National College of Ireland

FDFN-SA: The Lightweight Phishing detection system for endpoint devices

MSc Research Project
Cyber Security

Emmanuel Ikelia
Student ID:23284153

School of Computing
National College of Ireland

Supervisor: Dr. Mosab Hamdan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Emmanuel Ikelia
Student ID:	X23284153
Programme:	MSc. Cyber Security **Year:** One
Module:	MSc (Research) Practicum/Internship
Supervisor:	Dr. Mosab Hamdan
Submission Due Date:	11/08/2025
Project Title:	FDFN-SA: The Lightweight Phishing detection system for endpoint devices
Word Count:	**6,565** **Page Count: 20**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Emmanuel Ikelia

Date: 10/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

FDFN-SA: The Lightweight Phishing detection system for endpoint devices

Emmanuel Ikelia
X23284153

Abstract

Phishing is a type of cyber-attack that involves luring people into clicking on deceptive URLs designed to evade traditional phishing detection tools, resulting in a critical need for lightweight and more innovative anti-phishing solutions that can be deployed on endpoint devices. Over the past decade, security experts have increasingly turned to employing Artificial Intelligence-based solutions, which have proven to be more effective, but often come with the requirement of high computing costs that are not suitable for deployment on endpoint devices. To overcome these limitations, we introduce the FDFN-SA, short for Fuzzy-Driven Fusion Network with a Sparse Autoencoder, which is a lightweight, endpoint deployable version of the FDN-SA phishing detection model. This lightweight design was achieved by trimming the heavy inference steps and introducing an additional character-level branch to tackle obfuscation techniques, thereby improving detection capability. The resulting model utilizes only 5,676 parameters capable of delivering a decision within 100 milliseconds. Despite being lightweight, when trained and tested using the same dataset as the original FDN SA and another large language model (LLM), our proposed model achieved an accuracy of 94%, outperforming the FDN-SA's 92% accuracy, which required 156 milliseconds during inference. Additionally, our proposed model achieved 96% accuracy, compared to the 98% achieved by the LLM. The result is a model that can handle modern phishing tactics while remaining lightweight enough for deployment on edge devices. To encourage reproducibility, the code and dataset used in developing our model have been made publicly available.

Keywords: Phishing URL Detection, Lightweight Deployment, Quick Inference, Machine learning

1. Introduction

Phishing attacks remain one of the most persistent threats in cybersecurity, targeting not just individual users but also businesses and government agencies. In a typical phishing campaign, attackers create convincing spoofed emails and URLs that imitate legitimate websites. By exploiting human trust through social engineering, they aim to trick victims into revealing sensitive data, financial information, login credentials, or Personally Identifiable information. The scale of phishing attacks recently is rather staggering. The APWG (Anti-Phishing Working Group) recorded roughly 1,003,924 phishing incidents in the first quarter of

2025 alone (Anti-Phishing Working Group (APWG), 2025). The major challenge here is that phishing techniques have grown far more sophisticated, resulting in traditional defences such as URL blacklists, manual inspections, and other heuristic methods being less effective because attackers have mastered the art of URL obfuscation (Niharika, Vithya Ganesan, & Devi, 2025). For example, homographic attacks involve registering look-alike domains using characters from different alphabets, such as replacing a Latin "o" with a Cyrillic "o" to create a URL that is visually indistinguishable from the cloned legitimate one (Kowski, 2024). These tactics have proven capable of bypassing simple rule-based filters, browser blacklists, and other email scanning tools that rely on known-malicious URL databases, which often fail to update quickly enough to keep pace with new attacks.

While security awareness training, such as "Think before you click," does help, human vigilance alone cannot reliably detect these advanced tricks. This is why researchers have increasingly turned to artificial intelligence and machine learning over the past decade. AI-based approaches can learn patterns and features in URLs that humans and static rules would miss, with methods ranging from classical algorithms (e.g., decision trees, random forests) to deep learning architectures (e.g., CNNs, RNNs, transformers) that capture both lexical and semantic characteristics. (Cagatay Çatal, Görkem Giray, Bedir Tekinerdogan, Sandeep Kumar, & Suyash Shukla, 2022). However, most high-performing machine learning models are large and require significant computational resources; they often necessitate dedicated Graphics Processing Units (GPUs) or cloud computing environments. This demand for such high computing requirements makes them impractical for deployment on standard personal computers, such as corporate-owned, personally enabled (COPE) devices. This creates a critical gap: how can we achieve state-of-the-art detection accuracy without relying on heavy and specialized hardware?

This work addresses that gap by introducing FDFN-SA, the Fuzzy-Driven Fusion Network with a Sparse Autoencoder, as a compact, deployable phishing URL detector. Derived from the original FDN-SA (Fuzzy deep neural-stacked autoencoder) model proposed by Basit et al. (2025), FDFN-SA eliminates several heavy inference steps, optimizes each layer, and introduces a new character-level branch. The result is a model with just 56,760 trainable parameters (approximately 221 KB) that requires less than 1 GB of RAM and no dedicated GPU during inference, which takes 100 milliseconds to determine if an injected URL is phishing or not. Despite the reduced complexity, our proposed FDFN-SA model achieves a high accuracy in detecting phishing URLs, even outperforming its heavier predecessor. An experimental evaluation demonstrated that our proposed model achieved a classification accuracy of 94%, which is slightly higher than the 92% accuracy reported for the original FDN-SA model on the same dataset used for training and testing the original model. The observed improvement can be attributed to the inclusion of the character-level analysis branch in addition to the already existing Natural Language processing and structurally processed features of URLs. This enhancement enables the model to learn modern phishing obfuscation techniques, such as homograph domains and multi-layer redirects, that may evade NLP and structural feature analysis.

The contribution of this study is summarised as follows:

- We propose a lightweight phishing URL detection architecture (FDFN-SA): Our proposed model is a lightweight fusion network that integrates three branches of features that are locally extracted from URL samples without relying on any external API's, our final result is a quick and mostly accurate phishing detection system. This model will be tailored for on-device deployment without any heavy computational requirements.
- We improve on the inspired FDN-SA phishing detection Model: Through a comparative analysis in the context on inference time and accuracy, the inference speed of the proposed model was over 50% faster than that of FDN-SA model while also achieving 2% more accuracy compared to the FDN-SA model.

- We contribute to the Cybersecurity field: By combining efficient machine learning techniques with a deployment-oriented design, this work contributes a practical solution to the demand for endpoint-level phishing defences, enabling wider adoption of AI-driven phishing detection in everyday devices.

The rest of this paper is organised as follows: Section II provides a review of related works in the context of phishing URL detection, outlining existing approaches and their limitations. Section III details the design of the FDFN-SA phishing detection model, including its architecture and fuzzy strategy. Section IV showcases the experimental setup and results, comparing our proposed FDFN-SA to the original FDN-SA model, analyzing its performance both in the context of inference and accuracy, and Section V concludes this paper with recommended future works.

2. Background and Related Work

Over the past decade, researchers have moved from the traditional rule-based and black-list based defences against Phishing to automated machine learning models that can detect new and obfuscated phishing URLs by learning from their lexical, structural, and contextual patterns. This chapter presents a comprehensive review of the existing literature on phishing detection, exploring various models and datasets used in previous research. The aim is to highlight the strengths and limitations of these methods, identifying gaps that this current study needs to address.

Deep Learning-Based Phishing URL Detection: Studies involving deep learning models consistently show that the utilization of Deep Neural Networks (DNNs), particularly those with convolutional and dense architectures, has proven to improve the classification of phishing URLs when compared to standard rule-based or shallow models. (Dawabsheh, Jazzar, Eleyan, Bejaoui, & Popoola, 2022) Their work showed that augmenting URL extracted features with deep learning components improves the precision and recall when classifying URLs. (Boujiji, Berqia, & Hamadou, 2022) demonstrated that combining Extra-Tree ensemble methods with DNN can further improve the accuracy when classification is done in noisy environments. Additionally, (Ferdaws & Majd, 2024) proved that deep representations of URLs have better generalization attributes across varied phishing patterns, resulting in reduced false positives. Ultimately, these findings jointly suggest that the utilization of deep learning in the classification of URLs brings about robust and scalable improvements to the detection of phishing URLs.

