convolutional banks
for all kernel size $k \in \mathcal{K}$ **do**
    $C_k \leftarrow \text{Conv1D}(E;\ \text{kernel} = k,\ \text{filters} = f_k)$
    $C_k \leftarrow \sigma(C_k)$ ▷ apply ReLU
    $C_k \leftarrow \text{Normalize}(C_k)$ ▷ batch-norm / layer-norm
    $P_k \leftarrow \text{Pool}(C_k)$ ▷ max or average pooling
end for
Step 3: Concatenate and align
$F_{\text{raw}} \leftarrow \text{Concat}(P_{k_1}, \ldots, P_{k_m})$
If needed, upsample/interpolate F_{raw} to length T to obtain F
Step 4: Post-process
Optionally apply: dropout, residual connections, final normalization
return F

The CNN-based feature extraction procedure is formally described in Algorithm 1. The CNN Feature Extraction process takes as input a raw sequence X (such as a URL string, HTML token sequence, or network traffic data) along with a set of convolutional kernel sizes $\mathcal{K} = \{k_1, \ldots, k_m\}$, filter counts $\mathcal{F} = \{f_1, \ldots, f_m\}$, a chosen pooling strategy (max or average), an activation function $\sigma(\cdot)$ =ReLU, and a normalization method. First, the raw input X is preprocessed by tokenizing or byte-encoding it into a sequence of discrete units $\{x_1, \ldots, x_n\}$, which are then embedded into continuous vectors, forming an embedding matrix $E \in \mathbb{R}^{n \times d}$ through either learned embeddings or one-hot encoding. Next, for each kernel size $k \in \mathcal{K}$, a one-dimensional convolution operation Conv1D is applied on E using f_k filters, generating convolutional feature maps C_k, which are then passed through the ReLU activation $\sigma(C_k)$ to introduce non-linearity. These activated outputs are subsequently normalized and pooled using the selected pooling method, yielding pooled representations P_k. The pooled outputs from all kernels are concatenated—optionally upsampled or interpolated if necessary—to yield a time- or token-aligned feature sequence $F = \{f_t\}_{t=1}^{T}$. Finally, post-processing steps such as dropout regularization, batch normalization, or residual connections may be applied to enhance robustness and stability before returning the final feature map sequence F.

The sequence encoding process with LSTM and contrastive embedding loss begins by taking the feature sequence $F = \{f_t\}_{t=1}^{T}$ obtained from Algorithm 2 as input, along with a mini-batch of paired or augmented samples $\mathcal{B} = \{(F_i, F_j, y_{ij})\}$ where each label $y_{ij} \in \{0, 1\}$ indicates whether the pair is similar or dissimilar. The model consists of an LSTM encoder parameterized by θ_{enc}, followed by a projection head $g_\phi(\cdot)$, and is trained using contrastive loss with a base margin α, scaling factor δ, and temperature τ. Each sequence F_i in the batch is passed through the LSTM encoder $\text{LSTMEncoder}_{\theta_{\text{enc}}}$ to obtain a hidden representation h_i, which may correspond to the final hidden state or an attentive pooling of the LSTM outputs. This hidden state is optionally projected through a fully connected layer $c_i = \text{FC}(h_i)$ to form a bottleneck repre-

Algorithm 2. Sequence Encoding with LSTM and Adaptive Contrastive Embedding Loss

Require: Feature sequence $F = \{f_t\}_{t=1}^{T}$
Require: Mini-batch of pairs $\mathcal{B} = \{(F_i, F_j, y_{ij})\}$ with $y_{ij} \in \{0, 1\}$
Require: LSTM encoder parameters θ_{enc}, projection head $g_\phi(\cdot)$
Require: Margin α, scaling factor δ, temperature τ
Ensure: Embeddings z_i and contrastive loss $\mathcal{L}_{AM}$
Step 1: Encode each sequence
for all sequence F_i in batch $\mathcal{B}$ **do**
  $h_i \leftarrow \text{LSTMEncoder}_{\theta_{\text{enc}}}(F_i)$ ▷ final hidden state or attentive pooling
  $c_i \leftarrow \text{FC}(h_i)$ ▷ optional latent projection
  $z_i \leftarrow g_\phi(c_i)$
  $z_i \leftarrow z_i / \|z_i\|_2$ ▷ normalize to unit sphere
end for
Step 2: Compute adaptive contrastive loss
$\mathcal{L}_{AM} \leftarrow 0$
for all pairs (i, j, y_{ij}) in $\mathcal{B}$ **do**
  $s_{ij} \leftarrow z_i^\top z_j$ ▷ cosine similarity
  if $y_{ij} = 1$ **then**
    $\mathcal{L}_{ij}^{+} \leftarrow 1 - s_{ij}$
    $\mathcal{L}_{AM} \leftarrow \mathcal{L}_{AM} + \mathcal{L}_{ij}^{+}$
  else
    $m_{ij} \leftarrow \alpha + \delta \cdot (1 - s_{ij})$ ▷ adaptive negative margin
    $\mathcal{L}_{ij}^{-} \leftarrow \max(0,\ s_{ij} - m_{ij})$
    $\mathcal{L}_{AM} \leftarrow \mathcal{L}_{AM} + \mathcal{L}_{ij}^{-}$
  end if
end for
Step 3: Normalize loss and apply optional regularization
$\mathcal{L}_{AM} \leftarrow \mathcal{L}_{AM} / |\mathcal{B}|$
Optional: add embedding regularization (e.g., $\|z_i\|^2$)
Step 4: Update model parameters
Update θ_{enc} and ϕ via backpropagation on $\mathcal{L}_{AM}$
return embeddings $\{z_i\}$ and loss $\mathcal{L}_{AM}$

