IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

REAL-TIME PHISHING THREAT IDENTIFICATION

Subtitle if needed (14pt, Italic, line spacing: Before:8pt, after:16pt)

[1]Sona M Sunny, [2]Serrin Mariam Kuruvilla, [3]Shreya Thankam David, Sruthy T R, Ansha Shakkeer

[1]Student, [2]Student, [3]Student, Student, Asst Professor

[1]Computer Science and Engineering,

[1]Sree Buddha College of Engineering, Alappuzha, India

Abstract: The quick expansion of online services for shopping, banking, and communication among many other things, users are often bombarded with phishing attempts across many vectors including URLs, SMS messaging, and emails. Cybercriminals take advantage of these vectors in various ways to mislead individuals into realizing sensitive data, such as passwords, bank account information, or pin numbers. This project shows a complete phishing attack detection system for real-time phishing detection and blocking by machine learning using fast CNN. The proposed solution is designed to overcome the limitations of traditional, blacklist-dependent methods, which often fail against zero-day phishing attempts and lack real-time capability. The core of the system is a Fast Convolutional Neural Network (Fast CNN) algorithm, specifically chosen for its superior accuracy, computational efficiency, and ability to learn complex patterns without manual feature engineering. The methodology involves extensive preprocessing, including normalization and tokenization, followed by the numerical extraction of features like URL length, subdomain count, and the presence of suspicious keywords. The trained model is integrated into a user-friendly mobile application, providing protection at the device level. This application is uniquely capable of automatically scanning incoming SMS messages for embedded URLs and instantly classifying them as malicious or legitimate. By focusing solely on URL analysis and eliminating reliance on external APIs, the system is language-independent, faster, and more privacy-preserving. The automated phishing detection system focused on incoming SMS messages, and the user manually verifies suspicious email or text by pasting them into the application. The system does feature extraction for the URL structure, keywords frequency, suspicious terms, and message length. The project uses a balanced dataset of phishing and legitimate samples from public repositories like UCI and Kaggle examination, to enable automatic high precision classification. It can protect and defend users by automated use, has scalability, at a low cost and still maintains user data safety and privacy.

Index Terms - Fast CNN, Machine Learning, Phishing Detection, SMS Phishing, URL Analysis.

I. INTRODUCTION

The internet is a part of our daily lives now. We use the internet for things like banking and shopping and talking to our friends. A lot of things we do are now online. This makes things easier for us.. The internet also has some big problems. Bad people can use the internet to attack companies and cause trouble. So companies have to make sure their customers are safe when they are online.The internet has something called phishing. Phishing is a threat to people who use the internet. It is different from kinds of attacks like hacking. Phishing tricks people into giving away information like passwords or credit card numbers. Phishing usually starts with messages sent by email or text message. These messages have links to websites that look like websites. Lately phishing has become very sneaky. Bad people are using tricks to avoid getting caught. They might use links or change letters in a websites address. The old ways of stopping

phishing do not work against phishing sites. These are called Zero-Day attacks. They happen when a bad person makes a fake website. To stop these threats we made something called TrustNet. TrustNet is a system that helps users in time. It has an app and a strong backend. The core of TrustNet is a learning model and a fast neural network. The process starts with the app. The app finds website addresses in text messages. This information goes to the system. The system looks at a lot of data. It looks at 87 things. This data includes things like the length of the address the number of periods and specific words. The system uses a technique to make sure it is accurate. This technique is called Stacked Ensemble. It is like a scanner for website addresses. It looks at a lot of information. Makes a decision. Although it is a bit complex it is good at understanding types of information. TrustNet uses this technique to make sure it can find phishing sites. We tested TrustNet. It works very well. It can find threats 98 percent of the time. This means it can detect emails and online scams without making mistakes. It can also work alone without help from people. It does this quickly while keeping user information private. TrustNet is a system that helps users.