Transformer and Large Language Models (LLMs) approach: These works explore sequence-based models to understand URL semantics. During a study by (Chanchal, Debasis, Tanmoy Maitra, & Bibekananda, 2024) where they compared BERT variants, they discovered that the pretraining of the transformer on language data assists models to better generalize unseen phishing patterns by recognizing subtle token cues and contextual anomalies. (Zhou & Liu, 2025) pushed this approach further by using generative LLMs to create phishing-aware representations that can anticipate the tactics used by attackers, thereby improving the rate of early detections. Finally, (Jia, et al., 2023) proposed that combining ON-LSTM attention mechanisms with gradient-boosting (XGBoost) can potentially boost both interpretability and performance in the generalization of phishing URLs. These studies reveal that such models have an advantage in capturing deeper semantic patterns in URLs.

Phishing URL Detection using Hybrid and Feature Fusion Architectures: Hybrid machine learning architectures emphasize combining engineered features with a combination of different learning algorithms for a more comprehensive phishing detection. (Kumar, 2023) Introduced a hybrid method that merges lexical, host-based, and content-based features into a unified model, this approach was able to achieve higher detection accuracy when compared to

baseline models that only utilize singular features. (Ewasakul & Oransirikul, 2024) in their studies, they developed a user-centric approach that learns from both URL structures and contextual click patterns to improve real-world phishing detection without significant user delays. Together, both papers highlight that combining domain expertise in feature design results in practical detection systems.

Real-Time and Efficient Detection Systems: The focus of these systems is based on speed and operational deployment while maintaining reasonable accuracy. (Sameen, Han, & Hwang, 2020) proposed a real-time phishing detection system capable of quickly analysing incoming URLS using lightweight neural networks within less than a second while still achieving a high accuracy. Using the same approach, (Aulia Kharisma Putri, Sanjaya, Wijaya, Johan, & Faza, 2024) showed that random forest classifiers, when properly tuned and given a balanced dataset, are capable of classifying phishing URLs with commendable performance with minimal computational requirements. Combining these studies proves that real-time phishing detection does not necessarily require heavyweight models.

Table 1: Summary of related works.

REFERENCE	ML METHOD(S) USED	DATASET(S)	PERFORMANCE METRICS	RESULTS	LIMITATIONS
(Kumar, 2023).	Decision tree, adaboost, KNN, naive bayes, random forest	Manually collected URL dataset	Precision, recall, f1 score, accuracy	Random forest achieved the best performance with a 99.75% accuracy	Lack of hybrid feature based machine learning
(Ewasakul & Oransirikul, 2024).	Lexical-based and content-based features using random forest	Malicious and benign webpages dataset	Precision, recall, f1 score, accuracy	Content-based features achieved an accuracy of 97.89%	Did not account for all feature groups
(Zhou & Liu, 2025).	Transformer neural network using different finetuning constraints	A single Phishing URL	Precision, recall, f1 score, accuracy	The full finetuning achieved the best accuracy of 98.37%	Lack of diverse models
(Chanchal, Debasis, Tanmoy Maitra, & Bibekananda, 2024).	BERT, ALBERT, DistilBERT, RoBERTa	UCI and Phishtank datasets.	Precision, recall, f1 score, accuracy	BERT achieved the highest accuracy of 99.29%	Lacked improvements on robustness and efficiency.
(Ferdaws & Majd, 2024).	DT, RF, LR, NB, SVM, KNN, XGB, LSTM-Large, LSTM-medium, CNN, CNN-LSTM	Phishing site dataset from kaggle	Precision, recall, f1 score, accuracy	LSTM-Large performed best with an accuracy of 98.1%	Does not take note of NLP features
(Aulia Kharisma Putri, Sanjaya,	SVM, RF, KNN	Phishing dataset from	Precision, recall, f1	Random forest achieved the	Lack of diverse models

Wijaya, Johan, & Faza, 2024).		kaggle	score, accuracy	highest accuracy of 93.7%	
(Boujiji, Berqia, & Hamadou, 2022).	Extra-tree and DNN algorithm	Collected from openphis and phishtank sources	Precision, recall, f1 score, accuracy	The DNN achieved a better accuracy up to 99.35%.	Limited models and features utilized
(Jia, et al., 2023).	LSTM, BiLSTM, BiLSTM-Att, CNN-BiLSTM-Att and Xgboost	URL dataset	Precision, recall, accuracy	XGBoost achieved the best performance of 98.98%	Utilized only NLP features.
(Sameen, Han, & Hwang, 2020).	Random Forest Classifier	A dataset of 100,000 URLs, comprising both phishing and benign examples.	Precision, recall, f1 score, accuracy	Random forest achieved an accuracy of 98%	Limited models utilized
(Vidyasri & Suresh, 2025).	FDN-SA = Fuzzy Deep Neural Network (DNN) + Deep Stacked Autoencoder (DSA)	Dataset from Kaggle	Accuracy, True Positive Rate (TPR), True Negative Rate (TNR)	Accuracy: 0.920 TPR: 0.925 TNR: 0.921	No explicit runtime performance evaluation and unclear computational cost for real-time deployment.

Overall, as illustrated in Table 1, most related works provided by machine learning techniques tend to perform well in a controlled experimental lab environment but have been observed to underperform when deployed for endpoint protection, especially when faced with adversarially crafted phishing URLs.

III. Proposed Lightweight and deployable Phishing Detection system

This studies introduces a lightweight phishing detection model which can deployed through browser extensions is installed in a via a fast API server in conjunction with a browser extension with the primary function of extracting URL strings from a web browser while at the same time receiving the final decision from our proposed model via the fast API serve, the fast API server will be embedded within an execution file with our proposed FDFN-SA model as the main engine used in making the final decision below is a high level system diagram to be considered during the deployment:

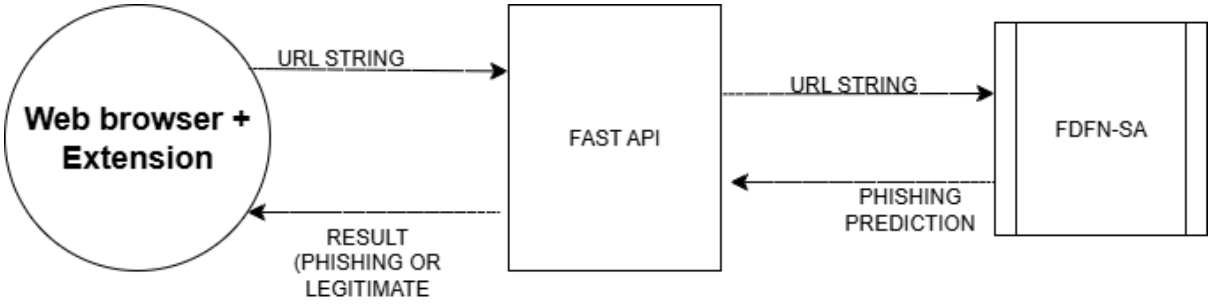

Figure 1. Methodology for deploying the FDN-SA model.

3.1 Proposed FDFN-SA Phishing Detection Model

The FDFN-SA model (Fuzzy Deep Fusion Network with Stacked Autoencoder) is a lightweight, multi-branch deep learning model designed for phishing detection. Our proposed model is inspired by and improves upon the FDN-SA approach by (Vidyasri & Suresh, 2025), instead of a heavy multi-stage pipeline (DNN + fuzzy layer + deep stacked autoencoder as in FDN-SA), our model integrates feature extraction and classification into a single end-to-end trainable network consisting of three feature processing branches (Character-level, Natural Language processing (NLP), and locally engineered structural features) whose outputs are meticulously fused via a learnable fuzzy-weighted layer and finally passed into a Deep Neural Network (DNN) where the final classification process takes place and decision is made if the inputted URL is phishing or not.

