MARMARA UNIVERSITY
FACULTY OF TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
BLM3053 INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS
FINAL PROJECT REPORT

STUDENT ID: 171421005
NAME LASTNAME: Muhammed Yasin ÖZDEMİR

STUDENT ID: 100220027
NAME LASTNAME: Eren DOĞAN

PROJECT TITLE: Phishing Attacks URL Detection.

Prof. Dr. Serhat ÖZEKES
Res. Asst. Abdulsamet AKTAŞ

1. INTRODUCTION

In the history of mankind, there have been important technological inventions that have affected the life of humanity. While most of these technological inventions have met the needs of people in certain areas and provided benefits in certain areas, information technologies have affected people's lives in almost every field. Information technologies have started to affect people's lives depending on the information produced and the speed of access to this information, and this technology; With its use in production, industry, education and health, there has been a great change in society at the degree of abyss. Today, information technologies are one of the most important parts of human life. As information technologies are constantly changing, the devices used in information technologies are changing. As the closest examples to the present day; computers, phones, tablets, cameras, modems.

It is possible to come across information technologies almost everywhere in our daily lives. Even if the above technological devices are not used; We are intertwined with information technologies in our daily lives in many events such as getting a queue number to withdraw money, waiting for a traffic light to cross the road, paying your bill, etc. With the Household Information Technologies (IT) Usage Survey (Turkish Statistical Institute [TurkStat], 2023) published by the Turkish Statistical Institute (TUIK) on August 29, 2023, it is seen that the proportion of households with internet access in Turkey as of 2023 was measured as 95.5%. While this rate was 94.1% in the previous year, it was measured as 83.8% in 2018. With an increase of 11.7% in five years, a significant part of the society has gained access to the internet, and with a rate of 95.5%, it is seen that a significant part of the society in general can access the internet. In other results of the study, the rate of internet usage in individuals between the ages of 16-74 was observed as 87.1%. While this rate was stated as 85.0% in the previous year. In 2018, the internet usage rate was stated as 72.9%. According to these rates, it can be seen that the rate of internet usage has increased by 14.2% in five years. The meaning of this increase is that in addition to the increase in the possibility of accessing the internet, it has been observed that individuals who have access to the internet but do not use the internet have started to use the internet. According to the findings of the research, it is seen that the use of the internet is increasing day by day in Turkey and people tend to use the internet and the opportunities offered by the internet more [1].

The widespread use of the Internet can cause some negative consequences as well as the positive results of accessing all kinds of information so easily and quickly. Various security problems are at the forefront of these negative consequences [2]. Some of the most important security problems are; violation of personal information security and malware. The emergence of these security problems has also revealed many security risks for both users and the security institutions of nation states [3].

The number of attacks on personal information security in Turkey is increasing day by day and is becoming more dangerous. One of the most used attacks in the breach of personal information security is phishing attack. Phishing attacks represent a type of fraud carried out by using the possibilities offered by information technologies in order to gain access to personal and confidential data and to use this data maliciously [4]. When a user connects to a legitimate site, they can unfortunately enter their personal information without realizing that the site is fraudulent. Such fraudsters can gain access to the user's sensitive information and use it for malicious purposes [5]. These types of attacks are often included in the category of social engineering attacks. A social engineering attack involves exploiting weaknesses in people's nature, such as the need to trust, haste, curiosity, or fear, in order to gain access to target systems [6].

Phishing attacks usually occur through a malicious link via email, SMS, or social media. When users click on these links, they are redirected to a fake site that looks quite similar to the real site, where once they enter their information, they fall into the hands of attackers. According to the Cyber Security Breaches Survey conducted by the UK public sector information site 2023, the type of attack that businesses are most exposed to is phishing attacks with a rate of 79%. This rate has given similar results with the studies conducted in 2022 and 2021. In other words, phishing attacks have long been the most exposed type of attack for businesses. . The same is true for charities with a rate of 83%. In terms of the general opinion in the society, people are the most malicious software types; While thinking that they may encounter dangers such as viruses, trojans, keyloggers and that these dangers will violate the security of personal information, the rate of attack with malicious software is limited to only 11%. Another common type of cyberattack, denial-of-service attack, is 7%. As a result of the comparison with other types of attacks, the rate of exposure to phishing attacks is higher than other attacks [7][8][9].

The main purpose of phishing attacks is to persuade the user to perform a certain action (usually such as clicking on a link or launching an application) and as a result of

this action, to obtain their personal information and use it maliciously. Phishing attacks can be carried out using different tactics and methods [10].

With the proliferation of smartphones and mobile internet, mobile devices have now become a popular target for phishing attacks. Mobile users are often less cautious than desktop users, and therefore phishing attacks on mobile devices can have a higher success rate. Due to the limited size of mobile device screens, it is more difficult to see URLs precisely and detect fraudulent sites. This can lead to more phishing attacks by malicious attackers against mobile users. Detecting an attack from a URL address can be a difficult process, even for many experts. Because the attacker can deceive even knowledgeable users by using different techniques. For this reason, it is of great importance to get software support for the detection of phishing attacks [5].

Artificial neural networks have become an effective tool in solving complex problems in recent years thanks to advances in deep learning and artificial intelligence. Artificial neural networks are very effective in detecting such security threats thanks to their ability to analyze data in depth and their capacity to learn complex features.

This study aims to develop a model that detects whether the URLs accessed on mobile devices belong to a phishing attack by using artificial neural networks technology and to develop a mobile application where this model can be used. This detection process is based on an artificial neural network model trained with datasets of phishing and legitimate URLs that have been pre-collected and tagged. The URL data received from the mobile application will be analyzed by the artificial neural network model located on a server, and the result will be transmitted back to the mobile application.

This mobile application brings the artificial neural network model to a practical application area. Thanks to this application, users can provide a more secure internet access via their mobile devices and get a more effective protection mechanism against phishing attacks.

With this study, we aim to contribute to the prevention of phishing attacks carried out through mobile applications and to provide stronger protection in the field of cyber security.

2. Literature Review

Phishing attacks are a type of cybercrime that is carried out using fake websites or emails designed to steal internet users' personal information, financial data, or system resources. These attacks threaten the security and privacy of individuals, institutions and governments, cause material and moral damage, and become a major problem of internet security [6]. The purpose of this section is to examine and evaluate the various methods used to detect and prevent phishing attacks through a literature review.

Published articles, theses, reports and books were examined for the literature review. Google Scholar, IEEE Xplore, ACM, Science Direct, Springer Link, DergiPark, YÖK Tez, Academia were used as databases. "phishing", "phishing detection", "Phishing URL", "URL detection" and their Turkish equivalents were selected as keywords. As search criteria, the studies should be related to phishing attacks and detection methods, should be written in English or other languages, and should be fully accessible. In this way, a total of 15 studies were included in the literature review.

The literature review divides the methods used to detect phishing attacks into four main categories: Artificial Neural Network Approaches, Machine Learning Based Approaches, Deep Learning Based Approaches, Natural Language Processing Approaches. These categories represent the most used and studied methods in studies. The advantages, disadvantages, performances and application areas of these methods are discussed in detail in the literature review.

2.1. Artificial Neural Network Approaches

In this part of the literature review, Artificial Neural Networks Based Approaches ar2e examined. These approaches use a variety of artificial neural network algorithms to classify malicious URLs. These algorithms include Multilayer Sensor (MLP), Radial Basis Function (RBF), Probabilistic Neural Network (PNN), Generalized Regression Neural Network (GRNN), Adaptive Resonance Theory (ART) and Self-Organizing Map (SOM) [11].

Ricardo Pinto Ferreira et al.'s study "Artificial Neural Network for the Classification of Websites with Phishing Features", published in 2018, investigates the use of Artificial Neural Network (ANN) to classify websites according to the presence of phishing features. In the study, it is stated that an ANN model with a Multilayer Perceptron (MLP)

structure achieves results with a high accuracy rate in solving this complex problem. ANN-MLP, which successfully classified the sites with an accuracy rate of 87.61% during the training phase, reached an accuracy rate of 98.23% during the testing phase. This performance reveals that it offers one of the best results among other studies using different artificial intelligence techniques. These results suggest that ANN-MLP is an effective option for phishing detection and may provide better performance with larger databases in the future [12].

The article titled "Predicting phishing websites based on self-structuring neural network", published in 2014 by Rami M. Mohammad, Fadi Thabtah and Lee McCluskey, introduces an artificial neural network model for the effective detection and prevention of phishing attacks. This model was developed to predict phishing websites using neural networks capable of self-configuration. Noted for its ability to adapt to the fast-changing nature of phishing threats, this model has demonstrated high predictive accuracy during the training and testing phases. The study by Mohammad et al. emphasizes the importance of an up-to-date anti-phishing tool for predicting timely phishing attacks, and especially the importance of improving the performance of the model by constantly updating the training data. In this article, the characteristics that are effective in detecting phishing websites are grouped. These features are extracted automatically without user intervention and developed using a computer-based tool. They have managed to collect and analyze 17 different characteristics that distinguish phishing websites from legitimate ones. Later, they developed a new rule for each feature. After calculating the frequency of each feature, the results showed that the "Request URL" was of the highest importance in detecting phishing websites, as this was presented in all data states. "Field Age" is presented in the case of 2392 data cases. The "HTTPS & SSL" feature generated revenue at a frequency of 92.8%. The least important feature in distinguishing phishing websites is the "Right-Click Disable" feature, which appears only forty times, followed by the "@ symbol in the URL" feature, which accounts for 3.6% of the dataset size [13].

The study titled "DTOF-ANN: an artificial neural network phishing detection model based on decision tree and optimal features", published by Zhu et al. (2020), deals with a method for effectively preventing phishing attacks, which are emerging as an increasing threat in everyday network environments. Phishing attackers use social engineering techniques such as emails and SMS to spoof legitimate URLs, typically masked with illegal URLs, in order to steal users' private information. Artificial neural networks can be

used to detect and prevent such attacks, as these networks have strong learning capabilities from large data sets and can provide high accuracy in data classification. However, duplicate data points in public datasets and negative and redundant features in feature vectors can lead to the problem of overfitting the training of neural networks, making the trained classifier weak when detecting phishing websites. In this context, this article proposes a method called DTOF-ANN (Decision Tree and Optimal Features based Artificial Neural Network) to address this shortcoming. First, the traditional K-medoids clustering algorithm has been improved by the increased selection of starting centers to clean up duplicate data points from the overall datasets. Subsequently, an optimized feature selection algorithm was designed on the basis of a new index "f_Değer", which is defined using the feature evaluation index, decision tree, and local search method. Finally, with the appropriate parameter settings, the optimal structure of the neural network classifier is created and trained with the selected optimal features. Experimental results show that DTOF-ANN outperforms many current methods with a 97.8% success rate. By using traditional methods of data cleansing and optimized feature selection, this model increases its success in detecting phishing websites. In conclusion, DTOF-ANN can be an important tool for security experts and researchers and contribute to studies in this area [14].

