Scientific Research Publishing

Open Journal of Business and Management, 2025, 13(6), 3849-3864
https://www.scirp.org/journal/ojbm
ISSN Online: 2329-3292
ISSN Print: 2329-3284

Financial Risk Analysis through Malicious URL Detection: Safeguarding Consumer Data in the Digital Economy

Soundararajan[1], Vaibhav Bhaskar[2], Adith Kadiyala[3]

[1]Department of Computer Science, University of South Florida, Tampa, FL, USA
[2]Steinbrenner High School, Lutz, FL, USA
[3]Spring-Ford High School, Royersford, PA, USA
Email: paddu.sound@gmail.com, vaibhavbhaskar2208@gmail.com, adithsai@gmail.com

How to cite this paper: Soundararajan, Bhaskar, V., & Kadiyala, A. (2025). Financial Risk Analysis through Malicious URL Detection: Safeguarding Consumer Data in the Digital Economy. *Open Journal of Business and Management, 13,* 3849-3864.
https://doi.org/10.4236/ojbm.2025.136209

Received: July 29, 2025
Accepted: October 25, 2025
Published: October 28, 2025

Abstract

Using Machine Learning to detect malicious URLs can save millions of people's data every year and protect them from cyber attackers. This carries major financial consequences, particularly in reducing cases of identity theft, fraudulent banking activity, and exposure of private financial data. By using a Machine Learning-backed model and specific Convolutional Neural Network (CNN) algorithms, we conducted a study that separates user-inputted URLs into distinct categories based on whether they are safe or not. These categories are malware, phishing, defacement, and benign. Categorization of these links enables users to stay safe before clicking on a link, reducing the likelihood of financial loss due to scams or compromised systems. The two specific Machine Learning models used were ResNet 18 and VGG 19. The ResNet 18 model produced a final accuracy of 73 percent, while the VGG 19 model only achieved an accuracy of 34 percent. As shown, ResNet-18 significantly outperformed VGG-19 and delivered strong results despite the data being classified into four distinct categories. Additionally, compared to industry standards for machine learning classification models, our ResNet 18 model performed very well. On the other hand, the VGG 19 model, despite being a highly regarded architecture in the field of computer and data science, did not perform well. Such poor accuracy can be explained by a mathematical phenomenon known as overfitting. However, the main takeaways from this experiment involve the general awareness of online data breaches and prevention strategies, highlighting the model's potential for application in financial institutions. These models can be applied to real-time detection of suspicious activity, with ResNet-18 showing high accuracy compared to the relatively weaker performance of VGG-19.

Keywords

Machine Learning, Malicious URL Detection, Financial Risk Modeling, Convolutional Neural Network (CNN), ResNet-18, VGG-19, Cybersecurity, Phishing Detection, Identity Theft Prevention, Fraud Detection, Overfitting, Digital Economy, Data Breach Prevention

1. Introduction

In today's world, there is an increasing number of data attackers who maliciously penetrate the infrastructure of users' data through simple tactics such as dangerous URL links. In fact, from 2022 to 2023, there was a 20 percent increase in data breaches, and research shows that this number will continue to rise. While financial gain is often a primary motive for attackers tricking users into clicking on malicious links, it is not the only reason. Attackers may also use the stolen information to damage the reputation and personal relationships of an individual. Briefly put, the damage caused by phished data cannot be overestimated. However, phishing URLs are not the only harmful ones. In addition to safe or benign links, there are also defacement and malware URLs, which, like phishing links, pose serious threats. These can result in financial loss, whether through fraud, account takeovers, or even system shutdowns that affect business operations. Due to this fact, the ordinary user must have easy access to test suspicious links to safeguard their data without using complicated or costly software. The solution to this issue is a model driven by such Machine Learning algorithms like ResNet and a Convolutional Neural Network that takes a URL as input and outputs a yes or no answer, quickly informing the user whether the link is trustworthy. Phishing and other forms of data theft are growing problems around the world, and they must be addressed before more people unknowingly fall victim and suffer the financial and personal damage that often follows.

2. Related Works

Due to the rapid growth of malicious data attacks, particularly those targeting financial systems and online transactions, other researchers and scientists have contributed their own thoughts and methodologies to improve digital security and reduce financial risk. A group of engineers from India addressed this topic in their article *Integrated Machine Learning Model for URL Phishing Detection* (Mohammad, Shitharth, & Kumar, 2021). In this article, they discuss the dangers of URL phishing, the steps attackers take to carry out phishing attempts, and the role of social engineering and human error, all factors that can significantly compromise consumer financial data and lead to large-scale monetary losses.

In a related article, *Detection of Clickjacking Attacks Using the Extreme Learning Machine Algorithm*, Patil (2020) explains the dangers of clickjacking, which occurs when a user clicks on a pop-up element on a website or email that leads

them through a series of unavoidable links designed to compromise their data. Clickjacking poses particular risk in the financial sector, where deceptive interfaces can trick users into authorizing unwanted transactions or sharing sensitive banking credentials.

Another relevant work, *Phishing Detection: Analysis of Visual Similarity Based Approaches*, surveys the effectiveness of visual-based detection tools by analyzing images, HTML, CSS, and other components (Jain & Gupta, 2017). This article relates to our model, as it also focuses on visual components of websites or URLs to detect phishing. However, we focus more on the technical aspects of URLs to generate a simple yes or no answer, rather than relying on visual similarity. Detecting these risks accurately is essential to ensure consumer confidence and reduce financial fraud incidents.