sentation, followed by a projection head $z_i = g_\phi(c_i)$ that produces the final normalized embedding, typically using ℓ_2 normalization such that $z_i \leftarrow z_i / \|z_i\|_2$. Once all embeddings are obtained, the contrastive adaptive margin loss $\mathcal{L}_{AM}$ is computed across all pairs (i, j, y_{ij}) in the batch. For each pair, the cosine similarity $s_{ij} = z_i^\top z_j$ is evaluated; if the pair is positive ($y_{ij} = 1$), the loss term encourages closeness using $\mathcal{L}_{ij}^{+} = 1 - s_{ij}$, while for negative pairs ($y_{ij} = 0$), a dynamic margin $m_{ij} = \alpha + \delta \cdot (1 - s_{ij})$ is applied and the loss is computed as $\mathcal{L}_{ij}^{-} = \max(0, s_{ij} - m_{ij})$ to push dissimilar samples apart. The total contrastive loss $\mathcal{L}_{AM}$ is accumulated over the batch and normalized by its size, i.e., $\mathcal{L}_{AM} \leftarrow \mathcal{L}_{AM} / |\mathcal{B}|$, with optional regularization terms such as embedding norm penalties $||z||^2$. Finally, the model parameters θ_{enc} and ϕ are updated through backpropagation using the computed loss $\mathcal{L}_{AM}$, yielding optimized embeddings $\{z_i\}$ that preserve semantic similarity in the latent space.

The Autoencoder Training and Anomaly Detection pprocedure is formally summarized in Algorithm 3. The process begins with a training dataset of benign samples, $\mathcal{D}_{\text{benign}}$, which may consist of feature vectors or sequential data. The architecture comprises an encoder $\mathcal{E}(\cdot)$, typically combining convolutional and LSTM layers, and a decoder $\mathcal{D}(\cdot)$ that reconstructs the input through upsampling, convolutional, and recurrent operations, parameterized by θ_E and θ_D, respectively. Training is performed solely on benign data for N epochs, with a batch size of B. In each iteration, the benign dataset $\mathcal{D}_{\text{benign}}$ is shuffled, and for each batch $\{X_b\}_{b=1}^{B}$, features are first extracted via a CNN-based module (Algorithm 1), resulting in representations F_b. These features are encoded into latent embeddings $z_b = \mathcal{E}(F_b)$ through the LSTM encoder and subsequently decoded so as $\hat{F}_b = \mathcal{D}(z_b)$ to reconstruct the original feature sequence. Optionally, the reconstructed representation $\hat{F}_b$ may be mapped back to the raw input domain, producing $\hat{X}_b = \text{ReconstructToInputFormat}(\hat{F}_b)$. The reconstruction loss $\mathcal{L}_{\text{rec}} = \frac{1}{B} \sum_b \ell_{\text{rec}}(X_b, \hat{X}_b)$, computed via mean squared error, cross-entropy, or token-level loss, is combined with a regularization term weighted by λ, yielding the total loss $\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{R}(\theta)$. Parameters θ_E