Gupta and Singhal's 2017 paper focuses on phishing attacks, a serious threat in the virtual world. These types of attacks are usually carried out through methods such as email, instant messaging, and social media. The article presents a method using artificial neural networks to classify Uniform Resource Locator (URL) addresses into phishing or non-phishing categories. It aims to improve performance by using particle swarm optimization (PSO) for the training of artificial neural networks. The researchers note that this model exhibits a better training performance compared to the Back Propagation Neural Network (BPNN). This study provides an effective method for researchers to detect phishing URLs and emphasizes that the use of PSO provides a cost-effective solution. It also emphasizes the importance of choosing a backpropagation algorithm suitable for artificial neural networks, stating that choosing the right algorithm is the key to success [15].

2.2.Machine Learning-Based Approaches

In the literature review section, Machine Learning Based Approaches are examined secondly. These approaches use a variety of machine learning algorithms to classify malicious URLs. These algorithms include Decision Tree, Random Forest, Support Vector Machine, Naive Bayes, K-Nearest Neighbor, Logistic Regression, and Linear Decomposer Analysis [16].

A study conducted by Şanliöz, Kara et al. in 2017 is a study that evaluates the effectiveness of machine learning methods in detecting web phishing attacks. Şanlıöz and his team compared various machine learning algorithms used to identify web phishing attacks. In the study, experiments were made on a total of 1353 website data using methods such as Classification and Regression Trees (CART), J48 (C4.5) Algorithm, Adaboost, Random Forest (RF) and Neural Networks (NNet); Of these, 702 are classified as harmful, 548 as legitimate, and 103 as suspicious sites. In the classification made on nine features, the prediction success rates of the algorithms were compared and the predictive capabilities of these algorithms were examined by experimental analysis. The findings obtained as a result of the research measured the success of different machine learning techniques and revealed the effectiveness of these techniques. When the accuracy rates are compared, the J48 algorithm has the highest accuracy rate (0.835), while the Classification and Regression Trees algorithm has the lowest accuracy rate (0.772) [17].

A study by Abdelhamid, Thabtah, and Abdel-jaber (2017) takes an in-depth look at the machine learning (ML) techniques used to prevent phishing attacks over the past decade. The researchers analyzed the effectiveness of various ML techniques in phishing detection and stated that knowledge-based approaches are more suitable for the detection of phishing attacks. It has been pointed out that methods such as the Bayesian Network and Support Vector Machines (SVM) can also offer high detection rates [18].

The study titled "Comparison of Machine Learning Methods for the Classification of Phishing Websites" published by Tahir Emre Kalaycı in 2018 deals with methods for detecting fake sites used by identity thieves that aim to steal users' personal information by imitating trusted websites. In order to serve this purpose, a classification study was carried out on the dataset containing 1353 website samples using common machine learning algorithms such as AdaBoost, multilayer perceptron (MLP), support vector machine (SVM), decision tree, k-nearest neighbor (k-NN), Naïve Bayes and random

forest. As a result of extensive tests and evaluations, it was determined that the random forest algorithm performed best among other algorithms, while MLP gave more effective results with the cross-validation method [19].

A study published by Ahammad et al. in 2022 used algorithms such as Random Forests, Decision Trees, Light GBM, Logistic Regression, and Support Vector Machine (SVM) to determine whether URLs are malicious. As a first step, there are processes of extracting features and then applying the model. The dataset contains a total of 3000 URLs, including 1500 malicious URLs and 1500 harmless URLs. The malicious URL data was collected from the open-source service called Phish Tank. Phish Tank is a database that provides collaborative data on phishing activities on the internet in various formats such as csv, json, which are updated hourly. The source of this dataset is the University of New Brunswick, and the number of legitimate URLs in this collection is 35,300. Malicious and harmless URLs are combined in this dataset. The findings obtained as a result of the examination show that the Light GBM algorithm gives better results compared to other algorithms with a rate of 89.5% [20].

This literature review focuses on the article published by Özgür Koray Şahingöz et al. in 2019 under the title "Machine learning based phishing detection from URLs". The article emphasizes that due to the rapid growth of the internet, users' preferences have shifted from traditional shopping to electronic commerce, and cybercriminals try to mislead their victims with techniques such as phishing. Detecting this type of phishing attack is a very challenging problem as it has a semantic-based structure that targets the weak points of computer users. It is stated that the anti-phishing products offered by software companies are based on a variety of techniques, such as blacklisting, heuristics, visual inspection, and machine learning-based approaches. However, these products are not able to prevent all phishing attacks. This article introduces a real-time anti-phishing system using seven different classification algorithms and natural language processing (NLP)-based features. The system has features such as language independence, the use of large amounts of phishing and legitimate data, real-time operation, detection of new websites, independence from third-party services, and the use of feature-rich classifiers. This system demonstrates significant performance by achieving an accuracy rate of 97.98% with the Random Forest algorithm, which only works with NLP-based features to detect phishing URLs [21].

2.3.Deep Learning-Based Approaches

In the continuation of the literature review, Deep Learning Based Approaches are examined thirdly. These approaches use a variety of deep learning algorithms to classify malicious URLs. These algorithms include Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM), Attention Mechanism (AM) and Doc2Vec [22].

This study, published by Benavides et al. (2020), examines the use of deep learning techniques as an effective method of combating phishing attacks, where cyber attackers try to obtain confidential information from users. It has been concluded that the most commonly used methods are deep neural network (DNN) and convolutional neural network (CNN) [23].

Ferdi GÜL's 2021 doctoral dissertation emphasized that the main cause of today's increasing cyber attacks is the human factor, and focused on phishing attacks in particular. The thesis makes it clear that such attacks, which are at the heart of fake sites and deception principles, target people's security awareness. The study presents a new approach to combating phishing attacks using deep learning techniques. The basis of the research is the inference of the characteristics specific to phishing attacks. These features are processed by a deep learning system to detect whether URLs are malicious or not. To assess the reliability of the URLs, the system compared the results with a large number of reputation building sites. The most effective method was investigated by comparing the performances of classifier algorithms such as Random Forest, Long Short Term Memory (LSTM) and Naive Bayes. The research also compared manual feature selection with automatic feature selection when it comes to detecting phishing sites. In the manual selection, 34 attributes were determined based on the URL structure and these properties were divided into four main groups: URL-based, anomaly-based, HTML and Javascript-based, and domain-based properties. The research results have shown that the LSTM algorithm achieves a high accuracy rate of 95.47% on this feature set. These findings reveal the potential of deep learning-based systems in the fight against phishing attacks [24].

Korkmaz (2023) proposes a hybrid phishing detection system in which URL and web page content are analyzed using deep learning techniques, as well as traditional methods. This system first detects suspicious phishing attempts with URL analysis and then refines

these detections with content analysis. Experimental studies have shown that the proposed two-stage hybrid system is 70.23% more effective than existing URL-based systems, especially in situations that allow content analysis. These findings reveal that it is appropriate to use the combination of Enhanced Deep Neural Network and Convolutional Neural Network (GDNN-CNN) in the URL analysis phase and the Deep Neural Network (DNN) model in the content analysis phase in the system developed by Korkmaz. The article also presents a large dataset of 14,782,355 data points, as well as a subset of 113,189 high-risk legitimate and phishing URLs, and another subset of 45,631 legitimate and 36,123 phishing URLs and content. The development of these datasets is an important contribution to the literature, especially given the paucity of datasets used for content analysis. In conclusion, this thesis expands on existing studies in the literature that aim to improve accuracy rates in phishing detection and develop more effective defense mechanisms in real-world conditions. The hybrid detection approach offers a more comprehensive and dynamic defense strategy in the field of cyber security, with the integration of systems that perform both URL and content analysis [25].

2.4.Natural Language Processing Approaches

In this section of the literature review, Natural Language Processing-Based Approaches are examined. These approaches use a variety of natural language processing algorithms to classify malicious URLs. These algorithms include methods such as Text Classification, Text Clustering, Text Summarization, Text Mining, Text Generation, and Text Translation [26].

In 2017, a study titled "Detecting phishing attacks from URL by using NLP techniques" by Ebubekir Buber, Banu Diri and Özgür Koray Şahingöz introduces a system that includes natural language processing (NLP) techniques used to detect URL-based phishing attacks. In this study, after examining the salient features of phishing attacks, a Machine Learning-based system is proposed to detect such attacks. The system has been used to identify URLs used in phishing attacks by using Natural Language Processing techniques. Features were examined under two distinct groups: the first group, human-determined features to distinguish between malicious and legitimate URLs; the second group is the direct vectorization of the words in the URL. As a result of the tests applied, it was found that the Hybrid approach, which consists of a combination of word vectors and NLP-based features, was the most effective. Random Forest, Sequantial

Minimal Optimization and Naive Bayes algorithms were tested under various scenarios and it was observed that the hybrid model achieved a success rate of 89.9% with the Random Forest algorithm. The research aims to add information from different sources to the feature sets and make improvements that will further improve the current success rate [5].

In a study published by Indrakshi in 2021, HAYNES, Katherine, SHIRAZI, Hossein, and RAY highlight that today, cyber attackers are increasingly launching phishing attacks via SMS and social media, and games and dating apps offer a new attack vector. However, it states that current deep learning-based phishing detection applications are not suitable for mobile devices due to the computational overhead. The study proposes a lightweight URL-based phishing detection algorithm using natural language processing transducers for mobile devices. . This recommendation focuses on the applicability of existing deep learning-based phishing detection applications on mobile devices and emphasizes the use of language converters to distinguish fake websites from legitimate websites from URLs. First, artificial neural networks (ANNs) were applied to URL-based and HTML-based web features, resulting in 15 ANN models, with accuracy rates of over 96%, which are comparable to the best available approaches. However, deep ANNs with URL-only properties have poor performance, with the highest accuracy rate of 86.2%, indicating that URL-based features are insufficient to detect phishing websites even with deep ANNs. The study results demonstrate the potential of pre-trained converters such as BERT and ELECTRA to detect fraudulent websites on mobile devices more quickly and effectively than other traditional methods. It is emphasized that the highest performance model achieved through the use of feature-based features and deep ANN training is time-consuming and computationally costly in terms of implementation. However, the performance of an ANN with only URL-based properties is low, so it has been concluded that transformer-based models are needed for high-performance phishing detection using direct URLs [27].

This article focuses on the study presented by Ebubekir Buber, Banu Diri and Özgür Koray Şahingöz in 2018 under the title of "NLP based phishing attack detection from URLs". The article focuses on phishing attacks, which in recent years have become an increasing threat in cyberspace, especially with the increasing use of messaging and social networks. In traditional phishing attacks, users are encouraged to visit a fake website that has been carefully designed in an attempt to obtain their personal information (credit card

numbers, usernames, passwords, and even money). Many phishers carry out their attacks by attacking the target website by email. Inexperienced users (even experienced ones) may visit these fake websites and share their sensitive information. An analysis of phishing attacks in 45 countries conducted in the last quarter of 2016 found that China, Turkey, and Taiwan were the most affected by malware at a rate of 47.09%, 42.88%, and 38.98%, respectively. Detecting a phishing attack is a challenging problem because such attacks are considered semantically based attacks that target the weak points of the computer user. This article presents a phishing detection system that can detect such attacks using some machine learning algorithms and using some natural language processing techniques. Many tests have been carried out on the proposed system, and experimental results have shown that the Random Forest algorithm performs very well with a success rate of 97.2% [28].

