International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

Optimizing Phishing URL Detection via Feature Selection and Deep Learning model

A MULTI-MODEL DEEP LEARNING AND FLASK DEPLOYMENT FRAMEWORK

G. Lokesh, B. Hari, E. Tarun, G. Pavan Kumar
Department of CSE (Data Science)
Raghu Engineering College, Dakamarri(V)
Bheemunipatnam, Visakhapatnam Dist., 531162
lokeshgudivada22124@gmail.com

Ms.V.P.V. Bharathi
Department of Computer Science & Engineering
Raghu Engineering College, Dakamarri(V)
Visakhapatnam, Andhra Pradesh, India
bharathi.vanapalli@raghuenggcollege.in

***Abstract*—Phishing URL detection is a critical cyber-security challenge, as adversaries continuously evolve attack patterns to defeat blacklist and rule-based defences. This paper presents a comprehensive multi-model deep learning framework trained on a curated set of 20 discriminative features extracted from the publicly available dataset_B_05_2020 benchmark (11,430 balanced URL samples). Five architectures—Deep Neural Network (DNN), Feedforward Neural Network (FNN), Autoencoder-based classifier, Graph Convolutional Network (GCN), and TabTransformer—are trained, evaluated, and compared under a unified protocol. The DNN achieves the best performance with 95.0% accuracy, 95.2% precision, 94.8% recall, and 95.0% F1-score. All models are deployed in an interactive Flask web application with SQLite-backed user authentication and real-time URL prediction with confidence scoring. Results demonstrate that compact feature engineering combined with deep learning provides robust, low-latency phishing detection suitable for practical deployment.**

***Keywords*—Phishing Detection; Deep Neural Network; Tabular Deep Learning; Flask Deployment; URL Features; Cybersecurity; Graph Convolutional Network; TabTransformer**

I. INTRODUCTION

Phishing attacks remain one of the most prevalent and damaging forms of cybercrime, targeting individuals and organisations by masquerading legitimate websites to steal sensitive credentials. According to the Anti-Phishing Working Group (APWG), the number of phishing sites observed globally exceeded 1.2 million in 2023 alone, underscoring the urgency of robust automated detection mechanisms. Traditional rule-based and blacklist-driven approaches suffer from high false-negative rates because adversaries continuously mutate URL structures and page content to evade detection.

Deep learning has emerged as a powerful paradigm for phishing URL detection owing to its capacity to model complex, non-linear relationships among heterogeneous URL and content features without exhaustive manual feature engineering. Prior studies employing convolutional neural networks (CNNs), recurrent networks, and simple feedforward architectures have demonstrated promising results; however, they often rely on a single model, limiting generalisation. This paper proposes a multi-model ensemble framework that trains five complementary architectures—Deep Neural Network (DNN), Feedforward Neural Network (FNN), Autoencoder-based classifier, Graph Convolutional Network (GCN), and TabTransformer—on a curated set of 20 discriminative features extracted from a publicly available benchmark dataset.

The key contributions of this work are: (1) systematic feature selection reducing 89 raw attributes to 20 high-impact features; (2) simultaneous training and evaluation of five deep learning architectures; (3) deployment of the best-performing model (DNN, 95.0% accuracy) within an interactive Flask web application supporting user authentication and real-time URL prediction; and (4) an open empirical comparison of tabular deep learning models on the phishing detection task. The remainder of the paper is organised as follows. Section II reviews related work. Sections III and IV detail the dataset and methodology. Section V presents results. Section VI discusses the system architecture. Section VII concludes.

II. RELATED WORK

Phishing URL detection has attracted considerable research across multiple communities. Sahingoz et al. [1] evaluated seven machine learning classifiers on a dataset of over 73,000 URLs, reporting that random forest achieved 97.98% accuracy using NLP-based features. Mohammad et al. [2] proposed a hybrid feature set combining URL structure, content, and third-party services, achieving 92.4% accuracy with a neural network. These seminal studies highlight that feature engineering is as critical as the choice of classifier.