Algorithm 3. Autoencoder Training and Anomaly Detection by Reconstruction Error

Require: Benign training set $\mathcal{D}_{\text{benign}}$
Require: Encoder $\mathcal{E}(\cdot)$ and decoder $\mathcal{D}(\cdot)$ with parameters θ_E, θ_D
Require: Loss weight λ, anomaly threshold τ_{err} (computed later)
Require: Batch size B, number of epochs N
Ensure: Trained parameters (θ_E, θ_D), threshold τ_{err}, detector function $\mathcal{A}(\cdot)$
Step 1: Train autoencoder on benign data
for $e = 1$ to N **do**
  Shuffle $\mathcal{D}_{\text{benign}}$
  for each batch $\{X_b\}_{b=1}^{B}$ **do**
    $F_b \leftarrow \text{CNN\_Feature\_Extractor}(X_b)$
    $z_b \leftarrow \mathcal{E}(F_b)$
    $\hat{F}_b \leftarrow \mathcal{D}(z_b)$
    $\hat{X}_b \leftarrow \text{ReconstructToInputFormat}(\hat{F}_b)$
    $\mathcal{L}_{\text{rec}} \leftarrow \frac{1}{B} \sum_b \ell_{\text{rec}}(X_b, \hat{X}_b)$
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{rec}} + \lambda \cdot \mathcal{R}(\theta)$
    Update θ_E, θ_D using optimizer on $\mathcal{L}$
  end for
end for
Step 2: Compute reconstruction error threshold
Compute validation errors $\mathcal{E}_{\text{val}} = \{e_v = \ell_{\text{rec}}(X_v, \hat{X}_v)\}$
$\tau_{\text{err}} \leftarrow \text{Percentile}(\mathcal{E}_{\text{val}}, p)$ ▷ e.g., 95th percentile
Step 3: Define anomaly detector $\mathcal{A}(\cdot)$
function $\mathcal{A}(X_{\text{test}})$
  $F \leftarrow \text{CNN\_Feature\_Extractor}(X_{\text{test}})$
  $z \leftarrow \mathcal{E}(F)$
  $\hat{F} \leftarrow \mathcal{D}(z)$
  $\hat{X}_{\text{test}} \leftarrow \text{ReconstructToInputFormat}(\hat{F})$
  $e_{\text{test}} \leftarrow \ell_{\text{rec}}(X_{\text{test}}, \hat{X}_{\text{test}})$
  if $e_{\text{test}} > \tau_{\text{err}}$ **then**
    return ANOMALY / PHISHING
  else
    return BENIGN / LEGITIMATE
  end if
end function
return (θ_E, θ_D), τ_{err}, $\mathcal{A}(\cdot)$

and θ_D are optimized using backpropagation. After training, reconstruction errors $\mathcal{E}_{\text{val}} = \{e_v\}$ are collected from a validation subset of benign samples, where each $e_v = \ell_{\text{rec}}(X_v, \hat{X}_v)$, and a detection threshold τ_{err} is determined statistically, often as the p-th percentile (e.g., 95th) of the error distribution. During inference, the anomaly decision function $\mathcal{A}(X_{\text{test}})$ operates by extracting CNN features F from an input X_{test}, encoding and decoding them via $\mathcal{E}$ and $\mathcal{D}$ to reconstruct $\hat{X}_{\text{test}}$, and computing the reconstruction error $e_{\text{test}} = \ell_{\text{rec}}(X_{\text{test}}, \hat{X}_{\text{test}})$. If $e_{\text{test}} > \tau_{\text{err}}$, the input is classified as **anomalous or phishing**, whereas if $e_{\text{test}} \leq \tau_{\text{err}}$, it is classified as **benign or legitimate**.

Optionally, the reconstruction-based anomaly score can be combined with embedding-based or contrastive similarity scores using a weighted fusion scheme to improve robustness to adversarial or ambiguous cases. The final outputs of this process include the trained autoencoder parameters (θ_E, θ_D), the computed anomaly threshold τ_{err}, and the detector function $\mathcal{A}(\cdot)$ for real-time or batch anomaly detection.

Time and Space Complexity:

The overall complexity of the proposed ZP-CNN-LSTM algorithm can be described as follows:

- **Time Complexity:** $O(n \cdot m \cdot k)$, where n is the number of URLs, m is the sequence length of each URL, and k is the number of layers in the CNN-LSTM-autoencoder network. The CNN layers contribute $O(m \cdot f \cdot k_{cnn})$, LSTM layers contribute $O(m \cdot h \cdot k_{lstm})$, and the autoencoder reconstruction contributes $O(m \cdot k_{ae})$ to the total runtime.
- **Space Complexity:** $O(n \cdot d + p)$, where d is the feature dimension of the latent representation, and p is the total number of trainable parameters in the CNN, LSTM, and autoencoder network.

6 Performance Evaluation

The performance of the proposed hybrid CNN-LSTM model is evaluated on a large-scale dataset of URLs curated specifically for this work. This dataset was designed with issues of model generalization ability and zero-day detection in mind, explicitly including a large, diverse dataset of benign and malicious traffic.

Phishing URLs: The malicious URLs were collected from live and historical phishing feeds to identify evolving attack patterns. Sources included public threat intelligence platforms such as PhishTank and research papers on phishing attack detection, which often publish dumps of recent high-profile attacks. This was crucial for collecting novel or recent samples representative of emerging and zero-day threats.

6.0.1 Dataset Statistics and Details

Table 1. Distribution and Composition of the Curated URL Dataset

Category	Samples	Percentage (%)
Total URLs	2,500,000	100
Benign URLs	1,500,000	60
Phishing URLs	1,000,000	40
Split: Training	2,000,000	80
Split: Testing/Validation	500,000	20

In this paper, we developed a dataset and applied deep learning techniques to detect phishing attacks, including zero-day cases. The dataset was manually created by collecting a total of ten thousand website and URL samples. These included both phishing and legitimate sites, labelled as "1" for phishing and "0" for benign websites. To make the data more representative, approximately 60% were global websites, while 40% were from local Pakistani sources. This helped the model learn patterns that occur locally, internationally, or globally, consistent with studies on detection [31].