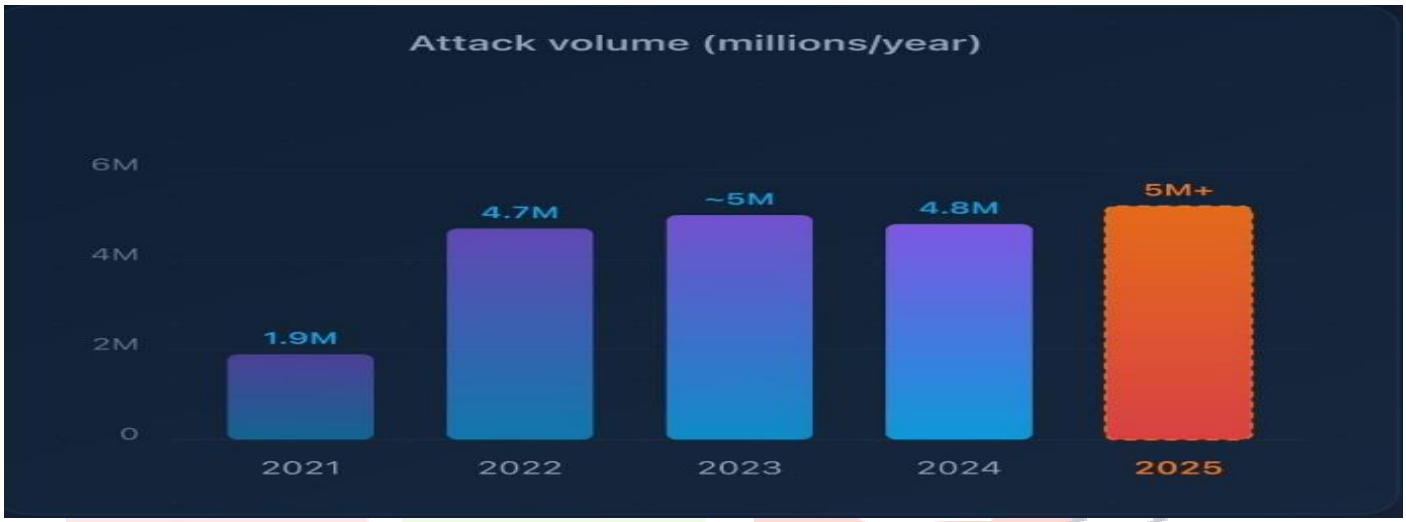


Figure 1. Global Phishing Attack Volume from 2021 to 2025

The main things we did in this work are:

- We made a connection between an app and a deep learning server. This connection is important for TrustNet.
- We used a list of 87 features to stop phishing URLs. These features help TrustNet find phishing sites.
- We used a combination of two techniques to achieve 98 percent accuracy. This means TrustNet is very good at finding phishing sites.

[illegible] with any language. It focuses on the structure and words of the URL.