3. Materials and Methods

In this study, the dataset named "The Ultimate Phishing URL Dataset for NLP Based Deep Learning" [29] developed by Dr. Ahmet Selman Bozkır provided by Hacettepe University was used. This dataset contains 800,000 tagged web URLs, half of which belong to phishing sites and half to legitimate sites. The dataset was collected between May 2019 and June 2021 and has a balanced distribution; It consists of 400 thousand phishing and 400 thousand legitimate URLs. The purpose of the dataset is to provide researchers with a URL dataset using a real-world sample for NLP-based deep learning studies. The average length of the URLs in the dataset is 86.21 characters for phishing sites and 46.43 characters for legitimate sites, respectively. The dataset is designed to be easy to use with various machine learning frameworks such as Sklearn, Pytorch, Tensorflow, and Keras In the dataset, "1" represents phishing and "2" represents legitimate URL.

Here are the example URLs:

- **2,"https://blog.sockpuppet.us/"**
- **2,"https://blog.apiki.com/seguranca/"**
- **1,"http://autoecole-lauriston.com/a/T0RVd056QXlNelF6T1RnPQ==/"**
- **1,"http://chinpay.site/index.html?hgcFSE@E$Z*DFcGVhBiNnikMOJIbhVgTFDRECTFgvBH"**
- **2,"http://www.firstfivenebraska.org/blog/article/covid-19-daily-digest#.XnvphYhKjIU"**

- **2,"http://streetoutlawsokc.com/"**
- **1,"https://maiiatt.weebly.com/"**

Convolutional Neural Networks (CNN) method was used to classify phishing and legitimate URLs. CNNs apply a process called convolution to distinguish various features in the data they receive as input. This process reduces the size of the data while preserving important information [30]. The developed CNN model is designed to perform in-depth analysis of URL character sequences and includes an architecture in which convolution layers work effectively on these sequences. This strategy allows for a detailed examination of the structural and semantic properties of URLs, thereby identifying patterns that are important for classification. Another reason why CNNs are preferred is the high accuracy rate and fast training process they provide. These features offer the capacity to work effectively on large data sets. The development and training process of the model was carried out in Jupyter Notebook, an interactive development environment that offers multi-language support and combines various components such as code, text, and visuals. Python is the core programming language used in this environment, and it has played a central role in building and testing the model. In addition, leading deep learning libraries such as TensorFlow and Keras were used in the model optimization and performance evaluation processes.

Within the scope of this study, in addition to the artificial intelligence model, an Android application was designed and developed using Android Studio. The main function of the application is to transmit the URLs that the user interacts with to a database server and to have this data retrieved and examined by our artificial intelligence model. The data obtained as a result of the analysis are sent back to the same database by the model. The mobile application is designed to query the results of these analyzes in the database and present them to the user in an understandable and accessible format. In this process, artificial intelligence operations are not performed within the application, but only undertake the tasks of directing the data flow and displaying the results. Android Studio serves as an official integrated development environment (IDE) offered by Google and designed for the Android operating system. This advanced environment contains all the tools and features required from the effective development of Android applications, to extensive testing processes, to their smooth operation, to their distribution to a wide range of users.

Google's Firebase platform was used to provide and manage the data flow between the Python-based artificial intelligence model and the mobile application developed on the

Android platform. Thanks to its real-time database features, Firebase instantly transmits the URL information clicked by the user in the mobile application to the artificial intelligence model and sends the response produced by the model back to the mobile application. This process was accomplished using Cloud Firestore's document-based, NoSQL database structure. Each URL click event is stored as a document in Filestore's collections, which is listened to by a function triggered by an artificial intelligence model running on the Python side. When the model completes its analysis on the processed data, it writes the result back to the Firebase database as a response in JSON format. The mobile app, on the other hand, provides a real-time response to the user by listening to this response through the data synchronization mechanisms provided by the Firebase SDK. This architecture is designed to meet the requirements of asynchronous data processing and bidirectional communication and contributes to the scalability of the system and the optimization of response time.

4. Propesed Approach

This section introduces an innovative and advanced approach designed for phishing URL detection. Based on the latest developments in the field of artificial intelligence, this approach aims to detect phishing attacks that direct internet users to deceptive and harmful websites, especially using Convolutional Neural Networks (CNN) technology. Developed to provide an effective solution against the ever-evolving threat of phishing in the world of cyber security, this model aims to set a new standard by surpassing existing approaches in this field.

This model performs in-depth characteristic analysis of URLs and classifies URLs as 'phishing' or 'normal' using these features. This model, which overcomes the limitations of traditional methods, aims to detect phishing URLs with a higher accuracy rate, while at the same time making this process faster and more efficient. The artificial neural networks that underpin the model consist of neuron-like units for data learning and prediction capabilities, while CNNs are designed to identify complex patterns in textual and visual data and play a critical role in detailed analysis of URL features in this model.

The most striking innovation of this model is its high accuracy rate, as well as its minimization of difficult processes such as data preprocessing and feature engineering. This feature not only speeds up the phishing URL detection process, but also makes it more efficient and user-friendly. In this section, the technical structure of the model, the data sources used, feature extraction, modeling, training and evaluation techniques will be discussed in detail.

Data Pre-Processing

Data preprocessing is a critical step in converting data into a format suitable for modeling. In this study, this process encompasses various stages such as loading and cleaning of data sets, URL tokenization, padding, label encoding, and one-hot encoding. Each step has been carefully designed to improve the overall performance and accuracy of the model.

Loading and cleaning of datasets

In the process of loading the datasets, pandas, a powerful and versatile library of the Python programming language, was effectively used. Pandas specializes in data analysis and processing in particular and uses table-like structures called DataFrames to process data. These structures organize data into columns and rows, allowing even complex data sets to be easily manipulated and analyzed. DataFrames make it easy to perform various operations on data; For example, operations such as data cleaning, sorting, grouping, and merging between data sets can be performed simply and effectively thanks to these structures. These capabilities of Pandas play a critical role in preparing datasets for the modeling phase, becoming an indispensable tool for data manipulation and preparation in data science projects.

In the data preprocessing process, as part of the hyperparameter optimization, a balanced sampling of 10% of the data set to be used in the training of the model was carried out. This approach involves sampling an equal proportion (10% each) of two classes of phishing and non-phishing URLs in the data set. This strategy aimed to enable the model to effectively classify both categories by allowing fair and equal representation of both classes in the educational process of the model. The balanced sampling method is designed to address the imbalance problems that may be encountered in the training process of the model. By representing both classes with equal weight, it is intended to increase the overall classification capability of the model and improve its ability to accurately recognize both phishing and non-phishing URLs. This method has contributed to the model performing effectively on larger and more diverse data sets, while also reducing common classification issues such as overfitting and underfitting.

URL tokenization

The tokenization process was carried out using the advanced Tokenizer class of the TensorFlow library. This process allows URLs to be analyzed at the character level and each character converted to a numeric value. The tokenizer class is specifically set up to tokenize at the character level, which allows each character to be recognized and processed by the model

as a separate element. This approach makes it possible to model the structural features of URLs and the character sequences they contain in more detail. The fit_on_texts method of the tokenizer class runs on the URLs used as training data, identifying each unique character in the URLs and assigning a unique numeric value to each one. This creates a comprehensive dictionary that converts characters to numeric values. Then, the texts_to_sequences method converts the URLs used as both training and test data into arrays of numeric values. This method matches the characters in each URL to their counterparts in the previously created dictionary and translates these characters into arrays that represent them with numeric values.

Padding

Padding has been implemented to effectively handle URLs in datasets. First, the length of the longest URL was determined by examining the lengths of all URLs in the data sets; This length was used as the reference maximum length value for the padding operation. Then, URLs that didn't reach this maximum length were brought to a standard length by filling in the missing parts with zero values. This standardization was accomplished through TensorFlow's pad_sequences function. pad_sequences function converts all URLs to arrays of the same length by adding the required number of zeros to the beginning or end of the URLs. This approach enables neural networks to process all URLs as the same size, increasing data consistency in the model's training process. Padding, especially in processing variable-length data, is critical and plays an important role in improving the performance and accuracy of the model.

Label encoding ve one-hot encoding

For the training of the model, the labels need to be converted to digital format. For this purpose, the LabelEncoder class of the scikit-learn library is used. LabelEncoder is a process that converts tags to numeric values, and in this study, phishing tags are coded as 0 and normal tags as 1. This classification allows the model to distinguish and correctly handle labels. The fit_transform method of the LabelEncoder class was used to convert the labels in the training dataset to numeric values. Likewise, the transform method was used to convert the labels in the test data set to numerical values.

In addition, the to_categorical function of the TensorFlow library was preferred for one-hot encoding. One-hot encoding is a method that converts tags into binary vector form, in the process, for example, the phishing tag is encoded as [1, 0], while the normal tag is encoded as [0, 1]. This binary coding allows the model to handle labels in more detail and perform better,

especially in multiclass classification problems. The to_categorical function has been used to convert labels from both training and test datasets to this binary vector format. These two methods play an important role in improving the overall performance and accuracy of the model by ensuring that the model correctly recognizes and processes labels.

Feature Extraction

Feature extraction is a critical process for extracting meaningful and actionable information from data. In this study, convolutional neural networks (CNNs) were used for detailed analysis of the data. CNNs are specifically designed to detect the complex, local characteristics of data. In this process, convolutional layers create activation maps using kernel matrices as they navigate over the data. These maps capture the character-level attributes of URLs, enabling the model to deeply analyze the structural and contextual properties of URLs. This analysis allows the model to distinguish phishing URLs from regular URLs more precisely and accurately. This is a significant contribution to improving overall model performance and accuracy. This CNN-based method of attribute extraction offers a more comprehensive and effective solution than traditional techniques, as it enables a more sophisticated classification by revealing the finer details and hidden properties of URLs.

Hyperparameter Optimization

Hyperparameter optimization aims to determine the optimal parameter values to maximize the performance of the model. In this study, a detailed optimization process was performed on various hyperparameters to maximize the performance of the CNN-based model. Different numbers of convolutional layers, filter counts, and kernel sizes have been tested for CNNs. These hyperparameters are factors that directly affect how the model processes the data and the accuracy of the resulting model. In this process, the grid search method was used. Grid search aims to determine the set of parameters that maximize model performance by trying all possible combinations of the specified parameters.