The collection process for data therefore consisted of scraping URLs from open repositories like PhishTank and other publicly available datasets. At the same time, the list of legitimate sites was manually verified using trusted sources. During labeling, each entry was vetted for correctness, ensuring the phishing samples were active and matched known attack types. We included a range of features to describe the URLs and website content. We used simple lexical features such as the number of dots or special symbols in the link, and also whether the link used HTTPS. When possible, we also examined webpage structures, such as the number of input forms or embedded scripts, since these details often reveal hidden phishing behaviour.

We performed multiple cleaning and preparation steps after data collection. Duplicates and broken links were removed, and the remaining data was reviewed and manually compiled from samples of similar scale to ensure fairness in learning across different features. According to [31], this kind of preprocessing is usually considered critical for deep learning-based approaches to phishing detection. After cleaning, the dataset was split into training and test sets in order to assess how well the model performs on unseen data. We also performed several runs to confirm that the results were stable and did not depend on the order of the data.

Our model design brought together several deep learning layers that captured both the visible patterns and the hidden structure of phishing URLs. The system comprises components that process token sequences and patterns in URLs, while another element learns general webpage behavior. For zero-day phishing, we further trained an additional recognizer for unusual or unseen patterns. The idea of identifying unknown threats through learned reconstruction patterns has been successfully applied in other intrusion detection work. By comparing how well the model reconstructed normal data versus suspicious data, we were able to identify anomalies that were likely to represent new phishing techniques. The whole pipeline—from data collection through labeling and cleaning to deep learning model training—was designed to make the detection system practical and reliable.

6.1 Result Analysis

The table presents the phishing detection accuracy of four models—ZP-CNN-LSTM, CNN, RNN, and Autoencoder—across four URL categories: Website URLs, Email URLs, Shortened URLs, and Social Media URLs, evaluated on datasets of 1,000, 2,000, 5,000, and 10,000 URLs. For each model, detection performance is reported separately for known phishing and zero-day phishing URLs. Across all categories and dataset sizes, ZP-CNN-LSTM consistently achieves the highest accuracy, ranging from approximately 0.935 to 0.968

Table 2. Phishing Detection Accuracy Across URL Categories for Different Models

2*URL Category	2*Model	Known Phishing Accuracy				Zero Phishing Accuracy			
		1k URLs	2k URLs	5k URLs	10k URLs	1k URLs	2k URLs	5k URLs	10k URLs
4*Website URLs	ZP-CNN-LSTM	0.936	0.943	0.955	0.965	0.912	0.919	0.931	0.941
	CNN	0.776	0.782	0.791	0.799	0.515	0.520	0.529	0.537
	RNN	0.779	0.783	0.791	0.798	0.569	0.573	0.581	0.588
	Autoencoder	0.797	0.800	0.806	0.811	0.615	0.618	0.624	0.629
4*Email URLs	ZP-CNN-LSTM	0.938	0.945	0.957	0.968	0.909	0.916	0.928	0.939
	CNN	0.769	0.775	0.784	0.791	0.519	0.524	0.533	0.541
	RNN	0.786	0.790	0.798	0.804	0.570	0.574	0.582	0.588
	Autoencoder	0.798	0.801	0.807	0.813	0.608	0.611	0.617	0.623
4*Shortened URLs	ZP-CNN-LSTM	0.935	0.942	0.954	0.965	0.907	0.914	0.926	0.937
	CNN	0.773	0.779	0.788	0.796	0.512	0.517	0.526	0.534
	RNN	0.784	0.788	0.796	0.802	0.569	0.573	0.581	0.587
	Autoencoder	0.799	0.802	0.808	0.814	0.610	0.613	0.619	0.624
4*Social Media URLs	ZP-CNN-LSTM	0.937	0.944	0.956	0.966	0.910	0.917	0.929	0.940
	CNN	0.771	0.776	0.785	0.793	0.514	0.519	0.528	0.536
	RNN	0.783	0.788	0.795	0.802	0.568	0.572	0.580	0.586
	Autoencoder	0.802	0.806	0.812	0.817	0.608	0.611	0.617	0.623

for known phishing and 0.907 to 0.941 for zero-day phishing, outperforming CNN, RNN, and Autoencoder. The CNN and RNN models show moderate accuracy, while the autoencoder exhibits the lowest detection performance. The results demonstrate that ZP-CNN-LSTM is highly effective at identifying both known and previously unseen phishing attacks across different URL types and data scales.