What was replaced from the original FDN-SA model with Justifications

Table 2. gives a brief description of how we reengineered the FDN-SA model into our lightweight and deployable FDFN-SA phishing detection model:

Table 2: Summary of related works.

Area	Original FDN-SA	FDFN-SA (Ours)	Why / Justification	Expected Impact
Structural handling	Hand-engineered features passed to the deep stack	Standardize → Sparse Autoencoder (33→33) → Dense(64)	AE de-noises and compacts correlated features; keeps input small and stable	Better generalization; reduced overfitting; lower latency
NLP handling	Lexical features from URL tokens using TF-IDF on word and character n-grams; fed directly into fusion stage with no dedicated NLP branch or fixed-dimension representation	Dedicated NLP branch with TF-IDF (1–2 grams) + BoW (1–3 grams) from sanitized URL text, reduced to fixed 600-D vector	Separate branch allows independent learning and optimization of lexical patterns; fixed size ensures the model has consistent input shape making it more efficient for deployment	More stable NLP performance; faster real-time inference; easier integration into lightweight models
Character handling	Not Applicable	Dedicated character-level CNN branch: embedding → Conv1D(64, kernel=5) → GlobalMaxPool	Learns visual, positional, and obfuscation patterns directly from raw URLs; complements entropy-based features for detecting homoglyphs	Better detection of obfuscated or manipulated URLs; stronger defense against adversarial URL modifications

Fusion method	Neyman similarity + DBN to generate fusion coefficients	Softmax “fuzzy” attention over branches (learned weights α_char, α_nlp, α_struct)	End-to-end learning removes hand sorting; interpretable weights per URL	Higher accuracy with fewer steps; improved interpretability; reduced complexity
Classifier head	DNN + Deep Sparse Autoencoder (DSA) stack for classification	Multi-layer dense head (32→16→8 neurons) with ReLU activations, dropout layers between each dense layer, and L2 weight regularization.	This technique gives a Smaller, more regularized network that reduces overfitting and computational load while maintaining accuracy.	Faster inference on edge devices; more stable predictions; easier deployment in low-resource environments

3.2 System Architecture and Algorithm

As illustrated in table 2, The core idea of this phishing detection model is to utilize only the URL string as an input with no additional calls or heavy content analysis, inject it into the model which then processes it into three different feature extraction branches comprising of the numerical structural features, NLP features and raw character level features and then combine these branches to give insights to the final classifier to make the final prediction if the url is phishing or not. The following high-level architectural diagram illustrates the implementation of the FDFN-SA phishing detection model.

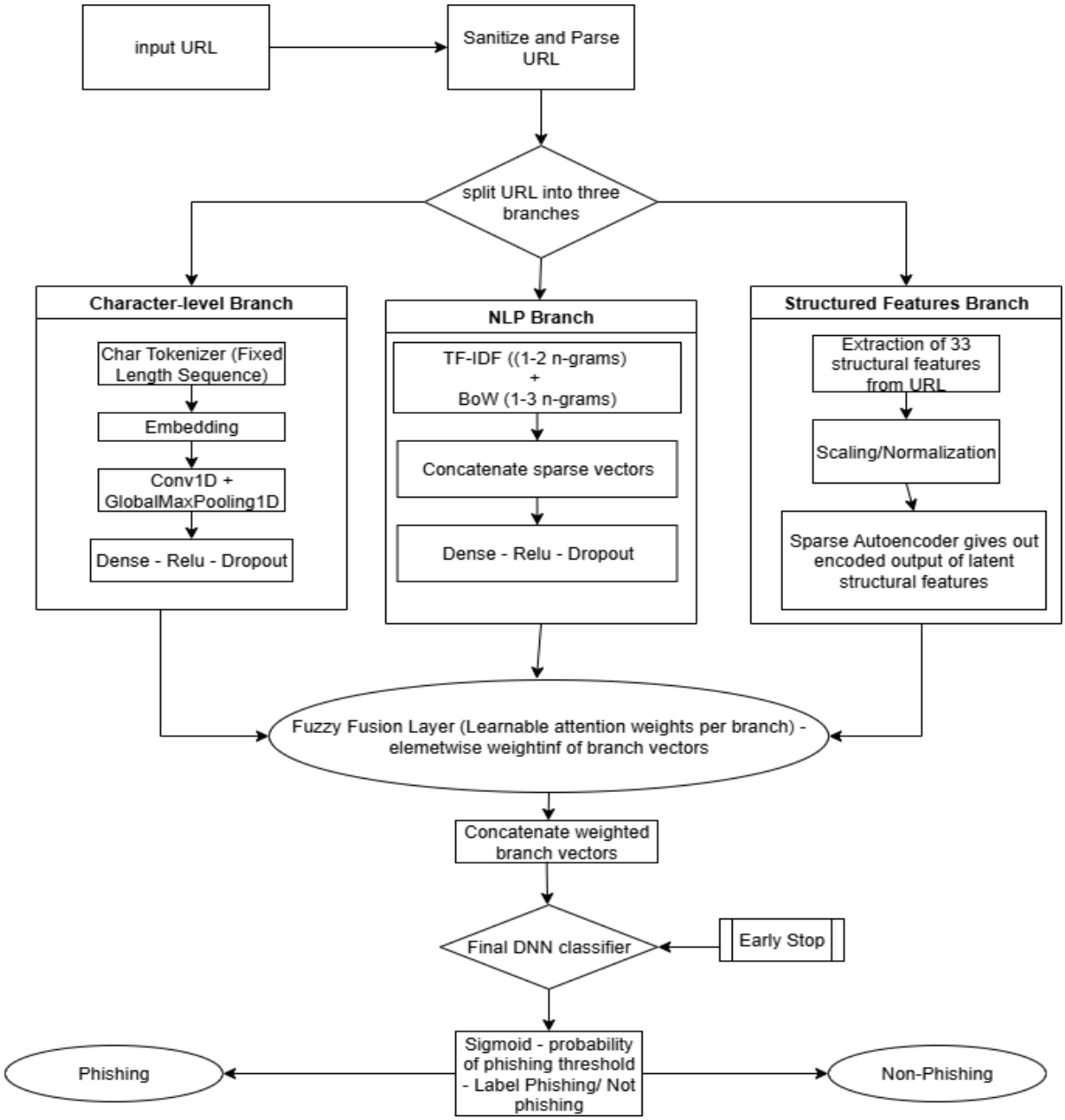

Figure 2. High-Level Architecture of the Proposed FDFN-SA Phishing URL Detection Model

For a more detailed illustration of the proposed FDNFA model, the pseudo-algorithm below outlines the main logic behind the implementation of the model:

Input: URL set U with labels Y (0=legitimate, 1=phishing)
Output: Trained model M (prediction: phishing or legitimate)
Feature Extraction
05 For each url ∈ U:
06 s_struct ← ExtractStructFeatures(url)
07 t_text ← CleanText(url)
08 c_char ← CharSeq(url, 260)
Train/Test Split
09 (train_idx, test_idx) ← StratifiedSplit(U, Y, test_size=0.2, seed=SEED)
NLP Feature Preparation
10 TFIDF_word ← Fit TFIDF(1-2, max_features=300) on text[train_idx]
11 TFIDF_char ← Fit TFIDF(1-3, max_features=300) on text[train_idx]
12 X_nlp_train ← Concat(TFIDF_word.transform(text[train_idx]), TFIDF_char.transform(text[train_idx]))
13 X_nlp_test ← Concat(TFIDF_word.transform(text[test_idx]), TFIDF_char.transform(text[test_idx]))
Structural Feature Preparation
14 scaler ← Fit StandardScaler on struct[train_idx]
15 E ← TrainSparseAutoencoder(scaler.transform(struct[train_idx]), input_dim=33, bottleneck=3)
16 X_struct_enc_train ← E(scaler.transform(struct[train_idx]))
17 X_struct_enc_test ← E(scaler.transform(struct[test_idx]))
Character Sequence Input
18 tokenizer ← Fit CharTokenizer(max_len=260) on text[train_idx]
19 X_charseq_train ← tokenizer.transform(char_seq[train_idx])
20 X_charseq_test ← tokenizer.transform(char_seq[test_idx])
Model Branches
21 Char: Input(260) → Embedding(32) → Conv1D(64,5,ReLU) → GlobalMaxPool → Dropout(0.1) → h_c(64)
22 NLP: Input(600) → Dense(64,ReLU) → Dropout(0.1) → h_n(64)
23 Struct: Input(3) → Dense(64,ReLU) → Dropout(0.1) → h_s(64)
Fusion & Classification
24 gates = Dense(3, softmax)(Concat(h_c,h_n,h_s)) → (α_c, α_n, α_s)
25 h_f ← α_c*h_c* + *α_n*h_n + α_s*h_s
26 h_f → Dense(32,ReLU) → Dropout(0.3) → Dense(16,ReLU) → Dense(8,ReLU) → Dense(1,sigmoid)
Training
27 Compile(model, Adam(0.001), binary_crossentropy, metrics=[AUC,Precision,Recall])
28 Train with EarlyStopping(patience=3), ModelCheckpoint, class_weight/focal_loss if imbalance
29 Fit on [X_charseq_train, X_nlp_train, X_struct_enc_train], Y[train_idx], validation_split=0.1
Return Artifacts
30 Save {model, TFIDF_word, TFIDF_char, tokenizer, scaler, encoder} → M
31 Return M
Prediction Procedure
32 s_enc ← E(scaler.transform(ExtractStructFeatures(url)))
33 t ← CleanText(url); x_nlp ← Concat(TFIDF_word.transform(t), TFIDF_char.transform(t))
34 c ← CharSeq(url,200)
35 p ← model.predict([c, x_nlp, s_enc])
36 Return phishing if p ≥ 0.5 else legitimate

Pseudo code 1: Pseudo Algorithm of the FDFN-SA model

The result of the implemented system is a lightweight, end-to-end phishing detection machine learning model that is capable of inputting raw URL strings and outputs a probability score between 0 and 1, where any score above 0.5 indicates that the injected URL is most likely a deceptive phishing URL. As logically outlined in Pseudo code 1, the FDFN-SA model integrates three feature branches with a learnable, fuzzy-weighted fusion, which is then classified by a compact Deep Neural Network classifier head.

3.2.1 Feature Branches

A. Structural processing branch: This branch collects a copy of the injected URL string, uses Python logic to engineer over 33 different structural features. These features capture measurable properties of the URL, such as length, hostname length, special character count, presence of an IP address, and statistical measures like entropy.

i. One high-impact feature is the Shannon entropy, which is a statistical measure that captures potential obfuscation in a URL string. It measures the unpredictability of characters in the string. The Shannon entropy can be mathematically represented as follows:

$$H = -\sum_{x \in X} p(x) \log_2 p\,(x) \tag{1}$$

Where:

- p(x) is the probability of character x in the string. The higher the entropy value, the more likely it is that the URL string was deliberately obfuscated, which often indicates phishing attempts.

ii. After these structural features have been successfully extracted, the resulting vector is standardized and passes through a dense transformation layer, as illustrated in equation (2). To produce the branch output "h_s:"

$$h_s = \text{Dropout}_{0.1}\big(\sigma(W_s x_{\text{struct}} + b_s)\big), \quad h_s \in R^{64} \tag{2}$$

Where:

- $h_s \in R^{64}$ = structural branch output vector with 64 dimensions
- x_{struct} = standardized structural feature vector extracted from the URL
- W_s = weighted matrix of the dense layer that maps input features into a 64-dimensional hidden space
- b_s = bias vector of the dense layer
- σ = ReLU activation function
- $\text{Dropout}_{0.1}$ = dropout function with rate 0.1 applied during training to prevent overfitting.

These extracted features (h_s) from equation (1) are further standardized and passed to a sparse autoencoder, where they are encoded and compressed, ready to be fused with other feature branches via the fuzzy fusion layer in equation (8).

A graphical representation of what usually happens within a sparse autoencoder is shown in Figure 3:

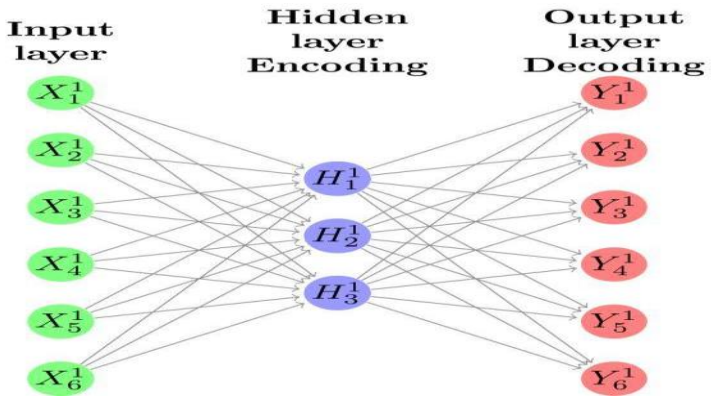

Figure 3. Sparse Autoencoder structure (Loey & Elsawy, 2017)**.**

B. Character level processing branch: in this branch, raw URL strings are collected and rendered as a fixed length character sequence which is then passed through a Convolutional Neural Network (CNN) where it is first embedded by converting each character index in the padded URL into a dense, and trainable vector presentation the next step is handled by the Conv1D that detects short patterns of character sequence like suspicious substrings, homoglyph patterns and inserted symbols, and finally Global Max pooling reduces each filters activation map to a single value (Pytorch, 2025).
The character level branch can be mathematically represented as follows:

i. Embedding and convolution with ReLu:

$$C_{t^{(k)}} = \sigma\left(\sum_{i=0}^{w-1}\sum_{j=1}^{d} \mathrm{v}W_{i,j}^{(k)}\ E_{\phi(x_{t+i}),j} + b^{(k)}\right) \quad (3)$$

where:

- $x_1 : L$ = the padded/truncated character sequence,
- $E \in \mathbb{R}^{v \times d}$ = the embedding matrix,
- W = Kernel size, d is the embedding dimension
- K = 1,…,K is the filter index,
- σ = ReLU activation function
- t = Position in the input character sequence
- d = Embedding dimensions (size of each character vector)
- $b^{(k)}$ = Bias term for filter index K
- $\phi(xt+i)$ = index of the character in the vocabulary
- $C_{t^{(k)}}$ = output activation for filter K at position t

ii. Global max pooling to form branch output:

$$h_c^{(k)} = \max_{1 \le t \le L-w+1} C_t^{(k)}, h_c = \left[h_c^{(1)}, \ldots, h_c^{(K)}\right]^T \in R^K \quad (4)$$

Where:

- $C_t^{(k)}$ = Activation value filter for k at position t after convolution + ReLU
- L = Padded or truncated sequence length (number of characters after processing
- $h_c^{(k)}$ = Single scalar summary for filter K after pooling.
- t = Position index over which the filter slides
- $L - W + 1$, derives the number of valid convolution steps for a kernel of with W in a sequence of length.