The hyperparameter optimization process includes the following steps:

- **Determination of Parameter Sets:** Various hyperparameters and their appropriate ranges of values have been carefully determined to maximize the performance of the model. These hyperparameters include the optimization algorithm, the number of convolutional layers, the number of filters in each layer, kernel sizes, batch size, and the number of training cycles (epochs). Each hyperparameter is an important factor

that directly affects the model's data processing capacity and learning dynamics. Determining these parameters is critical for optimizing the model's efficiency, speed, and overall accuracy. This process aims to maximize the performance of the model on the data set and make more accurate predictions.

- **Implementation of Grid Search:** Within the specified parameter ranges, all possible parameter combinations have been thoroughly tested using the grid search method. This process involves ranges of values set for each of the various hyperparameters, such as the number of convolutional layers, the number of filters, and the size of the cores. By systematically experimenting with each combination of these parameters, the grid search aims to determine the optimal set of parameters that can maximize the performance of the model. This method is an effective tool for objectively evaluating the different configurations of the model and determining the hyperparameter combination that gives the most effective results.
- **Creation of parameter combinations:** In the hyperparameter optimization process of the model, the product function of Python's itertools library was used to create all possible combinations in the specified parameter set. This function systematically generates all possible combinations for each value in the given parameter ranges. This method makes it possible to comprehensively study the interactions of different values of hyperparameters with each other and determine the optimal set of parameters. This process is an effective tool in detecting combinations of hyperparameters that will maximize the performance of the model.
- **Application of the Early Stop Function:** In the hyperparameter optimization process of the model, the early stop recall function was used effectively in order to prevent the problem of overfitting. Early stopping is designed to prevent the model from overfitting the training data set, thereby reducing its ability to generalize on the test data set. This function is defined by the 'EarlyStopping' module of the TensorFlow library. The early stop function automatically stops training when there is no improvement in the model's performance over a certain number of epochs. This not only optimizes training time, but also prevents the model from overadapting, resulting in more robust and reliable results. This approach plays an important role in improving the efficiency and overall performance of the model.

- **Creating, Training, and Testing the Model for Each Parameter Combination:** During the grid search process, model creation, training, and testing were performed separately for each determined parameter combination. Each model has a structure consisting of an input layer, multiple convolutional layers, a global maximum pooling layer, and an output layer. These layers and architecture of the model are customized according to the selected parameter combination. In the process of creating the model, the 'Sequential' class of the TensorFlow library was used. This class arranges the layers in a sequential manner, allowing the model to be easily built. For each combination of parameters, the model is structured according to parameters such as the number of convolutional layers, number of filters, core size, etc., and then trained on the training dataset. After the completion of the training process, the performance of the model was evaluated on the test data set. This comprehensive process aims to objectively assess the impact of each combination of parameters on the overall performance of the model and determine the optimal model configuration.
- **Determination of the Best Parameters and Accuracy:** In the hyperparameter optimization process of the model, the performance of the model was meticulously evaluated for each parameter combination. This evaluation was made using a variety of performance metrics, such as the model's accuracy, sensitivity, and recall. The accuracy values obtained for each parameter combination were compared with the best accuracy value ever recorded. In this process, if the accuracy value of a parameter combination is found to be higher than the best available accuracy value, this is updated as the best accuracy value and the corresponding parameter combination is recorded as the best parameters. This method provides a systematic and objective approach to determining the most effective configuration of the model. As a result, the optimal set of hyperparameters is determined to maximize the overall performance of the model. This process aims to maximize the prediction accuracy of the model on the data set and obtain more reliable results.

In this study, the set of parameters for grid search is as follows:

Parameter Category	Test Values
Optimization Algorithm	**RMSprop, Adam**
Number of Convolutional Layers	**1, 2, 3**
Number of Filters	**32, 64, 128, 256**
Kernel Size	**3, 6**
Batch Size	**32**
Epoch	**20**

TABLE 1 Parameter set for grid search

This set of parameters contains 24 different combinations. For each combination, the model is trained and tested. The performance of the model was measured as an accuracy rate. Accuracy is a metric that shows how well the model's predictions align with the actual labels. The accuracy rate takes a value between 0 and 1. 1 indicates that all of the model's predictions are correct, while 0 indicates that all of the model's predictions are wrong. In this study, the combination of parameters that gives the highest accuracy rate was selected.

In the table below, hyperparameter optimization is performed with 10 percent of the data of the data set. The title of the table summarizes the contents of the table. The column headers of the table show the hyperparameters that have been optimized. The row headings of the table show the parameter combinations. The data of the table shows the accuracy values obtained for each combination of parameters. A note is attached at the bottom of the table, in which the best parameters and accuracy are indicated.

Parameter Combination	**Number of Convolutional Layers**	**Number of Filters**	**Kernel Size**	**Batch Size**	**Epoch**	**Optimization Algorithm**	
						‘adam’	**‘rmsprop’**
						Accuracy	**Accuracy**
1	1	32	3	32	20	0.9231	0.9222
2	1	32	6	32	20	0.9252	0.9269
3	1	64	3	32	20	0.9418	0.9311
4	1	64	6	32	20	0.9297	0.9292
5	1	128	3	32	20	0.9396	0.9347
6	1	128	6	32	20	0.9361	0.9373
7	1	256	3	32	20	0.9329	0.9325
8	1	256	6	32	20	0.9409	0.9485

9	2	32	3	32	20	0.9447	0.9504
10	2	32	6	32	20	0.9383	0.9397
11	2	64	3	32	20	0.9482	0.9451
12	2	64	6	32	20	0.9476	0.9404
13	2	128	3	32	20	0.9384	0.9381
14	2	128	6	32	20	0.9503	0.9513
15	2	256	3	32	20	0.9514	0.9428
16	2	256	6	32	20	0.9443	0.9434
17	3	32	3	32	20	0.9490	0.9479
18	3	32	6	32	20	0.9506	0.9481
19	3	64	3	32	20	0.9452	0.9358
20	3	64	6	32	20	0.9483	0.9568
21	3	128	3	32	20	0.9541	0.9494
22	3	128	6	32	20	0.9449	0.9477
23	3	256	3	32	20	0.9504	0.9529
24	3	256	6	32	20	0.9548	0.9428

TABLE 2 Hyperparameter Optimization Table

In this study, it is aimed to determine the most appropriate hyperparameter combinations to maximize model performance through extensive experimental analysis on convolutional neural networks (CNNs). The analyses evaluated the effects of different optimization algorithms and hyperparameter settings on model performance.

Although the model ranked 20th in Table 2 and using the RMSprop optimization algorithm achieved the highest accuracy rate, model number 14 was found to be more advantageous when considering the balance of cost and performance. This model uses the Adam optimization algorithm and has a more stable performance. In terms of cost/performance relationship, the Adam algorithm offers a more balanced option than RMSprop. This is especially important when considering large data sets and lengthy training processes, as the Adam algorithm typically requires faster convergence and fewer computational resources.

For this reason, the model using the Adam algorithm number 14 was preferred for the optimization of hyperparameters such as batch size, number of filters, filter size, kernel size, number of layers. This choice is intended to optimize computational costs while maintaining the high accuracy of the model, thereby improving overall performance. This approach aims to manage resource usage in a balanced way while increasing the efficiency and applicability of the model.

Create a model

This section describes the architecture and parameters of the recommended model for phishing URL detection. The model is built in the Python environment using the Keras library. The model is a model created by combining artificial neural networks and CNN. A model is a model that extracts character-level attributes from URLs and uses those attributes to classify URLs as phising or normal.

The model consists of six layers. These layers can be listed as follows:

- **Embedding Layer:** One of the key components of the model, the Embedding layer is designed to extract character-level attributes of URLs. This layer transforms each character into a 50-dimensional vector, providing a rich and meaningful representation of the characters. The embedding layer takes the maximum number of characters in the dataset as input, which allows the model to handle URLs of different lengths. As an output, this layer produces detailed attribute vectors of URLs, which are further processed in later layers of the model. This configuration of the embedding layer allows the model to effectively process character-level data and learn more complex features, thus improving the overall performance and accuracy of the model.
- **Conv1D Layers:** Conv1D convolutional layers, which play an important role in the architecture of the model, create activation maps by performing convolution operations on feature vectors. These layers enable the model to effectively handle the character-level attributes of URLs. Each Conv1D layer is configured with a certain number of filters and core sizes. The model has two Conv1D layers, each using 128 filters and 6 size cores. In these layers, the ReLU activation function is preferred. These two Conv1D layers increase the capacity of the model to extract attributes from URLs and help it achieve more accurate classification results.
- **Use of ReLU Activation Function:** ReLU (Rectified Linear Unit) activation function was used in the Conv1D layers of the model. ReLU is a nonlinear activation function and is often preferred in deep learning models. This function equates the negative inputs to zero, while leaving the positive inputs intact. Thanks to this feature, ReLU makes it easier for the model to learn complex patterns and features, reducing the problem of gradient loss, allowing for faster training processes. Using ReLU on these layers of the model enables the efficient creation of activation maps and improves overall model performance.

- **BatchNormalization Layer:** The BatchNormalization layer, which is included in the structure of the model, is an important normalization layer used after the convolutional layers. This layer normalizes the values of the activation maps produced by the convolutional layers. Specifically, by setting the mean of the activation maps to zero and their standard deviation to one, it brings the inputs of each layer into a standard distribution. This normalization process accelerates the training process of the model and contributes to a more effective learning process. In addition, the BatchNormalization layer reduces the problem of internal covariate shift that the model may encounter during training, thus allowing the model to be trained more stably and quickly. The integration of this layer into the model architecture contributes significantly to the improvement of overall model performance and learning efficiency.
- **Dropout Layer:** In the model, convolutional layers are followed by the Dropout layer as an editor. This layer temporarily disables neurons at a randomly determined percentage in each training cycle (epoch), reducing the model's tendency to overfitting. Overfitting is when the model is overly attuned to the training data set but fails to perform as expected on the test data set. In this study, a dropout value of 25% was selected for the Dropout layer. This ratio provides a balanced approach to maintain the model's adequate learning capacity while also avoiding overfitting. This strategic use of the dropout layer increases the model's ability to generalize and helps it perform more reliably on the test data set.
- **GlobalMaxPooling1D Layer:** The pooling layer selects the largest values from the activation maps, reducing the size of these maps. This reduces the complexity of the model and the number of parameters, while preserving important features. The use of the GlobalMaxPooling1D layer allows the model to work more efficiently and also helps it achieve high performance with fewer parameters. The integration of this layer into the model architecture increases the overall efficiency and processing speed of the model, while also optimizing memory usage and strengthening the model's ability to generalize.
- **Dense Layer:** The Dense layer, which is in the last stage of the model, is fully connected and functions as the output layer of the model. This layer contains only a single neuron, which calculates the probability that URLs are phishing as a value between 0 and 1. The activation function used for this calculation is sigmoid. The

sigmoid function is especially ideal for binary classification tasks because it compresses the output to a probability value between 0 and 1. This allows the model to produce a clear probability score of whether or not phishing is present for each URL. Structuring the Dense layer in this way allows the model to present the results in an understandable and actionable format, so that the model's predictions can be easily interpreted and evaluated.