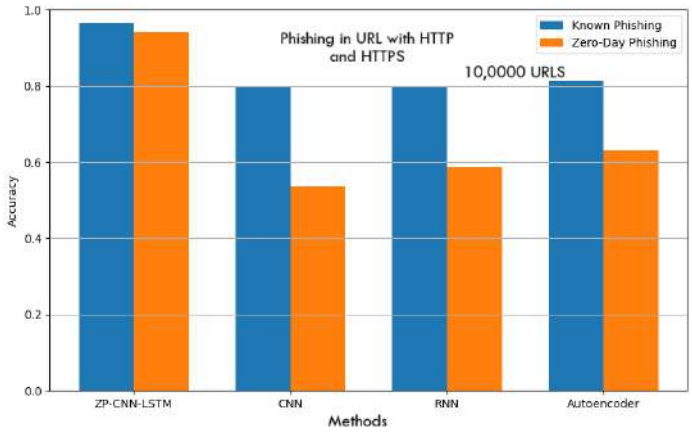

Figure 2. Known and Unknown Zero-Day Attack Detection in HTTP and HTTPS Efficient URLS

Figure 2 shows the "Phishing in URL with HTTP and HTTPS," outlining the foundational elements of a study focused on detecting phishing websites. The research is based on an analysis of a substantial dataset comprising 100,000 unique URLs. This dataset is divided into two categories to assess the robustness of the detection methods: "Known Phishing," which includes previously identified and cataloged phishing threats, and "Zero-Day Phishing," which comprises novel, previously unseen attacks not present in existing security databases. The primary metric for evaluating performance in this study is "Accuracy." The figure compares the effectiveness of several advanced machine learning models in tackling this problem, specifically naming ZP-CNN-LSTM, CNN, RNN, and an autoencoder. The inclusion of these models suggests a comparative analysis where the hybrid ZP-CNN-LSTM model is likely positioned as a proposed or benchmark method against other established architectures to determine which is most proficient at accurately identifying both known and emergent phishing URLs.

Figure 3 scopes the investigation beyond just website URLs, as indicated by its title, "Email and Random URLs 20,000". This suggests that the dataset has been scaled to 200,000 samples and now includes a mix of URLs extracted from emails alongside random web

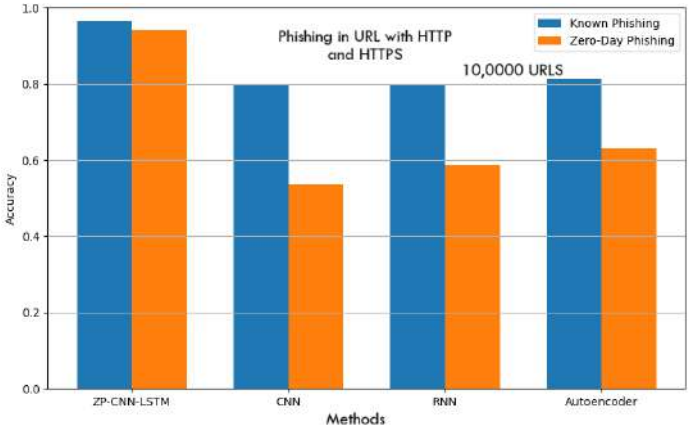

Figure 3. Known and Unknown Zero-Day Attack Detection in Email and Random URLS

URLs, providing a more diverse and realistic testing environment that better mimics how phishing links are often distributed. The central evaluation metric remains "Accuracy". The models under scrutiny are listed, with "ZP-CNN-LSTM" prominently featured at the top, followed by "CNN", "Methods", "RNN", and "Autoencoder". The layout follows a similar comparative structure to the first figure, aiming to demonstrate which algorithmic approach—particularly the highlighted ZP-CNN-LSTM—achieves the highest detection accuracy when applied to this larger, more varied pool of data, which includes the common attack vector of email-based phishing attempts.

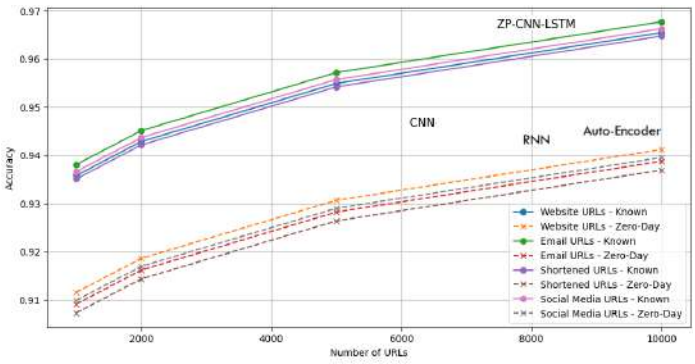

Figure 4. Performance Evaluation of All Algorithms

Figure 4 presents performance results. The y-axis represents "Accuracy" on a scale from 0.91 to 0.97, while the x-axis shows the "Number of URLs" tested, ranging up to 10,000. The performance of four models is demonstrated, such as ZP-CNN-LSTM, CNN, RNN, and Auto-Encoder. The graph clearly illustrates that the ZP-CNN-LSTM model consistently achieves the highest accuracy, peaking near 0.97, and generally outperforms the other models across the entire range of URL sample sizes. Furthermore, the results are broken down by eight distinct URL categories, demonstrating the models' performance on "Website URLs," "Email URLs," "Shortened URLs," and "Social Media URLs," with each category further split into "Known" and "Zero-Day" phishing types. This detailed visualization allows for a nuanced comparison, showing not only the overall superiority of the ZP-CNN-LSTM model but also its relative effectiveness across different types of web addresses and against both familiar and novel phishing threats.