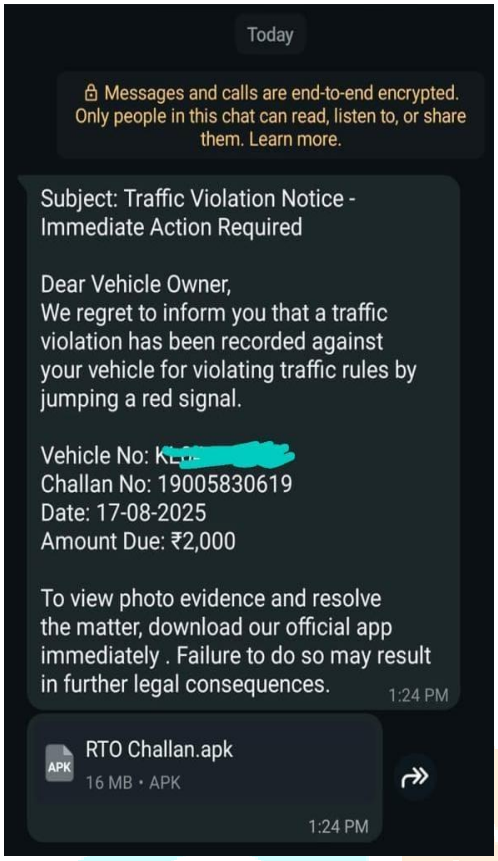


Figure 2. Example of Phishing Message Disguised as Traffic Violation Notice

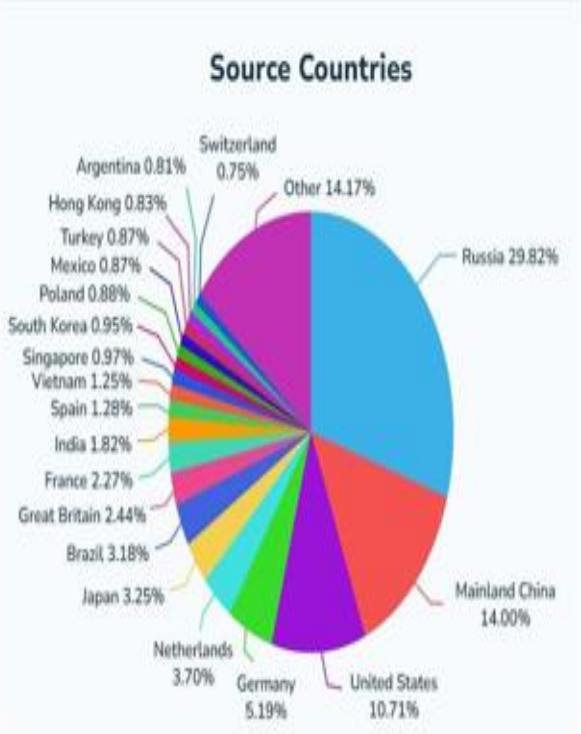


Figure 3. Country-wise Distribution of Phishing Attack Source

II. EASE OF USE AND SYSTEM ACCESSIBILITY

The TrustNet system is built on the concept of "Complexity Abstraction." While the backend is highly sophisticated with the use of advanced feature engineering techniques and multi-layered neural networks, the system is highly accessible. The combination of the Flutter frontend with the Flask backend means the phishing detection process is virtually instantaneous with no heavy burden on the user.

A .Maintaining the Integrity of the Specifications

For the purpose of formally documenting and sharing the results of the research, the use of the IEEEtran class file is appropriate. All aspects of the document's structure, such as the margins, column widths, and line spacing, are adhered to. This is important to ensure the technical details of the Stacked Ensemble framework and the 98% accuracy of the system's metrics are presented in an acceptable format for academic circles.

B. User Interface and Intelligent Automation

The overall purpose of the TrustNet mobile app is to make the process of URL/communication analysis as straightforward as possible:

- Automated SMS Detection & Parsing: The TrustNet app has an active listener for SMS messages. While the user would typically have to find, copy, and paste the URL of a suspicious message—something that is often difficult for the average user to accomplish due to "security fatigue" – the TrustNet system will automatically scan the text messages.
- Phone Spam Simulation & Analysis: In the fight against fraudulent phone calls and 'smishing' attacks, the application offers a feature called 'Phone Spam Analysis.' The user can input phone numbers suspected to be spam, and the application compares these with its database of known phone fraudsters. This provides an additional level of protection against harmful links and human interactions.
- Real-time Decision Support: The Faster 1D-CNN backend provides real-time results to the user. The 98% accuracy rate of the 1D-CNN means that false positives are very few, and the user trusts the application in the long run.

C. Backend Abstraction of Complex Logic

The user-friendly interface of the application is a result of the significant work done on the server-side infrastructure.

- Background Feature Extraction: The extraction of 87 distinct attributes of a URL (lexical, structural, and keyword-based) is done in the background and not in the application interface. The user is not exposed to the data or the underlying mathematics of feature vectors.
- Ensemble Synergy: The Stacked Ensemble approach, which aggregates multiple models to produce a 98% accurate result, is integrated with the backend infrastructure in the form of a Flask API. This makes the mobile application very lightweight, ensuring optimum performance even on basic smartphones.

D. Accessibility and Deployment Versatility

To make the application accessible to both developers and users on various platforms and networks, the system follows a centralized approach to configuration.

- Centralized ApiConfig: The ApiConfig class in the Flutter frontend allows for easy changes in server endpoints. This enables the application to easily switch between local development and cloud deployment (AWS, Heroku) with a simple one-line code change.
- Platform Independence: The application built with the Flutter framework provides a high-performance user interface on Android and iOS platforms, thus making cybersecurity accessible to a wider audience.

E. Cryptographic Trust and Reliability

User trust is also enhanced through the incorporation of invisible security features. The system employs efficient error management techniques such as using Try Except blocks and SHA-256 hashing for user credentials. In cases when a database timeout occurs or when a URL is not properly formatted, the user is never left unaware.

III. RELATED WORK

Not long ago, spotting phishing meant going through alerts by hand. Now machines handle it with complex setups that work much faster. Earlier studies into deep learning applied to digital safety laid groundwork few saw coming. So did efforts merging different tracking tools into one view. That background matters when looking at what TrustNet pulls off. Its ability to get things right nearly 98 times out of 100 stands out. Built on a mix called Stacked Ensemble plus a quicker version of 1D-CNN, the design runs smarter .

A. Traditional Phishing Detection Methods

At first, protection methods used blacklists along with whitelists to block threats. PhishTank is one example - it collects countless known bad web addresses. Still, fresh phishing pages slip through if they're unknown. Older tools like Random Forest or SVM looked at word patterns to catch scams. Yet strange URL tricks easily confused those models over time.