Improving the Model Compilation Process: In order for the model to be trained effectively, the model's optimization algorithm, loss function, and performance metrics should be carefully determined during the compilation phase. The optimization algorithm determines how to update the weights and parameters of the model, and this directly affects the efficiency of the model's learning process. The loss function measures how much the model's predictions deviate from the actual labels, which plays a critical role in understanding how well the model is performing. Performance metrics, on the other hand, are used to evaluate how successful the model is in the training and testing phases. These metrics typically include metrics such as accuracy, precision, and recall. Accurately identifying these components of the model is vital in optimizing the overall performance and prediction accuracy of the model. This process aims to maximize the effectiveness of the model on the data set and obtain more reliable results.

Optimization, Loss Function and Performance Metric Selection in the Model Compilation Process: In this study, the basic components carefully selected during the compilation phase of the model are the Adam optimization algorithm, the binary_crossentropy loss function and the accuracy performance metric. Adam is an advanced optimization algorithm with adaptive learning rates, which allows the model to effectively adjust the learning process for different parameters, thus allowing for a faster and more efficient training process. Binary_crossentropy is a loss function that precisely measures how much the model's predictions deviate from the actual labels in binary classification tasks, and this plays a critical role in understanding how well the model is performing. The accuracy metric, on the other hand, reflects the model's capacity to make accurate predictions, allowing it to easily evaluate its overall performance. The combination of these three components enables the model to perform the phishing URL detection task with high accuracy and reliability, thus maximizing the overall effectiveness and accuracy of the model. These choices contribute to the model's optimal results during the training and testing phases, improving its overall performance.

Model's Testing Process and Performance Analysis:

This section covers how the performance of the model on the test dataset is evaluated and analyzed. The testing process of the model is vital to measure its effectiveness on new data that it did not encounter during the training phase and to evaluate its ability to generalize. To evaluate the predictive capabilities of the model, the probability of phishing was calculated for each URL in the test data set using the model.predict function. These probability values were determined on a scale between 0 and 1 and were converted into a binary classification using a threshold value of 0.5. According to this threshold value, values higher than 0.5 are classified as 'phishing' (1), and values lower than 0.5 are classified as 'normal' (0). Metrics used to evaluate the overall performance of the model include accuracy, precision, recall, and F1 score. These metrics were used to determine the alignment of the model's predictions with the actual labels and how effectively it performed in various situations. These evaluations were carried out through the relevant functions of the sklearn.metrics library. These functions take the model's predictions and actual test labels, calculating various metric values that measure the model's performance. The metric values obtained by the model on the test data set are critical indicators of how reliable and effective the model is in real-world conditions. This analysis is used to understand how the model performs in different scenarios and to identify the model's strengths and areas for improvement. These considerations play a key role in assessing the effectiveness and reliability of the model on real-world data.

Visualization of Model Performance: A confusion matrix was used to present the performance of the model in a more understandable and visual way. A confusion matrix is a tool that shows in detail how the model's predictions fit in with the actual labels. This matrix is computed with the confusion_matrix function of the sklearn.metrics library. The function takes the model's predictions and the actual test labels and creates the confusion matrix.

The confusion matrix is visualized as a colored heatmap using the seaborn library's heatmap function. This visualization presents the model's performance in a more intuitive and interactive format. The confusion matrix shows the distribution of the model's predictions by actual labels over four basic categories: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). These categories are defined as:

- True Positive (TP): URLs that the model correctly guesses as phishing.
- False Positive (FP): URLs that the model incorrectly guesses to be phishing, but are actually normal.

- True Negative (TN): URLs that the model normally correctly predicts.
- False Negative (FN): URLs that the model normally incorrectly predicts, but are These cells indicate the accuracy and inaccuracy of the model's predictions, as well as how effectively it can distinguish between phishing and legitimate URLs. Visualization of the confusion matrix is an important tool in identifying the strengths of the model and areas that need improvement. This analysis plays a critical role in evaluating the model's performance on real-world data.

Model Integration with Mobile Application

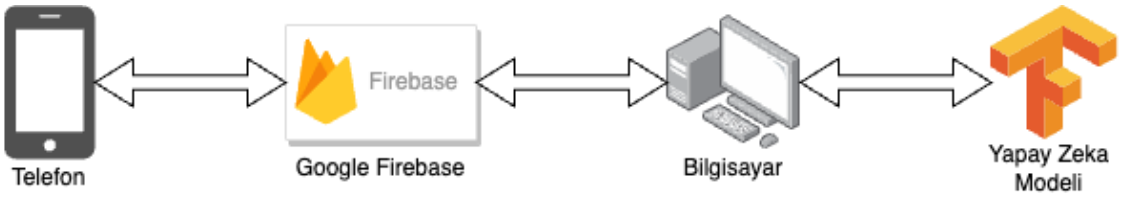

FIGURE 1

In this study, URL data provided by users will be collected through an Android mobile app and sent to Firebase Cloud. Firebase Cloud is a Cloud Services platform offered by Google and is often used as a cloud platform that offers app hosting, database management, authentication, storage, analytics, and a range of other services for app developers and businesses. Firebase Cloud provides app developers with a set of tools and services to develop, deploy, and manage their apps quickly and reliably. This process is designed to maximize the user experience and perform the data collection process seamlessly. URL data uploaded to Firebase Cloud will be continuously monitored by a Python script. This script will detect and retrieve this data for processing when new data is added to Firebase. The retrieved URL data will be converted by the Python script into a format that the model can process, so that the data can be effectively analyzed by the model. The transformed data will then be interpreted into a pre-trained machine learning model, which will make a prediction as to whether the URL is phishing or not. This prediction of the model will be stored by restoring it to Firebase Cloud. This process allows the model's predictions to be obtained quickly and efficiently. The mobile app will continuously monitor these updates in Firebase Cloud and instantly notify the user of the model's analysis result. This integration provides real-time and accurate information to the user, so users can quickly assess the security of the URL. This transmission process is visually explained by the diagram in figure 1 shown below.

5. Codes:

```
# Import required libraries

# Used for numerical operations like array manipulation

import numpy as np

# Pandas library for data manipulation and analysis

import pandas as pd

# Normalizes the inputs to a layer

from keras.layers import BatchNormalization

# Tokenizes texts into tokens

from tensorflow.keras.preprocessing.text import Tokenizer

# Pads sequences to the same length

from tensorflow.keras.preprocessing.sequence import pad_sequences

# Sequential model for linear stack of layers

from tensorflow.keras.models import Sequential

# Various layers for model construction

from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense

# Dropout layer to reduce overfitting

from keras.layers import Dropout

# Encodes labels to normalized format

from sklearn.preprocessing import LabelEncoder

# Callbacks for controlling the training process

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Visualization library based on matplotlib

import seaborn as sns
```

```
# Basic library for plotting graphs
import matplotlib.pyplot as plt
# Metrics for model evaluation
from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix
# Function to load dataset
def load_dataset(file_path):
    # Load a CSV file into a DataFrame without a header and with quotes around strings
    df = pd.read_csv(file_path, header=None, quotechar='"')
    return df

# Load training and testing data
# Load the training data
train_df = load_dataset('updated_train.csv')
# Load the testing data
test_df = load_dataset('updated_test.csv')
# Prepare URL and label data for training and testing
# Extract URLs from training data
train_urls = train_df[1].tolist()
# Extract labels from training data
train_labels = train_df[0].tolist()
# Extract URLs from testing data
test_urls = test_df[1].tolist()
# Extract labels from testing data
test_labels = test_df[0].tolist()
```

```
# Tokenize the URL data

# Initialize a tokenizer that works at the character level

tokenizer = Tokenizer(char_level=True)

# Fit the tokenizer on the training URLs

tokenizer.fit_on_texts(train_urls)


# Convert URLs to sequences of integers

# Convert training URLs to sequences

train_sequences = tokenizer.texts_to_sequences(train_urls)

# Convert testing URLs to sequences

test_sequences = tokenizer.texts_to_sequences(test_urls)


# Find the maximum sequence length for padding

# Find max length

max_length = max(max(len(s) for s in train_sequences), max(len(s) for s in test_sequences))


# Pad the sequences to ensure uniform length

# Pad training sequences

train_data = pad_sequences(train_sequences, maxlen=max_length)

# Pad testing sequences

test_data = pad_sequences(test_sequences, maxlen=max_length)


# Encode the labels
```

```
# Initialize the label encoder

label_encoder = LabelEncoder()

# Fit and transform training labels

train_labels = label_encoder.fit_transform(train_labels)

# Transform testing labels

test_labels = label_encoder.transform(test_labels)

# Model building

# Import the Adam optimizer

from keras.optimizers import Adam

# Create a sequential model

model = Sequential()

# Add embedding layer

model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=50, input_length=max_length))

# Add convolution layers

model.add(Conv1D(filters=128, kernel_size=6, activation='relu'))

model.add(Conv1D(filters=128, kernel_size=6, activation='relu'))

# Add global max pooling layer

model.add(GlobalMaxPooling1D())

# Add dense layer for output

model.add(Dense(1, activation='sigmoid'))


# Compile the model

# Initialize Adam optimizer with a learning rate

optimizer = Adam(lr=0.0001)
```

```
# Compile the model
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
# Print the model summary
model.summary()
# Callbacks for training
early_stopping = EarlyStopping(
    # Monitor validation accuracy
    monitor='val_accuracy',
    # Mode is max because we want to maximize accuracy
    mode='max',
     # Number of epochs with no improvement after which training will be stopped
    patience=8,
    # Minimum change to qualify as an improvement
    min_delta=0.00009,
    # Restore model weights from the epoch with the best value of the monitored quantity
    restore_best_weights=True,
    # Verbosity mode
    verbose=1
)
model_checkpoint = ModelCheckpoint(
    # File path to save the model
    'best_model.keras',
    # Monitor validation accuracy
    monitor='val_accuracy',
```

```
    # Mode is max because we want to maximize accuracy
    mode='max',
    # Save only when the monitored quantity has improved
    save_best_only=True,
    # Verbosity mode
    verbose=1
)
# Train the model
# Fit the model
history = model.fit(train_data, train_labels, epochs=100, validation_data=(test_data, test_labels), batch_size=32,
          callbacks=[early_stopping, model_checkpoint], verbose=1)
# Save the training history
import json
# Get history data from the model
history_dict = history.history
# Save history data to a JSON file
json.dump(history_dict, open("history.json", 'w'))
# Plot training and validation loss
# Plot training loss
plt.plot(history.history['loss'], label='Training Loss')
# Plot validation loss
plt.plot(history.history['val_loss'], label='Validation Loss')
# Title of the plot
plt.title('Training and Validation Loss')
```

```
# Y-axis label
plt.ylabel('Loss')
# X-axis label
plt.xlabel('Epoch')
# Add legend
plt.legend()
# Display the plot
plt.show()


# Plot training and validation accuracy
# Plot training accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
# Plot validation accuracy
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
# Title of the plot
plt.title('Training and Validation Accuracy')
# Y-axis label
plt.ylabel('Accuracy')
# X-axis label
plt.xlabel('Epoch')
# Add legend
plt.legend()
# Display the plot
plt.show()
```

```
# Evaluate model performance

# Predict on test data

predictions = model.predict(test_data)

# Convert predictions to binary

predictions = (predictions > 0.5).astype("int32")


# Calculate precision, recall, and F1-score

# Calculate precision

precision = precision_score(test_labels, predictions)

# Calculate recall

recall = recall_score(test_labels, predictions)

# Calculate F1 score

f1 = f1_score(test_labels, predictions)


# Print evaluation metrics

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1 Score: {f1}")


# Plot the confusion matrix

# Calculate confusion matrix

cm = confusion_matrix(test_labels, predictions)

# Plot confusion matrix using seaborn
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

# X-axis label

plt.xlabel('Predicted')

# Y-axis label

plt.ylabel('True')

# Title of the plot

plt.title('Confusion Matrix')

# Display the plot

plt.show()
```

6. Results

In this project, it is aimed to effectively detect phishing URLs by developing an artificial intelligence model based on convolutional neural networks (CNN). For the training of the model, a balanced dataset containing a total of 800 thousand URLs was used; this dataset consists of 400,000 phishing URLs and 400,000 legitimate URLs. This balanced approach is critical for the model to be able to recognize both types of URLs effectively.