6.2 Finding and Limitation

Findings: The study demonstrates that the proposed ZP-CNN-LSTM hybrid deep learning approach effectively detects both known and zero-day phishing URLs. Tested on a dataset of 2 million URLs, the system achieved an accuracy of 98%, outperforming existing phishing detection methods. The approach successfully combines static analysis, CNN-based signature detection, and LSTM-autoencoder-based zero-day detection, providing adaptive protection against evolving phishing attacks.

Limitations: Despite its high accuracy, the method may require significant computational resources for large-scale deployment due to the combined CNN, LSTM, and autoencoder architecture. Additionally, the system's performance may be affected by highly obfuscated or evolving phishing techniques that differ significantly from the training data, requiring continuous updates and retraining.

7 Conclusion and Future Work

In this study, we investigated the effectiveness of hybrid deep learning approaches for detecting both known and zero-day phishing attacks in URLs. The proposed intrusion detection system (IDS) successfully analyzed a testbed of 2 million URLs, distinguishing between legitimate and malicious URLs with a high accuracy of 98%. The system demonstrated robust

performance by integrating static analysis, signature-based detection, and heuristic techniques, effectively mitigating phishing threats in real-world scenarios. These results validated that the hybrid approach could provide adaptive and dynamic protection against emerging phishing attacks, outperforming conventional methods in practical application.

Future research will focus on enhancing the scalability and efficiency of the proposed IDS to handle larger and more diverse datasets in real time. We will explore integrating additional contextual features, such as user behavior and network traffic patterns, to improve zero-day phishing detection further. Moreover, we plan to implement online learning mechanisms so the system can continuously adapt to evolving phishing techniques and emerging cyber threats, ensuring comprehensive and proactive protection for users.

Dataset Statements

A phishing email and websiste URLs dataset is a structured, transparent, and carefully curated collection of URLs extracted from both malicious phishing emails and legitimate email messages, designed to support reproducible cybersecurity research, email filtering, and machine-learning-based detection systems, and in our study the dataset is publicly available at the following link to ensure openness and consistency: https://github.com/arlakhan/Phishing-URL-datasets/tree/main

This dataset includes the full URL, domain, protocol, path, lexical patterns, and structural characteristics of each link, along with essential email-level metadata such as sender details, subject lines, delivery context, and relevant header or body text, enabling deeper behavioural analysis of how phishing links are embedded within email communication. Each entry is clearly labelled as either a phishing URL or a legitimate URL, where legitimate samples are represented using zero-attack values that indicate completely benign behaviour with no spoofing indicators, redirection traps, social-engineering hooks, or malware-delivery patterns, thereby serving as the baseline against which malicious behaviours can be contrasted.

The inclusion of these zero-attack samples is critical, as modern machine-learning systems must develop an understanding of normal traffic patterns to reliably distinguish safe everyday email activity from real threats. To address the concerns of dataset transparency and reproducibility—specifically the absence of dataset links, inconsistent dataset sizes, and missing training details—the dataset has been standardised, fully documented, and made openly accessible, ensuring consistent benchmark usage and removing ambiguity in experimental setup. Using this dataset, we developed and evaluated three deep-learning architectures—Convolutional Neural Networks (CNN) for extracting spatial and lexical URL patterns, Long Short-Term Memory (LSTM) networks for modelling sequential URL behaviours and email-text flow, and an Auto-Encoder for anomaly-based detection capable of identifying unusual or zero-day phishing structures by learning the normal distribution of zero-attack URLs. This unified approach allows classification models and anomaly-detection systems to learn URL behaviour patterns, correlate temporal structures, recognise deceptive patterns, and detect emerging phishing techniques with improved accuracy.

Overall, this dataset therefore becomes a vital, reproducible asset for training, evaluating, and benchmarking next-generation phishing detection mechanisms while supporting academic research, strengthening email-gateway defences, and enhancing the robustness of cybersecurity solutions across modern digital communication environments.

Author Contributions

Nimra Shaikh: Conceptualization, Methodology. **Ambreen Hussain:** Software implementation. **Qurat-ul-ain Mastoi:** Data curation. **Asif Aziz Memon:** Investigation, Supervision, Software. **Atif Jamil:** Writing- Reviewing and Editing. **Abdullah Lakhan:** Supervision, Software Validation

Compliance with Ethical Standards

It is declared that all authors don't have any conflict of interest. It is also declare that this article does not

contain any studies with human participants or animals performed by any of the authors. Furthermore, informed consent was obtained from all individual participants included in the study.