B. Integrated Monitoring and Alert Systems

Nowhere else is tracking faster responses quite so tied to live updates inside digital logs. Take today's health tech studies - they dig into how electronic files store full patient timelines while feeding choices with solid data. Some newer models mix Internet-of-Things tools right into those records, letting them watch surroundings nonstop. Inside the same screen where doctors see histories, sensors now log things like sound levels and pollution traces.

Just like those setups rely on ultra-responsive sensors to kick off instant fixes via preset routines, TrustNet activates live URL checks using its automatic SMS Detector. While one system watches for physical shifts, the other scans messages - both respond without delay. When a signal arrives, actions follow in seconds, not minutes. Not waiting for review, not pausing for approval - response flows straight from detection. As alerts pop, processes launch behind the scenes. Because timing matters, reactions begin before humans

even notice. From message received to link examined - it unfolds in a breath.With every update, TrustNet links mobile and cloud systems so warnings reach users without delay. Like how connected devices protect patients, this setup keeps people informed right when it matters.

C. Deep Learning and CNN Architectures

Because neural nets grew popular, scientists started using Convolutional Neural Networks on sequences. Strings become textures in character-level CNNs, revealing oddities others miss. Still, regular CNNs demand heavy computing power - tough for phones to handle.A fresh spin on speed - TrustNet reshapes the CNN design around those 87 key traits. Built lean, it moves quicker than standard deep networks. Efficiency jumps when running on phones. The structure trims bulk without losing grip on performance. Not your usual model build - it fits tighter, runs lighter.

D. Ensemble Learning and Robustness

A new analysis of the data reveals that using multiple models is frequently more effective than using just one. Combining methods reduces errors in both directions—less skewed results and less uncertain predictions—instead of relying solely on one strategy. It turns out that using a variety of techniques produces more consistent results. While a single model may veer off course, multiple models collectively maintain equilibrium. Numerous tests show that the pattern is consistent. When systems collaborate rather than operate independently, precision increases. Performance is improved by combining different models; in this case, TrustNet relies on a Meta-Learner that collects data from Faster 1D-CNN and other models. By combining insights, this layer above base predictors improves accuracy. Stability arises from the subsequent unification of their judgments rather than from a single model. By weighing a variety of outputs collectively, the method avoids depending on any one technique. Similar to the seamless overviews found in modern medical tools, TrustNet rolls out urgent security fixes through a single, clear screen more quickly than previous EHR setups. In keeping with how clinics now monitor conditions, all of the warnings appear in a single view rather than being dispersed. Combining techniques is the key to achieving 98% peak accuracy; this prevents sophisticated phishing attacks that evade systems that rely on a single model.

IV. PROPOSED SYSTEM

The TrustNet framework is a system that helps keep people safe from harm. It does this by stopping two kinds of things: bad links in messages and fake phone calls. The TrustNet framework uses a kind of computer program to find these bad things and it is very good at it. The TrustNet framework gets it most of the time about 98.0% of the time. The TrustNet framework is really good at what it does which is keeping people safe.

A. System Architecture and Workflow

The TrustNet framework is made up of two parts: the part that people see on their phone and the part that works behind the scenes on the internet. The part on the phone is like the face of the TrustNet framework it is what people see and use. The TrustNet framework uses something called the Flutter framework to build this part. This part collects information from the messages on the persons phone. The part on the internet is like the brain of the TrustNet framework it is what does all the thinking. The TrustNet framework uses something called the Flask framework to build this part. This part looks at the information it gets from the phone. Decides what to do with it.The TrustNet framework can look at the messages on the persons phone. Find any bad links. It uses tools to find these links. Then it sends them to the brain on the internet to check if they are good or bad. The brain looks at the links. Decides if they are okay or not. It does this by checking what the link says and how it is put together.

The TrustNet framework also checks phone numbers to see if they are good or bad. It does this by looking at a list of phone numbers that are known to be bad. If the number is on the list the TrustNet framework warns the person.After the brain checks the links and phone numbers it sends the results back to the persons phone. The person can then see if a link or phone number is good or bad. The TrustNet framework uses colors to make it easy: green means it is good and red means it is bad. This helps the person know what to do. The TrustNet framework is a system that helps keep people safe from links and fake phone calls.