The training of the model was carried out on a powerful infrastructure provided by Microsoft Azure. This infrastructure includes a computer environment with a Linux operating system, 16-core CPU, 128 GB RAM and 256 GB SSD, allowing model development and testing to be carried out efficiently and effectively using the Python programming language and Jupyter Notebook. The training process of the model took about 13 hours and was trained for 21 epochs. In this process, the combination of hardware and cloud services contributed to the rapid and effective completion of data processing and learning processes.

The results of this project include significant achievements both quantitatively and qualitatively. The model's performance has been evaluated using various metrics such as accuracy, sensitivity, recall, and F1 score, and the model has shown quantitative achievements such as high accuracy rates. These results demonstrate the model's ability to effectively detect

phishing URLs and make a significant contribution to the field of cybersecurity in general. The success of the model reflects the quality of the developed artificial intelligence algorithms and the data set used, as well as the effectiveness of the infrastructure provided in the model training process.

Quantitative Results

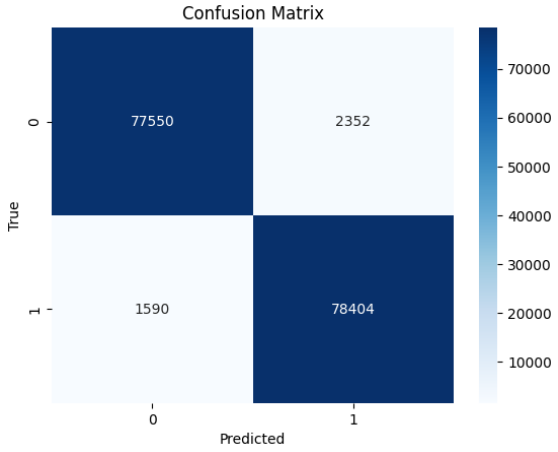

FIGURE 2

The performance of the model was analyzed in detail through the confusion matrix. The confusion matrix is an important tool that visually represents the predictive ability and accuracy of the model. In this matrix, it can be seen that the model is capable of predicting 77,550 true negatives (TN) and 78,404 true positives (TPs). The number of false negative (FN) predictions was 1,590, while the number of false positive (FP) predictions was determined as 2,352. These results show that the model has a high accuracy rate when it comes to detecting phishing URLs. A high number of true negative and true positive predictions indicates that the model is able to effectively distinguish between both phishing and legitimate URLs. On the other hand, a lower number of false negatives and false positives indicates that the model has a false alarm rate and is less likely to miss phishing URLs. This confusion matrix plays a critical role in assessing the overall performance and reliability of the model. The high accuracy rate of the model shows that it is an effective tool for protecting users in the field of cybersecurity. This analysis is an important reference point in assessing the effectiveness and reliability of the model on real-world data.

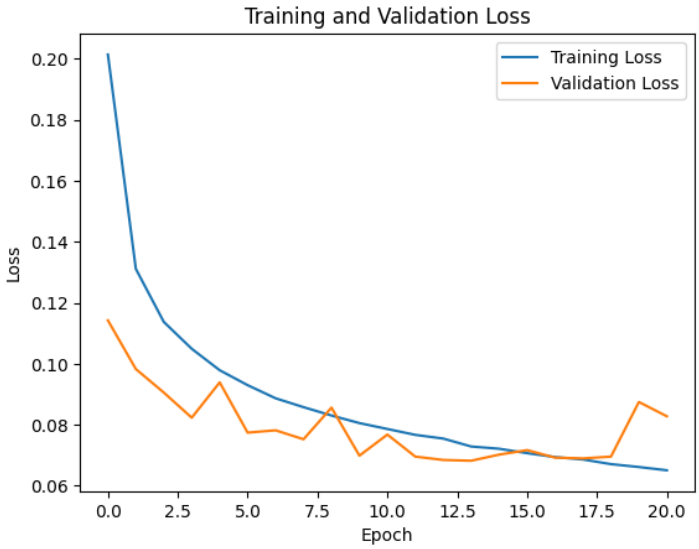

FIGURE 3

The analysis of the losses of the model in the training and validation processes was visually examined through the training and validation loss graphs. A loss is a metric that shows how far the model's predictions are from the actual values; Lower loss values indicate that the model is learning patterns in the data set more accurately. These graphs show how the model improved over the course of training and how its losses decreased. Training loss reflects the model's performance on the training dataset, while validation loss indicates the model's performance on the validation dataset. Over the course of the training process, training loss dropped rapidly, indicating that the model was effectively learning the patterns in the training dataset. On the other hand, the loss of validation decreased more slowly and eventually settled at a value of around 0.06. The training of the model was terminated in the 14th epoch due to the early stop mechanism. This indicates that the pattern does not show any further improvement after a certain point and meets the criteria for an early stop. Early stopping was used to prevent the model's tendency to overfit and to increase the efficiency of the training process. These churn graphs are an important tool in visually assessing the model's progress and performance in the training and validation processes. The model's low training and validation losses indicate the potential for high performance and reliability in cybersecurity applications. This analysis plays a critical role in assessing the effectiveness and reliability of the model on real-world data.

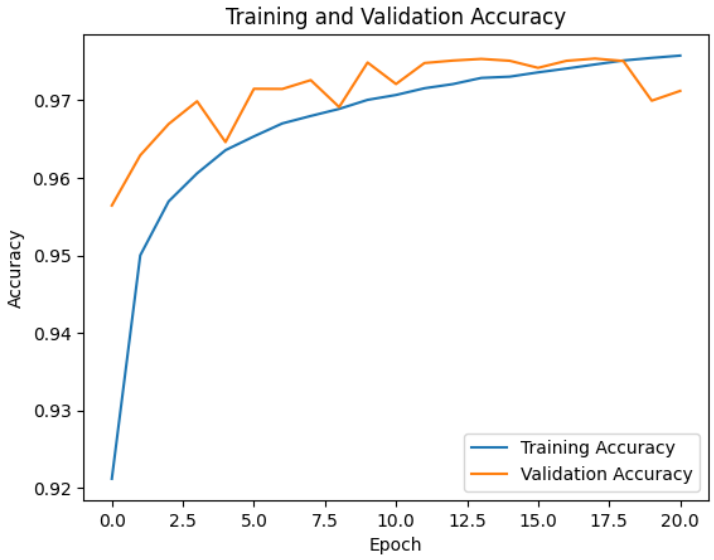

FIGURE 4

The performance of the model in the training and testing processes was visually examined through training and test accuracy graphs. These graphs reflect the development and stability of the model over the course of 21 epochs. Training accuracy is represented by a blue line, and validation accuracy is represented by an orange line. Both accuracy rates increased over the course of 21 epochs, reaching a value of approximately 0.97. Importantly, the model's training was terminated in the 21st epoch due to the mechanism of stopping early. This indicates that the model did not show any further improvement after the highest accuracy value it achieved in the 14th epoch and met the criteria for early stopping. Early stopping was used to prevent the model's tendency to overfit and to increase the efficiency of the training process. These high accuracy rates and early stop implementation show that the model learns effectively and is able to generalize in both the training dataset and the validation dataset. Training and validation accuracy graphs are an important tool in visually assessing the success of the model's training process and its performance against the validation dataset, demonstrating that the model can be used effectively in cybersecurity applications.

Precision	%97,09
Recall	%98,01
F1 Score	%97,55
Accuracy	%97,54

TABLE 3

The performance of the model was evaluated using important metrics such as precision, recall, and F1 score. These metrics reflect the reliability and accuracy of the model's predictions.

Precision: The model's 97.09% accuracy indicates how many of the URLs that the model flags as phishing are actually phishing. This high rate indicates that the model has a low rate of false alarms and makes reliable predictions.

Recall: The model's 98.01% sensitivity rate indicates how many of the URLs that are actually phishing are correctly detected by the model. This rate shows that the model is very unlikely to miss phishing URLs.

F1 Score: The model's 97.55% F1 score is the harmonic mean of the precision and precision metrics and indicates the overall balanced performance of the model. This high F1 score indicates that the model performs well in terms of both precision and responsiveness.

Overall Accuracy: The model's overall accuracy rate of 97.539% indicates how much of all the model's predictions are correct. This high ratio indicates that the model has an overall high level of reliability and performance.

Each of these metrics addresses different aspects of the model's predictive capability and provides a comprehensive assessment of the model's overall performance. High precision, sensitivity, and F1 score values indicate that the model is effective and reliable in detecting phishing URLs, making it an ideal candidate for use in cybersecurity applications. These assessments signal the effectiveness and reliability of the model on real-world data, thus demonstrating the model's potential to be used as an effective tool in the field of cybersecurity.

Qualitative Analysis

In terms of user experience, the model's ability to make fast and accurate predictions allowed users to effectively detect phishing threats. In addition to the artificial intelligence model developed as a result of this possibility, a mobile application was designed and integration with the Firebase platform was provided. In the field of cybersecurity, this project's integration of mobile app and AI model significantly enhances users' security and protection against phishing threats. This integration significantly improves the user experience and increases the security of users in their daily digital interactions. Users can easily transmit suspicious URLs through the mobile app, and this information is quickly transferred to the

server through Firebase. The AI model on the server quickly and accurately analyzes the trustworthiness of these URLs. The results of the analysis are reported back to users through the mobile app, so users can immediately evaluate suspicious URLs. This process allows users to detect phishing threats in real-time and act accordingly. With the fast and accurate feedback provided by the mobile app, users can be more aware and prepared for potential cyber threats. While this integration increases users' awareness of cyber security, it provides more effective protection against the risks they may encounter in their daily internet use.