References

[1] O. K. Sahingoz, E. Buber, and E. Kugu, "DEPHIDES: Deep learning based phishing detection system," *IEEE Access*, vol. 12, pp. 8052–8070, 2024, doi: 10.1109/ACCESS.2024.3352629.

[2] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, "Intelligent phishing detection scheme using deep learning algorithms," *J. Enterprise Inf. Manage.*, vol. 36, no. 3, pp. 747–766, Apr. 2023.

[3] R. Vinayakumar *et al.*, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[4] S. M. Sohi, J. P. Seifert, and F. Ganji, "RNNIDS: Enhancing network intrusion detection systems through deep learning," *Computers & Security*, vol. 102, p. 102151, 2021.

[5] T. A. Tang *et al.*, "Deep recurrent neural network for intrusion detection based networks," in *Proc. 4th IEEE Conf. Network Softwarization and Workshops (NetSoft)*, 2018, pp. 202–206.

[6] B. Wei *et al.*, "A deep-learning-driven light-weight phishing detection sensor," *Sensors*, vol. 19, no. 19, p. 4258, 2019.

[7] M. Soltani *et al.*, "An adaptable deep learning-based intrusion detection system to zero-day attacks," *J. Inf. Security Appl.*, vol. 76, p. 103516, 2023.

[8] Z. Alshingiti *et al.*, "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, p. 232, 2023.

[9] O. K. Sahingoz, E. Buber, and E. Kugu, "DEPHIDES: Deep learning based phishing detection system," *IEEE Access*, vol. 12, pp. 8052–8070, 2024.

[10] Q. E. U. Haq, M. H. Faheem, and I. Ahmad, "Detecting phishing URLs based on a deep learning approach to prevent cyber-attacks," *Applied Sciences*, vol. 14, no. 22, p. 10086, 2024.

[11] E. S. A. Alars and S. Kurnaz, "Enhancing network intrusion detection systems with combined network and host traffic features using deep learning," *Discover Computing*, vol. 27, no. 1, p. 39, 2024.

[12] E. A. Aldakheel *et al.*, "A deep learning-based innovative technique for phishing detection using uniform resource locators," *Sensors*, vol. 23, no. 9, p. 4403, 2023.

[13] M. Sameen, K. Han, and S. O. Hwang, "PhishHaven—An efficient real-time AI phishing URLs detection system," *IEEE Access*, vol. 8, pp. 83425–83443, 2020.

[14] N. Q. Do *et al.*, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, vol. 10, pp. 36429–36463, 2022.

[15] A. Al-Alyan and S. Al-Ahmadi, "Robust URL phishing detection based on deep learning," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 7, 2020.

[16] B. K. Sedraoui *et al.*, "Intrusion detection with deep learning: A literature review," in *Proc. 6th Int. Conf. Pattern Analysis and Intelligent Systems (PAIS)*, IEEE, 2024.

[17] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating deep learning approaches to characterize and classify malicious URLs," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1333–1343, 2018.

[18] A. Ozcan *et al.*, "A hybrid DNN–LSTM model for detecting phishing URLs," *Neural Comput. Appl.*, vol. 35, no. 7, pp. 4957–4973, 2023.

[19] H. Le *et al.*, "URLNet: Learning a URL representation with deep learning for malicious URL detection," arXiv:1802.03162, 2018.

[20] S. Afzal *et al.*, "Urldeepdetect: A deep learning approach for detecting malicious URLs using semantic vector models," *J. Netw. Syst. Manage.*, vol. 29, no. 3, p. 21, 2021.

[21] U. A. Butt *et al.*, "Cloud-based email phishing attack using machine and deep learning algorithms," *Complex Intell. Syst.*, vol. 9, no. 3, pp. 3043–3070, 2023.

[22] K. Sruthi, "A novel framework for effective phishing URL detection using an LSTM-based Siamese network," *Knowledge-Based Systems*, p. 114271, 2025.

[23] E. U. H. Qazi, M. H. Faheem, and T. Zia, "HDLNIDS: Hybrid deep-learning-based network intrusion detection system," *Applied Sciences*, vol. 13, no. 8, p. 4921, 2023.

[24] A. Basit *et al.*, "A comprehensive survey of AI-enabled phishing attacks detection techniques," *Telecommunication Systems*, vol. 76, no. 1, pp. 139–154, 2021.

[25] S. Srinivasan *et al.*, "DURLD: Malicious URL detection using deep learning-based character level representations," in *Malware Analysis Using Artificial Intelligence and Deep Learning*, Cham, Switzerland: Springer, 2020, pp. 535–554.