- L = Length of the padded or truncated input character sequence
- W = Kernel size of the convolution filter
- k = Index of the convolutional filter
- R^K = Final dimensionality of the character branch output vector

For better explainability, Figure 5. gives a graphical representation on what happens in the character level branch representing equation (2) and (3) above:

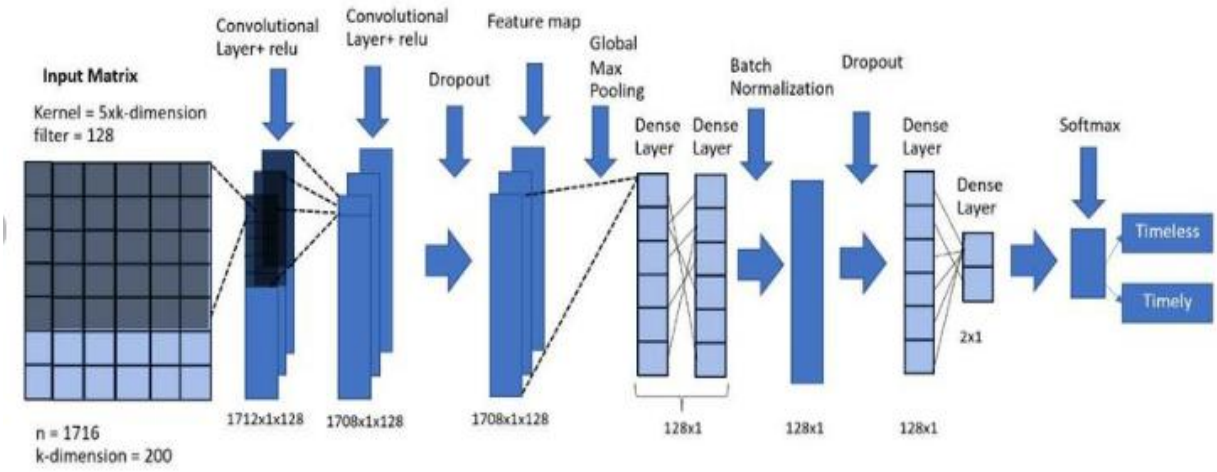

Figure 4. Convolutional neural network Conv1D architecture (Gatchalee, Waijanya , & Promrit, 2023)

C. Natural Language Processing Branch: The intended function of the NLP branch is the capturing of human readable patterns and token relationships that the structural and character level branches may not pick up, to achieve this, a complete URL string is inputted and sanitized by removing the protocols such as http://, ports and other special characters before converting the resulting string into lower case.

The feature extraction process was done in the following stages: First, TF-IDF vectorization was applied to the cleaned URL text using unigrams, bigrams, and trigrams. This captures common substrings and short sequences that frequently occur in phishing URLs, such as login, verify, secure, or their concatenated forms. In parallel, character-level n-gram features (1–3 characters) were extracted to capture fine-grained structural patterns. Both representations were concatenated into a single feature vector for the NLP branch input features (concatenated TF IDF + Char-Ngram):

$$x_{\text{nlp}} = [\text{TFIDF}(t),\ \text{CharNgram}(t)] \in R^M \tag{5}$$

Where:

- t = cleaned URL text (lowercased and safe characters kept)
- M = total dimensionality after concatenation
- CharNgram(t) = Character level n-gram features (1-3 characaters) extracted from the same cleaned text t
- TFIDF(.) and CharNgram(.) may internally use Bag of Words (BoW) representations before weighting

- x_{nlp} = final feature vector for the NLP branch, obtained by concatenating TF-IDF and character n-gram feature.

Weighting output from equation (4) (Dense layer + Dropout → output vector h_n):

$$h_n = \text{Dropout}_{0.1}\left(\sigma\left(W_n\, x_{\text{nlp}} + b_n\right)\right), h_n \in R^{64} \tag{6}$$

Where:

- $W_n \in R^{64\times M}$ = weight matrix
- $b_n \in R^{64}$ = the bias vector
- $\sigma(z) = max(0, z) - ReLUactivation$
- Dropout with rate 0.1 applied to reduce overfitting.
- $h_n \in R^{64}$ = Output vector of the NLP branch (64-dimensional)
- x_{nlp}= input feature vector from the NLP branch (concatenated TF-IDF and char n-gram features).

E. Fuzzy Fusion Layer

The main idea behind the fuzzy fusion layer, inspired by fuzzy logic systems, is to apply partial membership where all three branches of the FDFN-SA model contribute to the final classification, albeit in different proportions. This allows the model to rely more on robust branches when others are less informative for a given URL sample. This process helps speed up the model's inference time. This simple but effective concept replaces the more complicated Neyman similarity + DBN fusion method from the original FDN-SA phishing detection model (Price, Price, & Anderson, 2019). As follows is a detailed description of how the Fuzzy fusion layer functions: Concatenation of the three branch outputs: Here, three vectors representing the three branches, $h_c, h_n,$ and h_s derived from equations(4), (6), and (2), respectively, are concatenated into a single vector as follows:

i. Concatenation of the three branch outputs: Here, three vectors representing the three branches, $h_c, h_{n,}$ and h_s, derived from equations (4), (6) and (2), respectively, are concatenated into a single vector as follows:

$$z = [h_c, h_n, h_s]\ where\ each \in \mathbb{R}^{64} \tag{7}$$

Where:

- h_c = Output vector from the character branch
- h_n = Output vector from the NLP branch
- h_s = Output vector from the structural branch
- $z \in \mathbb{R}^{64}$ = Fused representation of the three branches with 64 dimensions each

ii. SoftMax attention weight: Here, a dense (3, SoftMax) layer processes z to produce three scalar weights which serve as a fuzzy attention mechanism. This mechanism dynamically adjusts the branch importance based on the input:

$$(\alpha_c, \alpha_n, \alpha_s), \alpha_i \geq 0, \sum_i \alpha_i = 1 \tag{8}$$

Where:

- $(\alpha_c, \alpha_n, \alpha_s)$ = weighted coefficients for all three branches
- $\text{i} \in \{\text{c}, \text{n}, \text{s}\}$ = index set referring to the three branches

iii. The final fuzzy fusion layer: The fusion layer is where all three branches of the FDFN-SA model is finally fused to be injected into the final classifier. This layer can be computed as:

$$f_{fuse} = \alpha_c, h_c + \alpha_n, h_n + \alpha_s, h_s \quad (9)$$

Where:

- α_c, α_n and α_s = weighted coefficients for all three branches
- h_c, h_n and h_s = Output vectors for all three branches

D. Classifier Head (Deep Neural Network)

This is where the final prediction is done by a lightweight Deep Neural Network (DNN) that receives the fused vector from the fuzz fusion layer, as represented in equation (9). The DNN utilized by the FDFN-SA model consists of three fully connected layers with 32, 16, and 8 neurons, respectively. The process is concluded by a sigmoid output neuron for the final binary classification (Subas, 2020).

What actually happens within these connected layers is the progressive transformation of inputting the fused features into a more separable space. This enables a more accurate classification between a phishing and a legitimate URL.

The general operation happening within the final DNN classifier can be expressed as:

$$\hat{y} = \sigma\big(W_4.ReLU\big(W_3.ReLU\big(W_2.ReLU\big(W_1\, f_{fuse} + b_1\big) + b_2\big) + b_3\big) + b_4\big) \quad (10)$$

Where:

- $\hat{y}$ = Predicted probability between 0 and 1 with a threshold of 0.5. An output greater than 0.5 indicates that the injected URL is most likely phishing-related, and an output less than 0.5 indicates that the URL is most likely legitimate
- f_{fuse} = Fused feature vector as injected from the fuzzy fusion layer.
- W_1,W_2,W_3 and W_4 = weighted matrices for each dense and fully connected layer within the final DNN classifier head.
- $b_1, b_2, b_3,$ and b_4 = Bias vectors/scalars for each dense layer
- ReLU = non-linear activation applied at the hidden layers.
- σ = sigmoid activation applied at the output layer to map the score into a probability

4. Experimental Results and Discussion

This section discusses how we evaluated the performance of our proposed model while comparing it alongside the FDN-SA model, which our proposed model is inspired by, alongside the Large Language Model by (Zhou & Liu, 2025) while utilizing the same datasets used to train and appraise the prior mentioned model to ensure consistency and reliability, we also experimented on the same hardware setup as follows:

- CPU: Intel core i5-8365U (4 physical and 8 logical cores at 1.6GHz each)
- RAM: 8GB
- GPU: We did not use any GPU for this experiment due to our target deployment niche are regular personal computers without specialized hardware setup.
- The software setup for the experiment conducted is as follows:
    - Operating system: Windows 11 (64bits)
    - Text editor: Jupyter Notebook
    - Data Handling & Analysis: pandas 2.2.2, NumPy 1.26.4

- Machine Learning: scikit-learn 1.4.2, XGBoost 2.0.3
- Deep Learning: TensorFlow 2.16.1 (Keras API) for character-level CNN branch
- Visualization: matplotlib 3.8.4, seaborn 0.13.2

4.1 Dataset and preprocessing: To establish a comprehensive and balanced foundation for our experimental setup, we utilized two publicly available phishing URL datasets from Kaggle. The first is the webpage phishing detection dataset by **(Shashwat Work, 2022)**. This dataset was also utilized during the training and testing of the FDN-SA model. This balanced dataset contains 11,430 URLs, of which 50% phishing and 50% legitimate, alongside 87 pre-extracted features, of which 24 are web page content features and 7 are features obtained from external services and APIs. The second dataset we used is the phishing and legitimate URL collections by **(Harisudhan411, 2023)**, contained in this dataset are 822,000 URL entries labelled as phishing or legitimate, but we only sampled 20,000 URLs, which we balanced into 10,000 phishing and 10,000 legitimate URLs. This dataset was also used by **(Zhou & Liu, 2025)** in the development of their large language model for generative phishing URL detection.