In short, this project provides users with a safer digital experience while strengthening their cyber security. It makes a significant contribution to ensuring the security of users better and becoming more susceptible to cyber threats. This integration can be considered as an innovation that significantly improves the user experience in the field of cyber security.

Strengths

High Accuracy and Reliability: The model achieved very high results in key performance metrics such as precision, sensitivity, and F1 score. These high values indicate that the model's predictions are both highly accurate and reliable. In particular, the model's high precision rate indicates that the vast majority of URLs it flags as phishing are indeed malicious, while its high sensitivity rate indicates that it can successfully detect a large proportion of URLs that are actually phishing. In addition, a high F1 score indicates that it strikes a perfect balance between precision and precision, and that the model performs well. These superior performance metrics highlight the model's ability to effectively detect phishing URLs and indicate that the model is a strong candidate for use in cybersecurity. This high performance of the model can make significant contributions to the protection of users in cyber security applications and provide a secure internet experience to a wide range of users. These results are important indicators that show the success and effectiveness of the model in the development and implementation process.

Effective Generalization Capability: The model was trained using a balanced and large data set and exhibited a high overall accuracy rate. This shows that the model can effectively adapt to different types of data sets and various scenarios. This generalization capability of the model indicates that it has a wide range of applications and increases its usability in various cybersecurity scenarios. The use of a large and balanced dataset ensures that the model can cover a wide range of threats and recognize different phishing tactics. This contributes to the model's ability to detect not only existing threats, but also new and advanced threats that may

arise in the future. The model's high overall accuracy indicates that a good balance is maintained between different data types and scenarios, and that the model can deliver reliable results in real-world applications. This ability to generalize allows the model to be used effectively in a variety of applications in the field of cybersecurity. For example, data from different geographies may be compatible with various internet usage habits and changing technological environments. This wide range of applications and generalization capability of the model provides a significant advantage in the development and implementation of cyber security solutions.

Fast and Efficient Processing Capacity: The powerful and advanced infrastructure provided by Microsoft Azure has played a critical role in the model's data processing and learning processes. This infrastructure has allowed the model to process large data sets quickly and efficiently, as well as develop and train complex model structures. The high processing power and large storage capacity provided by Azure have contributed to accelerating the training process of the model and optimizing the data processing processes. . This infrastructure is especially vital for the processing of large data sets and the development of complex model structures. Large data sets often require high processing power and storage, and the infrastructure provided by Azure has met these requirements, making it possible to train and test the model on larger and more diverse data sets. In addition, in the process of developing and training complex model structures, the infrastructure provided by Azure has enabled the model to learn and evolve more quickly and effectively. This infrastructure provided by Microsoft Azure has improved the overall performance and effectiveness of the model, allowing it to have a broader impact on cybersecurity. The technological advantages provided by this infrastructure have significantly improved the training process and overall applicability of the model. This is an example that emphasizes the importance of technological infrastructure in the process of developing and implementing the model.

Mobile App Integration: The integration of the model with a mobile app has significantly increased users' ability to detect phishing threats in real-time and react quickly. Through this integration, users can easily analyze suspicious URLs and receive instant feedback, giving them the ability to make quick and informed decisions against potential threats. Mobile app integration stands out as a factor that significantly improves the user experience. It improves the accessibility and ease of use of cybersecurity applications for users, enabling a wide range of users to be more effectively protected against cyber threats. This integration helps users take a more proactive and informed approach to the risks they may face in their day-to-day

digital interactions. Furthermore, mobile app integration has the potential to increase cybersecurity awareness and contribute to making users more susceptible to cyber threats. This tool in the hands of users provides them with a safer internet experience and encourages cybersecurity practices to become a part of daily life. This integration emphasizes the importance of a user-centered approach to cybersecurity and provides more effective protection against the cyber threats that users face in their daily lives. The integration of the model with the mobile app provides an excellent example of how the technology can be implemented in a user-friendly and accessible way.

Limitations

Scope and Diversity of the Data Set: The variety and scope of the dataset used in training the model can significantly affect the model's ability to generalize. If the dataset is not diverse and comprehensive enough, this may limit the model's ability to detect different types of phishing attacks. The limited diversity of the dataset used in training the model may not adequately represent the different types of threats that the model may encounter in real-world scenarios. Therefore, in order to improve the performance and generalization ability of the model, it is important to use more diverse and comprehensive data sets in the future. Various data sets can consist of different geographies, URLs in different languages, and various phishing tactics. Such diversity can improve the model's ability to recognize and effectively detect a wider range of threats. Furthermore, in order to adapt to the ever-changing cyber threat landscape, the model's training dataset needs to be regularly updated and expanded to include new types of threats. This will allow the model to be more effective against current threats and help it achieve more reliable results in real-world applications. Continuous updating and expansion of the model's training dataset will allow the model to have a broader impact in the cybersecurity space and increase the applicability of the model.

Real-World Applications: The success of the model in a laboratory environment may have different results in real-world applications. Laboratory conditions often involve controlled and confined scenarios, while real-world applications can accommodate much more complex and unpredictable situations. Therefore, thoroughly testing the effectiveness and reliability of the model on real-world data will improve the applicability and overall performance of the model. In real-world applications, the various challenges that the model may face include variable data quality, different user behaviors, and various cyber threat scenarios. These factors can directly affect the performance of the model in a laboratory environment and require

extensive testing to understand how the model behaves in real-world conditions. Real-world testing is vital for assessing how the model reacts to different data sources, user interactions, and cyber threats. These tests can reveal the strengths and potential weak points of the model, thus providing guidance for further development and optimization of the model. Furthermore, these tests provide valuable insights into how the model can be applied in real-world scenarios and allow the model to have a broader impact in the cybersecurity space. Therefore, thoroughly testing the effectiveness and reliability of the model on real-world data is a critical step to improve the success and applicability of the model.

Scope of Detection: The model's phishing detection is mainly based on URL analysis, but this approach creates some limitations in pinpointing phishing sites. Advanced phishing attacks can go undetected by URL analysis alone, especially if attackers use sophisticated masking techniques and dynamic URL structures. Such attacks can bypass the model's URL-based detection mechanisms and require a more comprehensive security solution. . For example, websites that appear legitimate but contain malicious content may not be easily detected by URL analysis. In addition, the constant change of URLs used in phishing attacks and multi-layered threat structures can limit the detection capability of the model. These limitations may require a combination of content analysis, user behavior monitoring, and machine learning techniques to increase the detection capability of the model. Such integration will allow the model to detect advanced phishing attacks more effectively and have a broader impact in the cybersecurity space.

Dynamic and Evolving Threats: Phishing attacks are constantly evolving by developing new methods and strategies, and in this process, new and different URLs are constantly created by attackers. This runs the risk that the dataset used to train the model will not cover these new and changed URLs. This shortcoming can limit the model's ability to detect the most up-to-date threats and reduce the model's effectiveness. In order for the model to remain effective against ever-evolving threats, it needs to be regularly updated and trained with new data sets. This continuous updating and training process ensures that the model can handle new and different URLs, thereby increasing the model's sensitivity to current threats. However, the fact that the model automatically updates and learns against these ever-changing threats comes with technical and algorithmic challenges. These challenges can include resource-intensive processes such as continuous data collection, processing, and model retraining. These processes require significant resources, both time and cost, and managing these resources effectively is critical to ensuring that the model remains up-to-date and

effective on an ongoing basis. Therefore, the integration of dynamic data sets, continuous learning mechanisms, and effective resource management strategies is of great importance in order for the model to maintain its effectiveness against ever-evolving cyber threats. This integration will ensure that the model remains relevant and effective in the ever-changing cybersecurity landscape, allowing it to have a broader impact on the cybersecurity space.

Screenshots and GitHub Repository

The source code of our project and related materials have been shared on GitHub as open source for the cybersecurity community to review, develop and implement. These resources can be accessed at [https://github.com/erendogan6/GuvenliTik-AI]. Open-source access increases the transparency of the project and encourages a wide range of users and developers to contribute. This sharing aims to contribute to the continuous development of the project and create a wider impact in the field of cybersecurity.

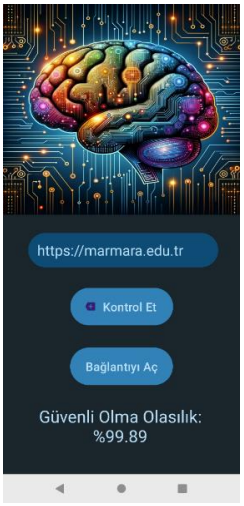

FIGURE 5

7. DISCUSSION AND CONCLUSION

Our project centered around the critical question of whether CNN-based models are effective tools for phishing detection. The results are compelling. High accuracy and other performance metrics not only demonstrate the model's capability but also validate the approach's efficacy. By achieving substantial success in identifying phishing URLs, the model illustrates the practical utility of CNNs in cybersecurity tasks. The findings resonate with the primary research question, providing strong evidence that CNN-based models are not only effective but also reliable in distinguishing between malicious and legitimate URLs. This

confirmation is a testament to the quality of the implemented solution and sets a benchmark for similar cybersecurity efforts.

The model's reliability and effectiveness have been illuminated through rigorous qualitative and quantitative analyses. The integration with a mobile application showcases the model's real-world applicability, offering users a tangible tool against phishing threats. This aspect of the project highlights the model's adaptability to various scenarios - a testament to its robust design and training. The successful attainment of our project's objectives is reflected in the model's consistent performance, even when faced with the complexities and unpredictabilities of real-world data. This adaptability and reliability mark a significant step forward in bridging the gap between theoretical AI models and practical, user-focused cybersecurity solutions.

The overarching ambition of this project was to enhance cybersecurity measures through innovative AI techniques. The model's impressive performance and the subsequent impact on user security reaffirm this goal. It stands as a significant contribution to the field of cybersecurity, reflecting the model's potential as a formidable tool against cyber threats. The strengths demonstrated in accuracy, generalization, and adaptability underscore the model's value and versatility. By providing a reliable and effective tool for phishing detection, the project not only advances the technological frontier of cybersecurity but also offers a proactive measure to safeguard individuals and organizations against the ever-evolving landscape of cyber threats. In essence, the model embodies the project's commitment to enhancing cybersecurity, embodying both the technological advancement and the practical application needed to tackle modern security challenges.

The following discussion and suggestions for future work are predicated on the outcomes and learnings of the current study. They aim to chart a path forward, addressing the limitations encountered and capitalizing on the vast opportunities for advancement in this domain. As we navigate through the complexities of malicious site detection, it is essential to build upon the existing knowledge while innovatively tackling the challenges that lie ahead. The subsequent section delineates a series of strategic recommendations and considerations that can significantly propel the effectiveness and efficiency of future endeavors in this ever-important field.

Multiple Dataset Integration: Future studies can benefit from the integration of multiple datasets to enhance the model's robustness and generalizability. By training and testing on a

diverse array of data sources, the model's ability to understand and classify URLs from various origins can be significantly improved.