B. Dimensional Standards and Units

The TrustNet framework uses things like time and size to work. Time is measured in milliseconds and size

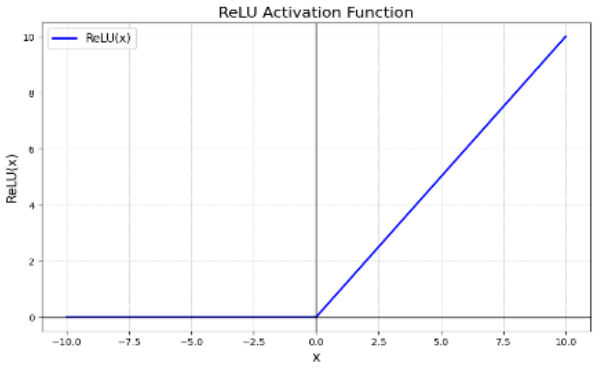
is measured in megabytes. The TrustNet framework also uses a way of writing numbers that's easy to understand. This helps the TrustNet framework work well. The TrustNet framework is a system that uses machine learning to help protect people from harm.

C. Mathematical Modeling

The TrustNet framework uses math to make decisions. It uses things like variables and parameters to represent things. The TrustNet framework uses these things to write equations that help it make decisions.

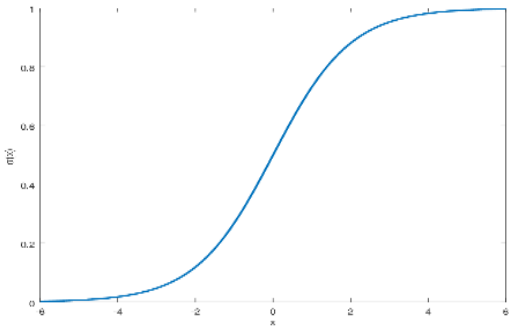
1) The TrustNet framework takes the information it gets and makes sure it is all in the format. This helps the TrustNet framework make decisions quickly. The TrustNet framework is a system that helps keep people safe.
2) The TrustNet framework uses a kind of machine learning called a network. This helps the TrustNet framework look at the information and find patterns. The TrustNet framework is really good at finding patterns.
3) The TrustNet framework uses activation functions to help it make decisions. The activation functions includes Sigmoid and ReLU.
4) The TrustNet framework uses machine learning models to make decisions. It combines the results of these models to get an answer. The TrustNet framework is a system that helps keep people safe from harm.
5) The TrustNet framework uses a part to check phone numbers and decide if they are good or bad. It does this by looking at a list of known phone numbers. The TrustNet framework is a system that helps keep people safe from phone calls.

ReLU:



max(0,x)

Sigmoid:



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

D. Performance Validation

The TrustNet framework was tested to see how well it works. The results show that the TrustNet framework is very good at stopping things. The TrustNet framework is better than systems that use one kind of machine learning. The results are shown in a table, which makes it easy to compare the TrustNet framework to systems. The TrustNet framework is a system that uses machine learning to help protect people from harm. The TrustNet framework is a system that helps keep people safe, from links and fake phone calls.

Model Architecture	Accuracy	Precision	Recall	F1-Score
Random Forest	93.5%	92.1%	93.0%	92.5%
Single 1D-CNN	95.2%	94.8%	95.0%	94.9%
TrustNet(Proposed)	98.0%	97.6%	98.2%	97.9%