4.2 Performance Analysis: The following Performance analysis was conducted using a balanced dataset from (Shashwat Work, 2022), this dataset contains 11,430 URLs alongside 87 extracted features. According to the authors, this dataset was synthetically designed to be utilized when training phishing detection systems that are machine learning powered.The parameters used in measuring our proposed model are as follows: Accuracy is the overall fraction of correct phishing predictions, Precision is the correct positives among predicted phishing URLs, Recall is the correct positives among actual positives, F1-score is the harmonic mean of precision and recall, ROC-AUC is the separability across thresholds, TPR is the proportion of positives correctly detected and TNR proportion of negatives correctly rejected.

4.2.1 Confusion Matrix: The confusion matrix in Figure 5. illustrates how well the FDFN-SA model performed in correctly identifying and incorrectly identifying phishing emails, as well as vice versa. The figures from the confusion matrix explain that out of 2,286 samples of an equal amount of legitimate and phishing URLs, the model was able to correctly identify 1,064 legitimate URLs and 1076 phishing URLs, while misclassifying 79 legitimate URLs as phishing and 67 phishing URLs as legitimate. Although this is not a perfect result, this still shows that the model has a somewhat strong performance in detecting phishing URLs.

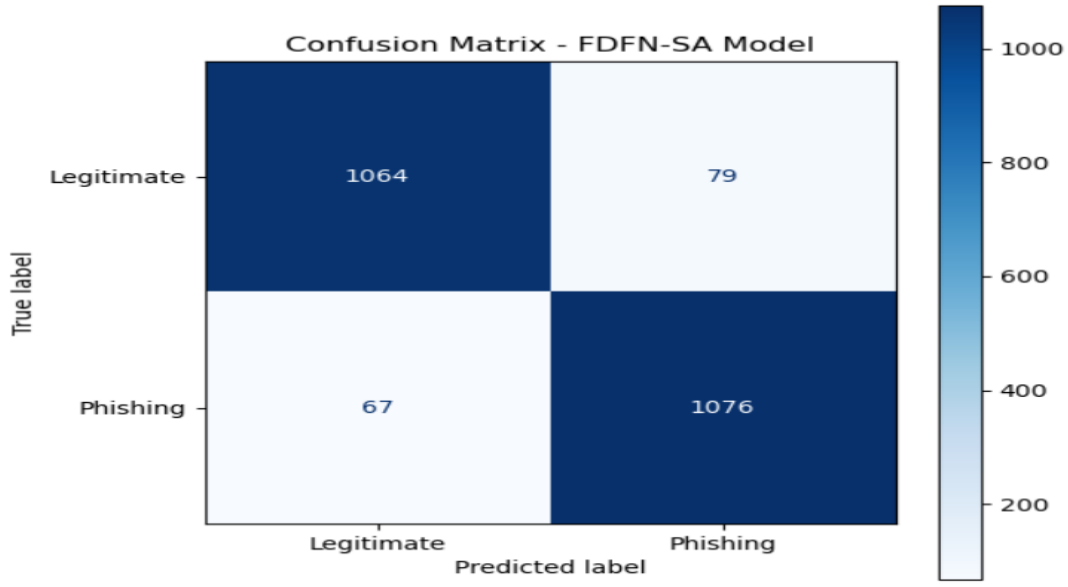

Figure 5. Confusion matrix

2. **ROC Curve with AUC Score:** The plot from Figure 6. illustrates the ROC curve of our proposed model on the test set; the model was able to achieve an AUC score of 0.9822 or 98%. And observing the curve's proximity to the top-left corner shows that the model has a good ability to discriminate between phishing and legitimate URLs

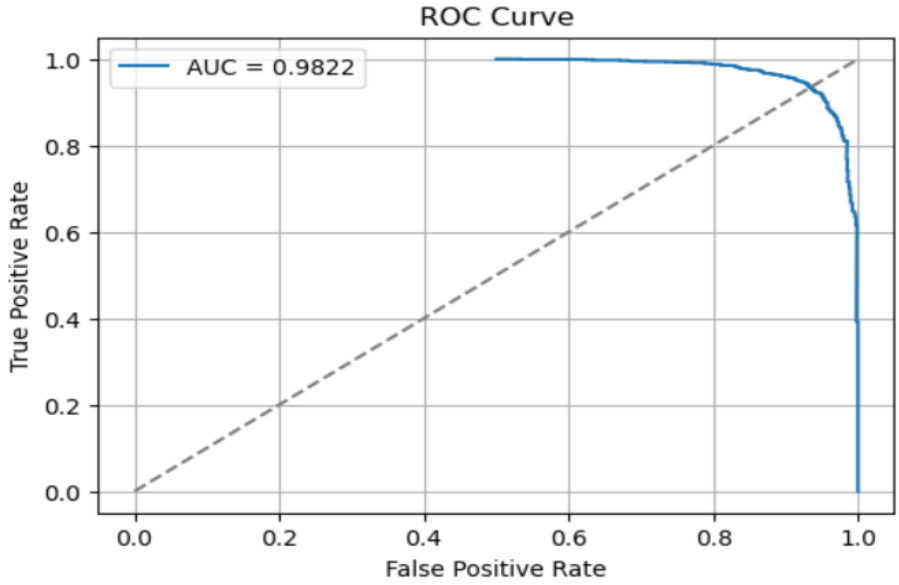

Figure 6. ROC Curve

4.3.1 Performance analysis using dataset from (Shashwat Work, 2022)

The original FDN-SA model was evaluated using three parameters: The Accuracy, The true positive rate and the true negative rate alongside other models, for the purpose of a fair comparison, we trained our model using exactly the same dataset by (Shashwat Work, 2022) which was used during the development of the FDN-SA model, Table 3. illustrates a fair comparison between both models:

Table 3. Performance table against FDN-SA

Model	TPR	TNR	Accuracy
Boosted DT	0.75	0.73	0.75
RF+Case	0.78	0.77	0.78
PDGAN	0.80	0.80	0.80
DMODL-PAD	0.83	0.83	0.85
DSA	0.89	0.90	0.90
ANN	0.88	0.87	0.88
FDN-SA	0.90	0.92	0.92
Proposed FDFN-SA	0.94	0.93	0.94

The proposed FDFN-SA model achieved a higher TPR and overall accuracy compared to the original FDN-SA model, while the TNRs of both models were comparable. Ultimately, the figures from Table 3 indicate improved phishing detection capabilities of our proposed model over the FDN-SA model without sacrificing its ability to recognise legitimate URLs

4.3.2 Performance analysis using the dataset from (Harisudhan411, 2023)

To further validate the performance of our proposed model, we tested it against a larger dataset of 20,000 entries and compared it with research done by (Zhou & Liu, 2025) on a heavy Large Language Model.

Table 4. Performance table against LLM

Model	Precision	Recall	F1	Accuracy
URL2Vec	0.87	0.87	0.87	0.87
URLNet	0.91	0.91	0.91	0.91
URLTran	0.97	0.97	0.97	0.97
URLLM	0.99	0.98	0.99	0.98
Proposed FDFN-SA	0.96	0.96	0.96	0.96

From the above table, it is rather apparent that the Large Language Model performed significantly better than our proposed model; however, such large language models have heavy computational requirements compared to ours, which is more suitable for lightweight deployments.