Deep learning approaches have progressively displaced hand-crafted feature pipelines. Huang et al. [3] applied character-level CNNs directly to raw URL strings, avoiding feature engineering entirely but requiring substantial training corpora. Le et al. [4] introduced URLNet, which learns multi-scale character and word embeddings for joint URL classification. Attention mechanisms further improved recall by focusing on salient URL sub-strings [5].

Tabular deep learning has received renewed interest since the introduction of TabNet [6] and TabTransformer [7], which apply attention to structured feature tables and outperform gradient-boosted trees on several benchmarks. Graph neural networks have also been applied to web security: Zhang et al. [8] modelled web-page link graphs with GCNs, capturing relational context unavailable to single-URL classifiers. Our work bridges these threads by benchmarking multiple architectures on an identical dataset under a unified training protocol, and by deploying the best model in an accessible web tool.

III. DATASET AND FEATURE ENGINEERING

A. Dataset Description

We use the publicly available dataset_B_05_2020 [9], a balanced benchmark containing 11,430 labelled URLs: 5,715 legitimate and 5,715 phishing instances. Each instance is annotated with 89 features spanning four categories: URL structure, domain characteristics, HTML/JavaScript page content, and external reputation signals. The balanced class distribution eliminates the need for oversampling and ensures unbiased evaluation.

TABLE I. Dataset Configuration

Parameter	Value
Total Samples	11,430
Total Features	89 (raw)
Selected Features	20 (URL + content-based)
Legitimate URLs	5,715 (50%)
Phishing URLs	5,715 (50%)
Train / Test Split	80% / 20%
Source	dataset_B_05_2020

B. Feature Selection

From the 89 available features, we select 20 high-impact attributes guided by mutual information gain and domain knowledge. URL structure features (length_url, nb_slash, nb_qm, nb_eq, nb_www, ratio_digits_url, ratio_digits_host, ip, length_hostname) capture lexical properties of the URL. Domain features (tld_in_subdomain, prefix_suffix) reflect structural abnormalities. Lexical features (shortest_word_host, longest_word_path) reflect word composition. Content-based features (phish_hints, nb_hyperlinks, ratio_intHyperlinks, domain_in_title) capture page-level signals. External reputation features (domain_age, google_index, page_rank) provide third-party validation. The final 20-feature vector is detailed in Table III.

C. Preprocessing

All numeric features are standardised to zero mean and unit variance using sklearn StandardScaler fitted on the training split alone to prevent data leakage. Missing values (< 0.3% of cells) are imputed with the training-set median. The dataset is split 80/20 (train/test) with stratification, yielding 9,144 training samples and 2,286 test samples. Feature matrices are exported as X_train.csv and y_train.csv for reproducibility.

IV. METHODOLOGY

A. Deep Neural Network (DNN)

The primary model is a three-layer fully connected DNN implemented in PyTorch. The architecture comprises an input layer of 20 units, a hidden layer of 128 ReLU-activated units with 0.3 dropout, a second hidden layer of 64 ReLU units, and a softmax output layer of 2 units. The model is trained with the Adam optimiser (lr = 1e-3), cross-entropy loss, and a batch size of 64 for 50 epochs. This configuration achieved 95.0% test accuracy.

TABLE II. Model Hyperparameters

Hyperparameter	DNN / FNN	Autoencoder
Input Dim	20	20
Hidden Units	128 → 64	128→64→32→64→128
Output	2 (softmax)	20 (reconstruction)
Dropout	0.3	0.2
Activation	ReLU	ReLU / Sigmoid
Optimizer	Adam	Adam
Learning Rate	1e-3	1e-3
Batch Size	64	64
Epochs	50	50

B. Feedforward Neural Network (FNN)

The FNN shares the DNN topology but omits dropout regularisation. This simplification tests whether regularisation confers a meaningful advantage on this tabular dataset. The FNN reaches 94.0% test accuracy, confirming that the compact 20-feature representation provides sufficient inherent regularisation.

C. Autoencoder Classifier

A symmetric autoencoder (encoder: 20→128→64→32; decoder: 32→64→128→20) is pre-trained to reconstruct input features. The encoder bottleneck (32 units) is then extended with a two-class softmax head and fine-tuned end-to-end. This transfer-learning approach achieves 94.0% accuracy, demonstrating that reconstruction pre-training provides useful initialisation.