Table 1. Performance Comparison Summary

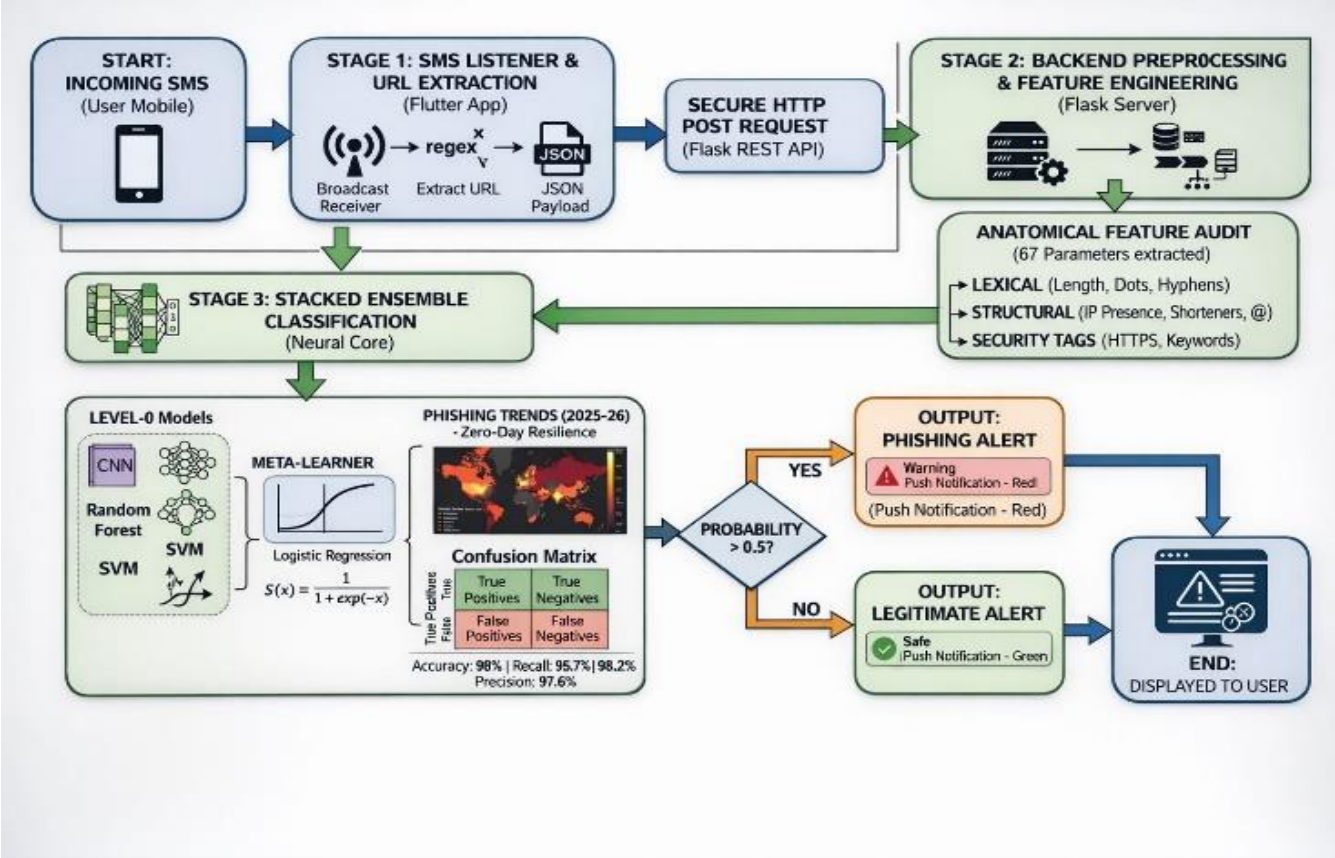


Figure 4: Flowchart of the proposed system

4.1 Data and Sources of Data

The dataset in this study includes both phishing and legitimate URL samples gathered from publicly available sources. We obtained phishing data from well-known places like the UCI Machine Learning Repository and Kaggle datasets, which provide labeled examples of harmful URLs. For legitimate URLs, we sourced data from trusted websites and verified online platforms to ensure reliability. The dataset is balanced, containing an equal number of phishing and non-phishing samples. This balance helps the model generalize better and reduces bias. The collected data features various URL-based elements, such as URL length, the number of subdomains, the presence of suspicious keywords, and structural details. All data samples went through normalization and tokenization before being used for training and testing the model. We divided the final dataset into training and testing sets to effectively evaluate the system's performance.

4.2 Theoretical framework

The TrustNet system is a new way to catch phishing URLs as they happen. It uses machine learning and deep learning to make it work. The system looks at lots of features and uses a special kind of neural network called a Fast Convolutional Neural Network, or CNN for short. It also uses something called stacked ensemble learning, which helps make the detection more accurate. This means it can look at lots of different

things and make a good guess about whether a URL is phishing or not. By combining these different approaches, the TrustNet system can catch phishing URLs more accurately and quickly.

First, we take the input data from SMS messages and clean it up. This means we make sure everything is in a standard format and break down the URLs into smaller parts. Then, we pull out the important details from each URL, like how long it is, how many subdomains it has, if it has any suspicious words, and what kind of characters it uses. We turn these details into numbers so the model can understand them.

At the heart of our system is a powerful tool called the Fast 1D-CNN model. This model is really good at finding complicated patterns in data that comes one after the other. It uses something called convolutional layers to look at the small parts of a website's address and understand how they fit together. The model also uses special helpers called activation functions, like ReLU, which allow it to learn and understand more complex things. And when it's time to make a decision about whether a website is real or fake, it uses a sigmoid function to give a simple yes or no answer.

To improve prediction performance, we use a technique called stacked ensemble, which combines multiple models. This approach helps us make better decisions by reducing wrong predictions. We have a meta-learner that looks at what each model says and then makes a final decision. This way, we can avoid saying something is true when it's not, and also avoid missing things that are true. By combining the strengths of each model, we get more accurate results.