4.3.3 Ablation Studies

An ablation study was conducted on our model with a focus on the three branches of the model to determine which branch has the highest effect on the performance of our model. Therefore, this analysis will be conducted in three phases as follows:

Table 5. Ablation table

Branch	Precision	Recall	F1	ROC AUC	Accuracy
No structural features branch	0.93	0.92	0.93	0.98	0.93
No NLP features branch	0.94	0.92	0.93	0.98	0.93
No Character level branch	0.89	0.90	0.90	0.96	0.90
All Branches present	0.94	0.94	0.94	0.98	0.94

The results from Table 5. Shows that the character level branch caused the most drop in performance of the FDFN-SA model, confirming its critical role in identifying obfuscated phishing URLS, although the other branches of NLP and Structural branches had minimal effects on the overall performance, they still play key roles in improving the precision of the model in a real world setting, for example, the structural branch captures statistical and layout based anomalies, this has the potential to reduce false positives by flagging abnormal URL lengths, entropy and formatting patterns. In contrast, the NLP branch, which focuses on token and phrase semantics, has the potential to boost recall by identifying brand-squatting and social engineering patterns that other branches may miss.

4.3.4 Most effective Structural Features

By side training a random forest model on the structural features and using the feature importances attribute, we were able to come up with the below plot, which illustrates the top 20 structural features that have the most importance in the context of detecting phishing URLs.

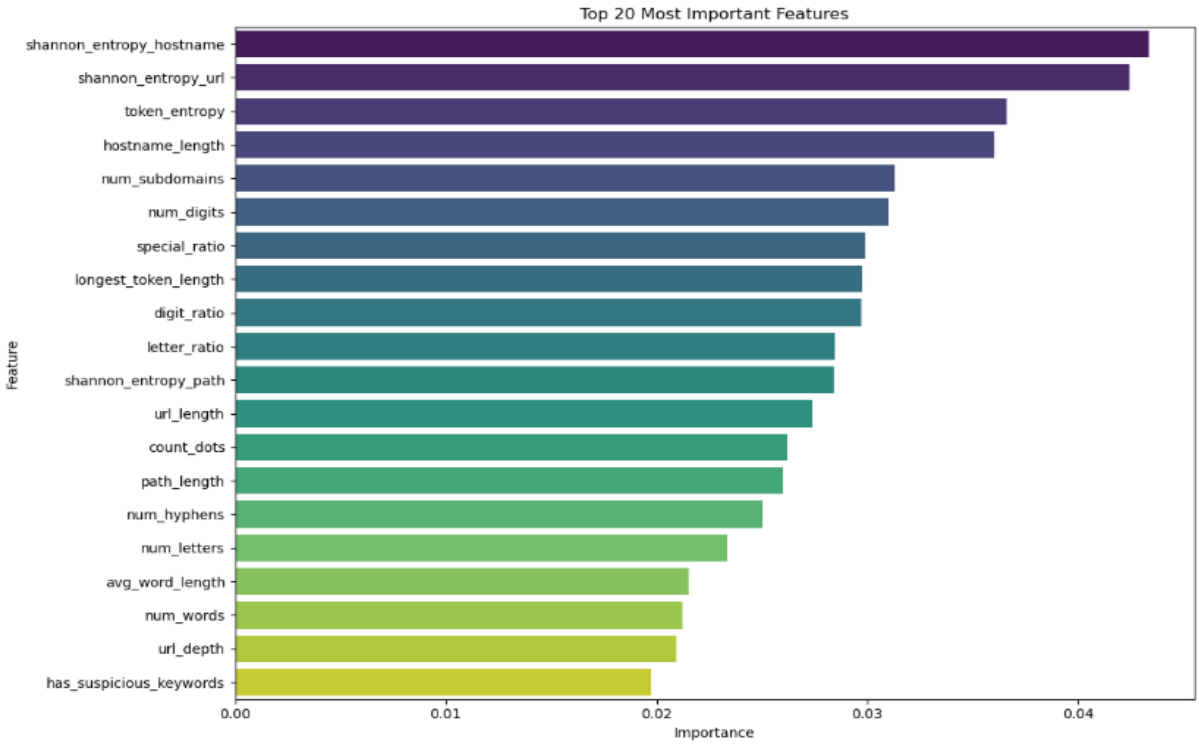

Figure 7. Top 20 most important features

From the plot in Figure 7. We can boldly state that Shannon entropy for both hostname and the URL string has the most impact on the structural features of a URL string. This confirms the theory from equation (1). Host names and URLs with higher entropies indicate a strong signal of obfuscation in phishing URLs. High entropies simply imply that such URL strings have long, random-looking strings of letters, numbers, and symbols to evade detection.

4.3.5 Comparative analysis on inference.

For a fair analysis on how our model compares with the original FDN-SA model in terms of how quick it takes to classify a URL and the computational cost required, we had to re-engineer the FDN-SA model to carry out a side-by-side comparison on the same computer hardware which was earlier indicated in the methodology section.

Table 6. Inference Benchmark

Model	Parameters	RAM Usage	CPU Usage	FLOPS	Average Inference Latency
FDFN-SA	56,676	533 MB	1,596	4,115,115	81.25 ms
FDN-SA	91,688	791.91MB	1,596 MHz	42,044	151.19 ms

The benchmarks from Table 6. demonstrates the minimal requirement required to run the FDFN-SA model while still giving a quick inference of less than one tenth of a second.

4.4 Findings

The final findings from the above evaluation demonstrate that the proposed FDFN-SA model has the potential to be a reliable yet lightweight phishing detection system suitable for endpoint deployment on computers with limited computing resources. On a balanced dataset, the FDFN-SA model outperformed the FDN-SA model from all standpoints, including overall accuracy,

true positive rate, and true negative rate, while requiring fewer computing resources. When compared against a Large Language Model by (Zhou & Liu, 2025) On a larger and more diversified dataset, the LLM achieved a higher accuracy of 98% compared to the 94% percent achieved by our proposed model. However, Large Language Models are known to be resource intensive, often requiring specialized hardware, therefore, despite it higher accuracy, they are not suitable for our niche of endpoint machines with limited computing resources. The ablation study demonstrated that the character-level branch made the most significant contribution to the overall performance of the FDFN-SA model. In contrast, the other branches showed little to no improvement in the overall performance of the model; however, they were still able to demonstrate minor but important gains in precision and recall. The analysis of structural features to determine their impact on phishing email detection revealed that Shannon entropy was at the top, further confirming that obfuscation remains a major tactic used by attackers in social engineering, such as using URLs to evade detection. The inference benchmark further prove that our proposed model meets the requirements for endpoint deployment on resource limited computers such as employees laptop thereby adding another layer of protection against the ever evolving phishing campaigns, during the benchmarking, our proposed model showed a quicker inference time compared to the original FDN-SA model while demanding about 38% fewer parameters and about 33% less RAM usage, but it was noticed that the FLOPs used by our model is much more higher when compared to the FDN-SA model, and this can be attributed to the important character level branch as it requires a lot of computational calculations.

5. CONCLUSION AND FUTURE WORK

In conclusion, this study successfully addressed the core research aim of designing a phishing URL detection model that delivers high detection accuracy while remaining lightweight for deployment on endpoint computers with limited computing resources. When compared to the original FDN-SA model on the same balanced dataset, the proposed FDFN-SA outperformed it in accuracy, true positive rate, and true negative rate, while utilizing approximately 38% fewer parameters, about 33% less RAM, and achieving a latency of less than 100ms during inference. The ablation studies conducted showed that the character-level branch has the most significant impact on the phishing detection accuracy of our model, with its removal resulting in a 4% decrease in accuracy. In contrast, the structural and NLP branches had minimal but complementary gains in other aspects of a phishing URL that the character-level branch may miss. Additionally, the inclusion of the character level branch is expected to give the model the capacity to flag unseen obfuscation tricks such as homoglyph and symbol insertions, the character level branch is also capable of capturing positional patterns and unusual spacing of characters, which are among the new tactics employed by attackers in the recent phishing URL trends. Although a lightweight phishing detection system for endpoint devices fills a critical gap, it comes with its trade-offs, including reduced accuracy compared to heavier and more sophisticated systems. First, reduced capacity headroom: Our proposed lightweight model is designed with a compact head that has fewer filters and parameters, allowing it to learn core URL patterns quickly but with limited room to represent rare composite patterns, such as IDN homoglyph chains with deep path bait. Secondly, our proposed lightweight model poses limited context breadth due to its reliance on URL-only signals without consulting the page content or 20 external feeds from third-party sources such as WHPIS, passive DNS, and domain age, which can assist in detecting certain families of phishing URLs and learn from newer trends and tactics. The third trade-off is in the context of adversarial resilience: with fewer parameters and little redundancy, carefully crafted obfuscations might easily escape the detection capabilities of the lightweight model.