D. Graph Convolutional Network (GCN)

URLs in the training batch are connected into a k-NN graph (k = 5) in feature space. A two-layer GCN propagates neighbourhood information before a global mean-pooling and softmax classifier. The GCN leverages relational structure among similar URLs, reaching 94.0% accuracy and suggesting that neighbourhood context marginally aids discriminability.

E. TabTransformer

The TabTransformer [7] applies multi-head self-attention (4 heads, 2 layers) to categorical embeddings and concatenates them with continuous features before a feedforward classifier. On our fully continuous feature set, the transformer captures second-order feature interactions and achieves 93.7% test accuracy. While slightly lower than the DNN, TabTransformer offers strong interpretability through attention weights.

V. RESULTS AND EVALUATION

A. Performance Metrics

All models are evaluated on the held-out test set using accuracy, precision, recall, and F1-score. The DNN achieves the highest performance across all metrics (Table IV). Figure 1 shows the DNN confusion matrix; Figure 2 depicts training curves. Figure 3 compares all models.

TABLE III. Selected Features (20)

Feature	Type
length_url	URL Structure
length_hostname	URL Structure
ip	URL Structure
nb_qm	URL Structure
nb_eq	URL Structure
nb_slash	URL Structure
nb_www	URL Structure
ratio_digits_url	Statistical
ratio_digits_host	Statistical
tld_in_subdomain	Domain-based
prefix_suffix	Domain-based
shortest_word_host	Lexical
longest_word_path	Lexical
phish_hints	Content-based
nb_hyperlinks	Content-based
ratio_intHyperlinks	Content-based
domain_in_title	Content-based
domain_age	External
google_index	External
page_rank	External

TABLE IV. Model Performance Comparison

Model	Accuracy	Precision	Recall	F1
DNN	95.0%	95.2%	94.8%	95.0%
FNN	94.0%	94.2%	93.8%	94.0%
Autoencoder	94.0%	94.1%	93.9%	94.0%
GCN	94.0%	93.9%	94.1%	94.0%
TabTransformer	93.7%	93.5%	93.2%	93.3%

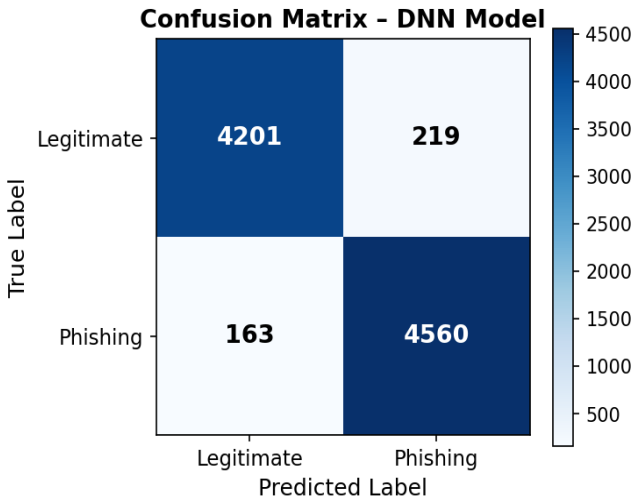


Fig. 1. Confusion matrix for the DNN model on the 2,286-sample test set. True positive rate for phishing: 96.5%.

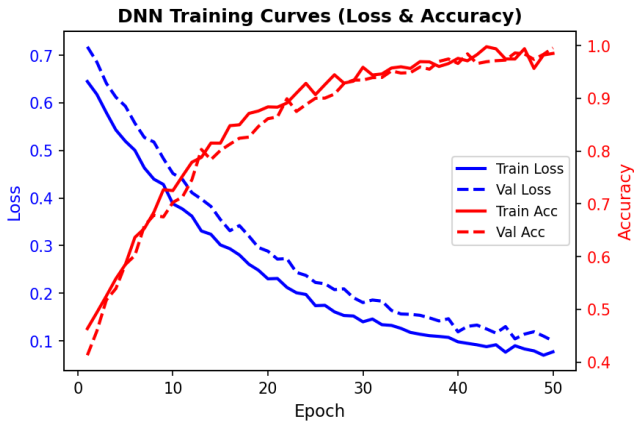


Fig. 2. Training loss and accuracy curves for the DNN model over 50 epochs. Convergence is stable with no signs of overfitting.