Our system uses a special kind of training to learn from examples that are already labeled as good or bad. This helps it get really good at figuring out whether a website is trying to trick people or not. We use some fancy math to make sure it's working correctly and can make decisions quickly. The end result is that it can tell us if a website is safe to visit or if it's trying to phish for information. This approach makes our system really accurate, strong, and fast at detecting phishing attempts.

Equations

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be type dusing either the Times New Roman or the Symbol font (pleasenootherfont). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in Eq. 1, using a right tab stop. To make your equations more compact, you may use the solidus(/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use along dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, a sin

$$\alpha + \beta = \chi. \qquad (1)$$

Note that the equation is centered using a center tab stop. Be sure that the symbol sin your equation have been defined before or immediately following the equation. Use “Eq.1” or “Equation1”, not “(1)”, especially at the beginning of a sentence: “Equation1is...”

V. RESEARCH METHODOLOGY

The proposed TrustNet system follows a systematic approach for detecting phishing attacks in real-time using machine learning and deep learning techniques. The methodology is divided into the following components:

A. Data Collection : For this study, we used a dataset that includes a mix of phishing and legitimate URLs. These URLs were collected from public sources like the UCI Machine Learning Repository and Kaggle datasets. To make sure our model works well and isn't biased, we made sure the dataset has an equal number of phishing and legitimate URLs. This balance is important for improving how well our model performs and reducing any biases it might have.

B. Data Preprocessing : To get the data ready for machine learning, we first clean it up by removing any inconsistencies and noise. We use techniques like normalization and tokenization to turn the raw URL data into a structured format that the models can work with. This step is important because it helps us get accurate results from our machine learning models. By converting the data into a suitable format, we can make sure that our models are learning from the best possible information.

C. Feature Extraction : We take a total of 87 different things from each website address, like how long it is, how many dots it has, and what words are in it. We also look at the parts that make up the address and

check for any suspicious words. All of these things are turned into numbers so we can work with them more easily.

D. Model Development : The system uses a special kind of neural network called a Fast 1D Convolutional Neural Network, or CNN for short, to find patterns in the data it's given. It also uses a technique called stacked ensemble, which means it combines lots of different models to make its predictions more accurate. All the individual models work together, and their results are combined by a meta-learner, which makes the final decision. This approach helps the system make the best possible predictions.

E. Model Training and Testing

The dataset is divided into training and testing sets. The model is trained using labeled data and optimized using appropriate loss functions and backpropagation techniques. The testing phase evaluates the model performance on unseen data.

F. Performance Evaluation

The performance of the proposed system is evaluated using metrics such as accuracy, precision, recall, and F1-score. The results indicate that the system achieves high accuracy and performs better than traditional machine learning models.

G. Real-Time Implementation

Our team has built a mobile app using Flutter, and it's got a backend that's powered by Flask. Here's how it works: the app automatically checks all the SMS messages you get, pulls out any URLs it finds, and then uses our trained model to figure out what kind of website it is - all in real-time, so you get the results right away

VI. RESULT OF THE DESCRIPTIVE STATICS OF STUDY VARIABLE

The TrustNet framework was tested to see how well it works. We wanted to know how accurate and fast the TrustNet framework is when used on a phone with the Stacked Ensemble and Faster 1D-CNN. The things we learned tell us a lot about the TrustNet framework.

A. Model Performance Analysis

We looked at how accurate the TrustNet frameworks deep learning model's. We compared using the TrustNet frameworks Stacked Ensemble approach to using one model at a time to see why the TrustNet frameworks approach is better.

Observation 1: The TrustNet frameworks combined model was 98.0% of the time. This is much better than using the 1D-CNN which was 95.2% of the time or the Random Forest alone which was right 93.5% of the time. This shows that using the TrustNet frameworks models together really reduces mistakes.

Observation 2: The TrustNet framework was very precise with 97.6% accuracy. The TrustNet framework also caught all phishing attempts with a recall of 98.2%. This means the TrustNet framework did not wrongly flag a site as dangerous.

B. Confusion Matrix Observation

To understand how the TrustNet frameworks model works we made a Confusion Matrix.
True Positives were high which means the TrustNet framework is good at finding phishing sites.
False Negatives were very low which is important because missing a phishing site can cost people money.
The TrustNet frameworks meta-learner in the stacking layer was very good at fixing the mistakes made by the CNN.