Incorporating Additional Features: Beyond just analyzing URLs, future iterations of the model could incorporate a wider range of features such as the content of the website, IP address, SSL certificate status, and other metadata. This would allow for a more comprehensive analysis of websites, potentially improving the accuracy and reliability of malicious site detection.

Utilizing External Validation Sets: Instead of partitioning a single dataset into training and test sets, it's advantageous to validate the model against completely independent datasets obtained from different sources. This approach will better assess the model's performance and generalization capabilities across varied data environments.

Hyperparameter Optimization Expansion: While this project tested hyperparameter optimization with 10% of the data, future efforts could explore a broader range of hyperparameters and test them with larger portions of the data. By expanding the scope and scale of hyperparameter tuning, the optimal configuration for the model can be more accurately identified, potentially leading to improved performance.

Exploring Alternative Models: While a Convolutional Neural Network (CNN) was used in this study, there are many other models worth exploring, including different architectures of neural networks or even ensemble methods. Each model comes with its own set of advantages and trade-offs, and comparing their performance could yield interesting insights and possibly more effective solutions.

Enhanced Computing Resources: Model training and hyperparameter optimization are resource-intensive tasks. Utilizing more powerful computing resources can significantly reduce the time required for these processes and enable the use of more complex models or larger datasets. Future work could leverage high-performance computing or cloud resources to explore more extensive hyperparameter spaces and train more complex models.

By addressing these areas, future work can significantly extend the capabilities and performance of the current model, offering a more robust, accurate, and versatile tool for malicious site detection. These suggestions aim to build upon the foundation laid by this study, pushing the boundaries of what's possible in website security and integrity assessment.

8. REFERENCES

[1] Turkey Statistical Institute (TÜİK). (2023). Household Information Technology (IT) Usage Survey. Retrieved on October 30, 2023, from https://data.tuik.gov.tr/Bulten/Index?p=Hanehalki-Bilisim-Teknolojileri-(BT)-Kullanim-Arastirmasi-2023-49407

[2] Kaya, M. N. Ö. V. A. (2013). Siber Güvenliğin Milli Güvenlik Açısından Önemi ve Alınabilecek Tedbirler. *Güvenlik Stratejileri Dergisi*, 9 (18), 145-181. Retrieved October 30, 2023, from https://dergipark.org.tr/tr/pub/guvenlikstrtj/issue/7525/99154

[3] Bıçakcı, S., Ergun, F. D., Çelikpala, M., (2016). Türkiye'de Siber Güvenlik. *Türkiye'de Siber Güvenlik ve Nükleer Enerji* (s. 28-74). İstanbul: Edam. Retrieved October 30, 2023, from https://edam.org.tr/turkiyede-siber-guvenlik-ve-nukleer-enerji/

[4] ÜNVER, M.; MIRZAOĞLU, A. G. Yemleme ("Phishing"). *Bilgi Teknolojileri ve İletişim Kurumu*. Ankara, 2011. Retrieved October 30, 2023, from https://www.academia.edu/24841831/Yemleme_Phishing

[5] BUBER, Ebubekir; DIRI, Banu; SAHINGOZ, Ozgur Koray. Detecting phishing attacks from URL by using NLP techniques. In: *2017 International conference on computer science and Engineering (UBMK)*. IEEE, 2017. p. 337-342. Retrieved October 30, 2023, from https://ieeexplore.ieee.org/abstract/document/8093406

[6] HEKIM, Hakan. Oltalama (Phishing) Saldırıları. Retrieved October 30, 2023, from http://www.academia.edu/35136881/Oltalama_Phishing_Saldirilari

[7] United Kingdom public sector information site (gov.uk). *Cyber Security Breaches Survey* (2021). Retrieved on October 30, 2023, from https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2021/cyber-security-breaches-survey-2021.

[8] United Kingdom public sector information site (gov.uk). *Cyber Security Breaches Survey* (2022). Retrieved on October 30, 2023, from https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/cyber-security-breaches-survey-2022.

[9] United Kingdom public sector information site (gov.uk). Cyber Security Breaches Survey (2023). Retrieved on October 30, 2023, from https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023.

[10] JAKOBSSON, Markus; MYERS, Steven. Delayed password disclosure. *ACM SIGACT News*, 2007, 38.3: 56-75. Retrieved October 31, 2023, from https://dl.acm.org/doi/abs/10.1145/1324215.1324228

[11] HOPFIELD, John J. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 1988, 4.5: 3-10. Retrieved November 5, 2023, from https://ieeexplore.ieee.org/abstract/document/8118/

[12] FERREIRA, Ricardo Pinto, et al. Artificial neural network for websites classification with phishing characteristics. 2018. Retrieved November 5, 2023, from http://www.sadil.ws/bitstream/handle/123456789/1946/14.pdf?sequence=1&isAllowed=y

[13] MOHAMMAD, Rami M.; THABTAH, Fadi; MCCLUSKEY, Lee. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 2014, 25: 443-458. Retrieved November 5, 2023, from https://link.springer.com/article/10.1007/s00521-013-1490-z

[14] ZHU, Erzhou, et al. DTOF-ANN: an artificial neural network phishing detection model based on decision tree and optimal features. *Applied Soft Computing*, 2020, 95: 106505. Retrieved November 5, 2023, from https://www.sciencedirect.com/science/article/pii/S1568494620304440

[15] GUPTA, Surbhi; SINGHAL, Abhishek. Phishing URL detection by using artificial neural network with PSO. In: *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*. IEEE, 2017. p. 1-6. Retrieved November 5, 2023, from https://ieeexplore.ieee.org/abstract/document/8343553/

[16] EL NAQA, Issam; MURPHY, Martin J. What is machine learning?. *Springer International Publishing*, 2015. Retrieved November 5, 2023, from https://link.springer.com/chapter/10.1007/978-3-319-18305-3_1

[17] ŞANLİÖZ, Şevki Gani, et al. Makine Öğrenmesi Yöntemleri İle Oltalama Websitesi Saldırı Tespiti. Retrieved November 5, 2023, from http://www.hasanbalik.com/yayinlar/d/27.pdf

[18] Abdelhamid, N., Thabtah, F., Abdel-jaber, H. (2017). Phishing detection: A recent intelligent machine learning comparison based on models content and features. In *2017 IEEE international conference on intelligence and security informatics (ISI)* (pp. 72–77). IEEE. Retrieved November 5, 2023, from https://dl.acm.org/doi/10.1109/ISI.2017.8004877

[19] KALAYCI, Tahir Emre. Kimlik hırsızı web sitelerinin sınıflandırılması için makine öğrenmesi yöntemlerinin karşılaştırılması. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 2018, 24.5: 870-878.Retrieved November 5, 2023, from https://dergipark.org.tr/en/pub/pajes/issue/39683/469468?publisher=pamukkale;

[20] AHAMMAD, SK Hasane, et al. Phishing URL detection using machine learning methods. *Advances in Engineering Software*, 2022, 173: 103288. Retrieved November 5, 2023, from https://www.sciencedirect.com/science/article/pii/S0965997822001892?casa_token=eqYR9bMqv34AAAAA:LkzF-g-7NYsrcAXHZ0MGKX2_II7twV6LuDiDe_DN9UfotJ8AgvVk7KUUrVb9HtvdoBZSaCIJ4Z4[21] SAHINGOZ,

[21] Ozgur Koray, et al. Machine learning based phishing detection from URLs. Expert Systems with Applications, 2019, 117: 345-357. Retrieved November 5, 2023, from https://www.sciencedirect.com/science/article/pii/S0957417418306067?casa_token=GyLHDvl4-_8AAAAA:sm0A3tpHQs9O0xe-NQAwW__tFGB1jjwRsokKS6UzdjZiSdhlh6k_m0iaBpVw5KrVgoajSxqzQMg

[22] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. nature, 2015, 521.7553: 436-444.Retrieved November 5, 2023, from https://www.nature.com/articles/nature14539

[23] Benavides, E., Fuertes, W., Sanchez, S., & Sanchez, M. (2020). Classification of phishing attack solutions by employing deep learning techniques: A systematic literature review. In *Developments and advances in defense and security* (pp. 51–64). Springer. Retrieved November 5, 2023, from https://www.academia.edu/43588689/Classification_of_Phishing_Attack_Solutions_by_Employing_Deep_Learning_Techniques_A_Systematic_Literature_Review

[24] GÜL, Ferdi. Derin Öğrenme Tabanlı Oltalama Saldırıları Tespit Sistemi. 2021. PhD Thesis. Marmara Universitesi (Turkey). Retrieved November 5, 2023, from https://search.proquest.com/openview/9c6b34b735f8716a2a223bb79a1c907f/1?pq-origsite=gscholar&cbl=2026366&diss=y

[25] Korkmaz, M. (2023). Oltalama Saldırılarının Derin Öğrenme Tabanlı URL ve İçerik Analizi ile Hibrit Tespiti (Doktora tezi). Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü. Retrieved November 5, 2023, from https://tez.yok.gov.tr/UlusalTezMerkezi/TezGoster?key=G_oJ1rKE4SgJUkomyAKpR8egLeOceXiog6ewkrOjyjdpFJ6nTIAiuqYgChTJBhqE

[26] CHOWDHARY, KR1442; CHOWDHARY, K. R. Natural language processing. *Fundamentals of artificial intelligence*, 2020, 603-649. Retrieved November 5, 2023, from https://link.springer.com/chapter/10.1007/978-81-322-3972-7_19

[27] HAYNES, Katherine; SHIRAZI, Hossein; RAY, Indrakshi. Lightweight URL-based phishing detection using natural language processing transformers for mobile devices. *Procedia Computer Science*, 2021, 191: 127-134. Retrieved November 5, 2023, from https://www.sciencedirect.com/science/article/pii/S1877050921014368

[28] BUBER, Ebubekir; DIRI, Banu; SAHINGOZ, Ozgur Koray. NLP based phishing attack detection from URLs. In: *Intelligent Systems Design and Applications: 17th International Conference on Intelligent Systems Design and Applications (ISDA 2017)* held in Delhi, India, December 14-16, 2017. springer international Publishing, 2018. p. 608-618. Retrieved November 5, 2023, from https://link.springer.com/chapter/10.1007/978-3-319-76348-4_59

[29] Bozkır, A. S. (2022). grambeddings [Veri seti]. Hacettepe Üniversitesi. Retrieved November 26, 2023, from https://web.cs.hacettepe.edu.tr/~selman/grambeddings-dataset/

[30] ALBAWI, Saad; MOHAMMED, Tareq Abed; AL-ZAWI, Saad. Understanding of a convolutional neural network. In: *2017 international conference on engineering and technology (ICET)*. IEEE, 2017. p. 1-6. Retrieved November 26, 2023, from https://ieeexplore.ieee.org/abstract/document/8308186?casa_token=2L4LKJGaonYAAAAA:S7Rai30NlWp0fKb9dE9yCCe-gdthmEJm2oztyOtcD2iIXDNiX71SsLyEi2dExYUJqmMfxI6z-N0