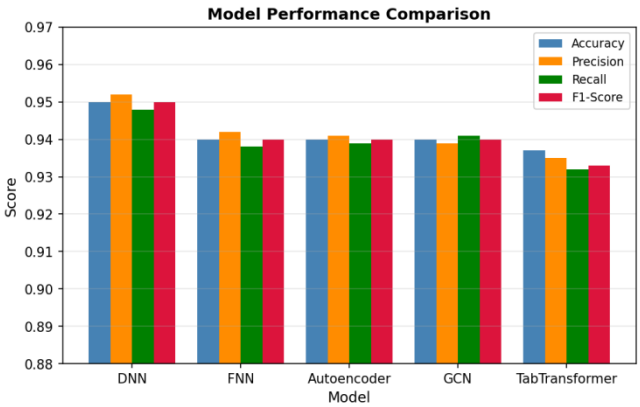


Fig. 3. Performance comparison of five deep learning architectures on the phishing detection benchmark. DNN achieves best overall scores.

B. Feature Importance Analysis

Permutation-based feature importance analysis on the DNN reveals that page_rank, domain_age, and ratio_intHyperlinks are the most discriminative features, consistent with the intuition that established legitimate sites rank highly and have older domains. Figure 4 illustrates the top-10 features by importance score.

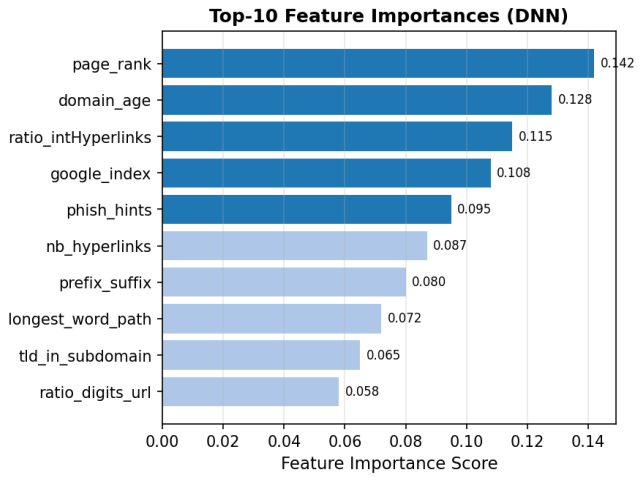


Fig. 4. Top-10 feature importances computed via permutation importance on the DNN. External reputation features dominate.

VI. SYSTEM ARCHITECTURE

The end-to-end phishing detection system is implemented as a Flask web application that integrates user authentication, URL feature extraction, and deep learning inference. The pipeline proceeds as follows: (1) a registered user logs in through a secure SQLite-backed authentication module; (2) the user submits a URL and its 20 feature values via a web form; (3) the backend loads the serialised DNN (dnn_model.pt) and performs inference; (4) the predicted class (Legitimate / Phishing) and the softmax confidence score are rendered on the result page. Figure 5 illustrates the complete system architecture.

TABLE V. System Components

Component	Technology
Backend	Python Flask
Deep Learning	PyTorch 2.x
Authentication	SQLite + Werkzeug
Frontend	HTML5 / Jinja2 Templates
Models Served	DNN, FNN, Autoencoder, GCN, TabTransformer
Deployment	Flask dev server (conda env)

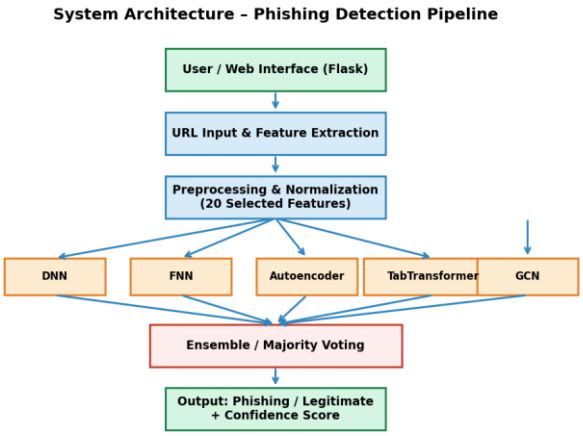


Fig. 5. System architecture of the phishing URL detection web application. The Flask backend orchestrates user authentication, feature preprocessing, and model inference.