[26] I. A. Essien *et al.*, "Neural network-based phishing attack detection and prevention systems," *J. Frontiers Multidisciplinary Research*, vol. 2, no. 2, pp. 222–238, 2021.

[27] A. K. Yamarthy and C. Koteswararao, "MDepthNet based phishing attack detection using integrated deep learning methodologies for cyber security enhancement," *Cluster Computing*, vol. 27, no. 5, pp. 6377–6395, 2024.

[28] A. AlEroud and G. Karabatis, "Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks," in *Proc. 6th Int. Workshop Security and Privacy Analytics*, 2020.

[29] P. K. Roy, A. Kumar, and A. Singh, "Advanced learning for phishing URLs detection to secure consumer-centric applications," *IEEE Trans. Consumer Electron.*, vol. 70, no. 3, pp. 5756–5763, 2024.

[30] H. Bibi *et al.*, "Phishing website detection using improved multilayered convolutional neural networks," *J. Computer Science*, vol. 20, no. 9, pp. 1069–1079, 2024.

[31] M. Korkmaz, E. Kocyigit, O. Sahingoz, and B. Diri, "A hybrid phishing detection system using deep learning-based URL and content analysis," *Elektronika ir Elektrotechnika*, vol. 28, no. 5, pp. 1–8, 2022.

[32] N. Q. Do *et al.*, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, vol. 10, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.

[33] Z. Dai *et al.*, "An intrusion detection model to detect zero-day attacks in unseen data using machine learning," *PLOS ONE*, vol. 19, no. 9, e0308469, 2024, doi: 10.1371/journal.pone.0308469.

[34] "Systematic review of deep learning techniques for phishing email detection," *Electronics*, vol. 13, no. 19, p. 3823, 2024, doi: 10.3390/electronics13193823.

[35] M. F. Memon, R. Shah, and A. Ali, "A novel primary key infrastructure IoT-enabled secure access control framework for smart home applications," *VFAST Trans. Software Eng.*, vol. 13, no. 1, pp. 37–48, 2025.

[36] T. M. Grønli, H. Wu, M. Younas, and G. Ghinea, "A novel homomorphic blockchain scheme for intelligent transport services in fog/cloud and IoT networks," *IEEE Trans. Intell. Transp. Syst.*, 2024.

[37] Z. A. A. Alyasseri *et al.*, "Sustainable secure blockchain-assisted AIoT and green multi-constraints supply chain system," *IEEE Internet Things J.*, 2025.

[38] Q. Qazi, "Metaverse-assisted healthcare body sensor network architecture," in *Proc. IEEE 20th Int. Conf. Body Sensor Networks (BSN)*, Oct. 2024, pp. 1–4.

[39] M. A. Mohammed *et al.*, "Restricted Boltzmann machine assisted secure serverless edge system for internet of medical things," *IEEE J. Biomed. Health Inform.*, vol. 27, no. 2, pp. 673–683, 2022.

[40] T. M. Grønli, P. Bellavista, S. Memon, M. Alharby, and O. Thinnukool, "IoT workload offloading efficient intelligent transport system in federated ACNN integrated cooperated edge-cloud networks," *J. Cloud Comput.*, vol. 13, no. 1, p. 79, 2024.

[41] M. A. Mohammed *et al.*, "Augmented IoT cooperative vehicular framework based on distributed deep blockchain networks," *IEEE Internet Things J.*, vol. 11, no. 22, pp. 35825–35838, 2024.

[42] M. A. Mohammed *et al.*, "Secure blockchain-assisted Internet of Medical Things architecture for data fusion enabled cancer workflow," *Internet Things*, vol. 24, p. 100928, 2023.

[43] M. Elhoseny, M. A. Mohammed, and M. M. Jaber, "SFDWA: Secure and fault-tolerant aware delay optimal workload assignment schemes in edge computing for Internet of Drone Things applications," *Wireless Commun. Mobile Comput.*, vol. 2022, p. 5667012, 2022.

[44] Q. U. A. Mastoi *et al.*, "Hybrid workload-enabled and secure healthcare monitoring sensing framework in distributed fog-cloud network," *Electronics*, vol. 10, no. 16, p. 1974, 2021.

[45] M. Ahmad, M. Bilal, A. Jolfaei, and R. M. Mehmood, "Mobility-aware blockchain-enabled offloading and

scheduling in vehicular fog cloud computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4212–4223, 2021.

[46] M. A. Mohammed, J. Nedoma, R. Martinek, P. Tiwari, and N. Kumar, "Blockchain-enabled cybersecurity efficient IIOHT cyber-physical system for medical applications," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2466–2479, 2022.

[47] A. Cohen, "Client-side zero-shot LLM inference for comprehensive in-browser URL analysis," arXiv:2506.03656, 2025.

[48] Y. Xue, E. Spero, Y. S. Koh, and G. Russello, "MultiPhishGuard: An LLM-based multi-agent system for phishing email detection," arXiv:2505.23803, 2025.