Overall, the proposed FDFN-SA model successfully met our design objectives of improving upon the original FDN-SA model in both accuracy and computational cost. Ultimately, this work provides a practical solution to the cybersecurity landscape by introducing a lightweight and deployable phishing detection machine learning model.

During the development of this paper, we encountered the following limitations:

- Synthetic dataset: both datasets used for the training and evaluation of the proposed model may not reflect real-world conditions.
- Hardware benchmarking scope: The inference benchmarking was performed on a specific laptop configuration; this means that results may vary significantly across other devices with different configurations.
- Absence of adversarial testing: Our model was not evaluated against adversarially generated URLs that are intended to evade machine learning based phishing detectors
- Old dataset: Both datasets used for the training and evaluation of the proposed model are old, with the latest dating as far back as 2 years; this might not reflect the ever evolving obfuscation techniques currently used by attackers.

Future work should conduct rigorous tests on the FDFN-SA model, utilizing more recent, imbalanced, and possibly real-world datasets to evaluate its performance in such scenarios. Also, the model should be deployed in a real-world environment to validate its performance for its intended use on endpoint systems with limited resources.

References

Anti-Phishing Working Group (APWG). (2025). *Phishing Activity Trends Report, 1st Quarter 2025.* Lexington (Massachusetts): Anti-Phishing Working Group. Retrieved Jul 10, 2025, from https://docs.apwg.org/reports/apwg_trends_report_q1_2025.pdf

Aulia Kharisma Putri, J. W., Sanjaya, S. A., Wijaya, S. F., Johan, M. E., & Faza, A. (2024). Web URLs Phishing Detection Model with Random Forest Algorithm. *2024 5th International Conference on Big Data Analytics and Practices (IBDAP 2024)* (pp. 1-5). IEEE. doi:10.1109/IBDAP62940.2024.10689685

Boujiji, H., Berqia, A., & Hamadou, S.-H. (2022). Phishing URL Classification Using Extra-Tree and DNN. *2022 10th International Symposium on Digital Forensics and Security (ISDFS 2022)*, (pp. 1-6). doi:10.1109/ISDFS55398.2022.9800795

Cagatay Çatal, Görkem Giray, Bedir Tekinerdogan, Sandeep Kumar, & Suyash Shukla. (2022). Applications of deep learning for phishing detection: a systematic literature review. *Knowledge and Information Systems, 64*(6), 1457–1500. doi:10.1007/s10115-022-01672-x

Chanchal, P., Debasis, G., Tanmoy Maitra, & Bibekananda, K. (2024). A Comparative Study on Detecting Phishing URLs Leveraging Pre-trained BERT Variants. *2024 6th International Conference on Computational Intelligence and Networks (CINE 2024)* (pp. 209–214). IEEE. doi:10.1109/CINE63708.2024.10881521

Dawabsheh, A., Jazzar, M., Eleyan, A., Bejaoui, T., & Popoola, S. I. (2022). An Enhanced Phishing Detection Tool Using Deep Learning From URL. *2022 International Conference on Smart Applications, Communications and Networking (SmartNets 2022)* (pp. 1-6). IEEE. doi:10.1109/SmartNets55823.2022.9993984

Ewasakul, N., & Oransirikul, T. (2024). Machine-Learned Defense Mechanism Against Phishing URL Exploitation for User Protection. *2024 9th International Conference on Business and Industrial Research (ICBIR 2024)* (pp. 1614–1619). IEEE. doi:10.1109/ICBIR61386.2024.10875056

Ferdaws, R., & Majd, N. E. (2024). Phishing URL Detection Using Machine Learning and Deep Learning. *2024 IEEE World AI IoT Congress (AIIoT 2024),* (pp. 485–490). Seattle, W: IEEE. doi:10.1109/AIIoT61789.2024.10579005

Gatchalee, P., Waijanya , S., & Promrit, N. (2023). THAI TEXT CLASSIFICATION EXPERIMENT USING CNN AND TRANSFORMER MODELS FOR TIMELY-TIMELESS CONTENT MARKETING. *ICIC Express Letters 17, 1*, 91-101. doi:10.24507/icicel.17.01.91

Harisudhan411. (2023). Phishing and Legitimate URLs. Kaggle. Retrieved from https://www.kaggle.com/datasets/harisudhan411/phishing-and-legitimate-urls

Jia, Q., Guo, X., Zhang, M., Liu, M., Tian, X., Jin, X., & Ma, D. (2023). Phishing URL Recognition Based on ON-LSTM Attention Mechanism and XGBoost Model. *2023 5th International Conference on Electronics and Communication, Network and Computer Technology (ECNCT 2023)* (pp. 158-163). IEEE. doi:10.1109/ECNCT59757.2023.10280927

Kowski, J. S. (2024). Email obfuscation tactics elude security protections. Troy, Michigan: BNP Media. Retrieved from https://www.securitymagazine.com/blogs/14-security-blog/post/100653-email-obfuscation-tactics-elude-security-protections

Kumar, J. (2023). Hybrid Feature-Based Machine Learning Method for Phishing URL Detection. *2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC 2023)* (pp. 222-227). IEEE. doi:10.1109/ICSCCC58608.2023.10176901

Lim, K., Lee, K., Sommese, R., Jonker, M., Mok, R., claffy, k., & Kim, D. (2025). *Registration, Detection, and Deregistration: Analyzing DNS Abuse for Phishing Attacks.* Knoxville, USA: arVix.

Loey, M., & Elsawy, A. (2017). Deep Learning Autoencoder Approach for Handwritten Arabic Digits Recognition. doi:10.48550/arXiv.1706.06720

Niharika, P., Vithya Ganesan , & Devi, V. A. (2025). An Ensemble Deep Learning Based Detection System to Identify Phishing Attacks. *Journal of Information Systems Engineering and Management, Vol. 10 No.*(48s). doi:10.52783/jisem.v10i48s.9533

Price, S. R., Price, S. R., & Anderson, D. T. (2019). Introducing Fuzzy Layers for Deep Learning. *EEE International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA,*, 1-6. doi:arXiv:2003.00880

Pytorch. (2025). *torch.nn.Conv1d, PyTorch Documentation*. Retrieved Aug 09, 2025, from https://docs.pytorch.org/docs/stable/generated/torch.nn.Conv1d.html

Sameen, M., Han, K., & Hwang, S. O. (2020). PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System. *IEEE Access (Vol. 8, 2020), 8*, 83425–83443. doi:10.1109/ACCESS.2020.2991403

Shashwat Work. (2022). Web Page Phishing Detection Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset

Subas, A. (2020). Machine learning techniques. *Practical Machine Learning for Data Analysis Using Python*, 91-202. doi:doi.org/10.1016/b978-0-12-821379-7.00003-5.

Vidyasri, P., & Suresh, S. (2025). FDN-SA: Fuzzy deep neural-stacked autoencoder-based phishing attack detection in social engineering. *Computers & Security, 148*, Article 104188. doi: 10.1016/j.cose.2024.104188

Zhou, B., & Liu, J. (2025). Generative Phishing URL Detection Based on Large Language Model. *2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT 2025)* (pp. 1838–1842). IEEE. doi:10.1109/AINIT65432.2025.11035267