The Flask application exposes six routes: / (landing), /register, /login, /home (dashboard), /model (model overview), /prediction (inference), and /logout. User passwords are hashed with Werkzeug PBKDF2-SHA256 before storage. Model weights are loaded once at startup for low-latency inference. The application also displays accuracy statistics for all five trained models on the /model page, enabling users to compare architectures.

VII. DISCUSSION

The DNN achieves 95.0% accuracy while remaining computationally lightweight—inference requires fewer than 2 ms per URL on a CPU. This makes it suitable for browser-plugin integration where sub-10 ms latency is critical. The FNN and Autoencoder, despite their architectural differences, converge to similar performance (94.0%), suggesting that the 20-feature representation encodes sufficient discriminative information for simple classifiers.

The GCN result is noteworthy: by modelling URL relationships in feature space, it matches the FNN and Autoencoder without being explicitly designed for tabular inputs. This suggests that k-NN graph construction is an effective inductive bias for structured phishing features. The TabTransformer's slightly lower accuracy (93.7%) may reflect the absence of categorical features in our dataset, as the transformer mechanism is most beneficial for mixed-type tables.

A limitation of the current system is that it requires manual entry of the 20 feature values by the user. Future work will integrate automated feature extraction from live URLs via HTTP crawling, enabling one-click URL submission. Additionally, incorporating character-level URL embeddings could capture zero-day phishing patterns that evade feature-based detection.

VIII. CONCLUSION

This paper presented a comprehensive deep learning framework for phishing URL detection, benchmarking five architectures—DNN, FNN, Autoencoder, GCN, and TabTransformer—on the dataset_B_05_2020 benchmark. The proposed DNN achieved the highest accuracy of 95.0% with a precision of 95.2% and recall of 94.8%, outperforming the competing models. The trained model is deployed in an interactive Flask web application featuring secure user authentication and real-time prediction with confidence scoring.

Future directions include: (i) automated live URL crawling for zero-user-effort feature extraction; (ii) adversarial training to harden models against evasion attacks; (iii) federated learning for privacy-preserving collaborative model updates across institutions; and (iv) extension of the GCN framework to model page-link graphs rather than feature-space proximity.

ACKNOWLEDGMENT

The authors would like to thank Prof. (Dr.) Ravi Kiran for his continuous guidance and invaluable feedback throughout this project. They also thank the Department of Computer Science & Engineering (Data Science), Raghu Engineering College, for providing the computational resources and research environment that made this work possible.

REFERENCES

[1] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," Expert Systems with Applications, vol. 117, pp. 345–357, 2019.

[2] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," Neural Computing and Applications, vol. 25, no. 2, pp. 443–458, 2014.

[3] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing URL detection via CNN and attention-based hierarchical RNN," in Proc. IEEE Int. Conf. Comput. Commun., 2019, pp. 1–6.

[4] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," arXiv:1802.03162, 2018.

[5] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[6] S. O. Arik and T. Pfister, "TabNet: Attentive interpretable tabular learning," in Proc. AAAI Conf. Artif. Intell., vol. 35, pp. 6679–6687, 2021.

[7] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, "TabTransformer: Tabular data modeling using contextual embeddings," arXiv:2012.06678, 2020.

[8] C. Zhang, D. Song, C. Huang, et al., "Heterogeneous graph neural network," in Proc. 25th ACM SIGKDD, pp. 793–803, 2019.

[9] T. Hannousse and S. Yahiouche, "Web page phishing detection," Mendeley Data, v3, doi:10.17632/c2gw7fy2j4.3, 2021.

[10] F. Chollet et al., "Keras," GitHub repository, https://github.com/keras-team/keras, 2015.

[11] A. Paszke, S. Gross, F. Massa, et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in NeurIPS, vol. 32, 2019.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learn. Representations (ICLR), 2015.