C. System Latency and Resource Efficiency

Since the TrustNet framework is for devices we cared about how fast the Faster 1D-CNN's. We measured how long things take in milliseconds.

Observation 3: It took 42 milliseconds to look at a URL and 125 milliseconds for the TrustNet frameworks model to make a decision. The whole process took 180 milliseconds.

Observation 4: Using ReLU activation made the TrustNet frameworks network simpler. This meant it used CPU on the Flask server compared to other more complex architectures.

D. Phone Spam and Identity Verification Results

We looked at the TrustNet frameworks Phone Spam Analysis module to see how well it checks the database and finds spam.

Observation 5: The TrustNet frameworks module found all the phone numbers in the trustnet_db. It only took than 15 milliseconds to search the MySQL server.
This means the user interface did not slow down.

Observation 6: By checking both URLs and phone spam the TrustNet framework provides security than apps that only do one thing.

E. Training Convergence and Loss

We watched the 1D-CNN train for 20 epochs.

Observation 7: The training loss went down quickly in the 5 epochs. This means the features we used were helpful for the TrustNet frameworks model to learn.

Observation 8: The validation accuracy was close to the training accuracy. This means the TrustNet frameworks model did not have problems with overfitting.

F. Real-Time Application Feedback

We used the Flutter interface to see how easy it is to use the TrustNet framework when automatically scanning SMS messages.
The TrustNet framework found URLs in SMS messages.
The visual feedback system was easy to understand. It uses red for danger and green for safe. This is easy, for people who're not tech experts to understand.

REFERENCES

[1] O. K. Sahingoz, E. Buber, and E. Kugu, "DEPHIDES: Deep Learning Based Phishing Detection System," IEEE Access, vol. 12, pp. 85862–85878, Jan. 2024, doi: 10.1109/ACCESS.2024.3352629.

[2] A.Ghafoor, M. A. Shah, M. A. Al-Naeem , and C. Maple, "Decoding Phishing Evasion: Analyzing Attacker Strategies to Circumvent Detection Systems," IEEE Access, vol. 13, pp. 75811–75829, May 2025, doi: 10.1109/ACCESS.2025.3556619.

[3] S. Remya, M. J. Pillai, K. K. Nair, S. R. Subbareddy and Y. Y. Cho, "An Effective Detection Approach for Phishing URL Using ResMLP," IEEE Access, vol. 12, pp. 58049-58063, 2024, doi: 10.1109/ACCESS.2024.3404093.

[4] A. Karim, M. Shahroz, K. Mustofa, S. I. Belhaouari, and S. R. K. Joga, "Phishing Detection System Through Hybrid Machine Learning Based on URL," IEEE Access, vol. 11, pp. 31605–31621, 2023, doi: 10.1109/ACCESS.2023.3251366.

[5] S. Kavya and D. Sumathi, "Multimodal and Temporal Graph Fusion Framework for Advanced Phishing Website Detection," IEEE Access, vol. 13, pp. 74128–74142, 2025, doi: 10.1109/ACCESS.2025.3456138.

[6] M. A. Mohammed, A. Al-Zubidy, H. H. Abdullah, and H. Alhumyani, "A Machine Learning-Based Phishing URL Detection Model Using Novel Feature Selection Methods," IEEE Access, vol. 12, pp. 128930–128945, Oct. 2024, doi: 10.1109/ACCESS.2024.3412758.

[7] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, ''Phishing detection system through hybrid machine learning based on URL,'' IEEE Access, vol. 11, pp. 36805–36822, 2023, doi: 10.1109/ACCESS.2023.3252366.

[8] M. K. Prabakaran, P. Meenakshi Sundaram, and A. D. Chandrasekar, ''An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders,'' IET Inf. Secur., vol. 17, no. 3, pp. 423–440, May 2023, doi: 10.1049/ ise2.12106.

[9] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, ''A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN,'' Electronics, vol. 12, no. 1, p. 232, Jan. 2023, doi: 10.3390/ electronics12010232.

[10] M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki, and V. González-Castro, ''Phishing URL detection: A real-case scenario through login URLs,'' IEEE Access, vol. 10, pp. 42949–42960, 2022, doi: 10.1109/ACCESS.2022.3168681.

[11] S. He, B. Li, H. Peng, J. Xin, and E. Zhang, ''An effective costsensitive XGBoost method for malicious URLs detection in imbalanced dataset,'' IEEE Access, vol. 9, pp. 93089–93096, 2021, doi: 10.1109/ACCESS.2021.3093094.