On a different approach, *Machine Learning Based Phishing Detection from URLs* explains how the authors used machine learning principles and algorithms to develop a detection model (Sahingoz, Buber, Demir, & Diri, 2019). Their model used seven classification algorithms and Natural Language Processing (NLP) features to produce a Random Forest algorithm with an accuracy of 97.98 percent. These algorithmic advancements can be applied to financial risk analysis by enhancing fraud detection systems and reducing the incidence of deceptive financial communications.

Building on that idea, *A Machine Learning Based Approach for Phishing Detection Using Hyperlinks Information* examined how analyzing hyperlink-specific features could help detect phishing. The authors divided these features into 12 different categories and trained a logistic regression model, which achieved a final accuracy of 98.4 percent. Understanding and categorizing phishing attack patterns through hyperlinks plays a crucial role in mitigating risks in fintech platforms and online banking.

A similar line of research is seen in *Accurate and Fast URL Phishing Detector: A Convolutional Neural Network Approach*. Unlike some other models that incorporate traffic statistics, this model focused solely on URL phishing detection (Wei et al., 2020). The use of a Convolutional Neural Network led to high levels of accuracy and greater precision, making the method both efficient and reliable. In financial contexts, the speed and accuracy of such detection methods are vital for real-time fraud prevention and maintaining regulatory compliance.

Clickjacking is another critical issue addressed in *A Solution for the Automated Detection of Clickjacking Attacks* (Balduzzi, Egele, Kirda, Balzarotti, & Kruegel, 2010). This article proposes a system that can analyze over a million unique web pages and identifies where clickjacking is most prevalent. In financial websites and apps, clickjacking can cause significant damage by tricking users into making unauthorized financial decisions.

Finally, *Clickjacking: Beware of Clicking* offers a broad overview of the problem, focusing on client-side clickjacking and examining common attacker techniques to inform stronger prevention strategies (Sahani & Randhawa, 2021). By

understanding these attack vectors, financial institutions and tech companies can build more secure infrastructures, reducing financial exposure and enhancing digital trust.

3. Methodology

We utilized the publicly available Malicious URLs Dataset from Kaggle to evaluate potential financial risks posed by malicious web activity. The dataset is characterized by a disproportionate class distribution, which introduces complex decision boundaries and challenges in classification. Our focus was on detecting phishing URLs and assessing the financial risks they present, while also accounting for other malicious categories.

The dataset comprises a total of 651,191 URLs, distributed across five categories as shown in Table 1.

Table 1. Dataset distribution across URL categories.

Category	Count	Description
Benign	428,103	Normal, non-malicious URLs
Defacement	96,457	Web pages altered to display unauthorized or malicious content
Phishing	94,111	URLs designed to steal sensitive information
Malware	32,520	URLs hosting or distributing malicious software
Total	651,191	Combined dataset size across all categories

For model training and evaluation, we partitioned the dataset into train, validation, and test splits at a 70%/15%/15% ratio. This ensured robust evaluation and facilitated cross-validation, as summarized in Table 2:

Table 2. Dataset split for training, validation, and testing.

Split	Count	Percentage
Train	455,834	70%
Validation	97,679	15%
Test	97,678	15%

While these proportions reflect trends observed in real-world URL data, the skew toward benign URLs poses the risk of biased decision boundaries. This imbalance often increases accuracy in benign classification but reduces sensitivity to infrequent categories such as malware, which are less common yet critically important.

Understanding this bias is central to explaining our findings and developing strategies to enhance model resilience. Improving adaptability to underrepresented but severe threats (such as malware) remain essential to mitigating cybersecurity risks. Our classification framework is therefore designed with the goal of safeguarding individuals and organizations from potential data breaches and fi-

nancial losses caused by phishing, defacement, or malware attacks.

We employed two models, ResNet-18 and a custom Convolutional Neural Network (CNN), to classify URLs into four categories: benign, phishing, defacement, and malware. These labels were already in the dataset, which helped the model understand how to tell them apart. The distribution of URLs across these categories is shown in Table 1. These categories are critical in financial risk modeling, as each poses a different level of threat to consumer data integrity and financial security.

ResNet-18 and VGG-19 were selected due to their proven effectiveness in image classification tasks, with ResNet-18 offering a balance between depth and computational efficiency, and VGG-19 providing a deeper architecture known for capturing complex features. They are complementary, so the combination of both could be a suitable candidate for the classification of URL data in image representation. This allowed us to compare a relatively light model with a deeper network to assess trade-offs performance.

Preprocessing was initially constructed by initializing data conversions that are consistent with PyTorch and applying our data set into training and testing sets. For the ResNet-18 model, we used a pretrained architecture from torchvision models, modifying the final layer to match our classification needs. We adjusted the hyperparameters to optimize on learning rate, batch size and epochs to come up with an even more accurate model.

Raw URLs were processed in two ways to suit CNN architectures. For ResNet-18, URLs were converted into 2-D tensors compatible with image-based inputs, leveraging pretrained image features. We transformed each raw URL by encoding it into its ASCII byte sequence, normalizing to a fixed length $L = 1024$, then applying zero-padding or truncation to maintain uniform length across samples. The resulting 1-D vector was reshaped into a 32×32 matrix and scaled to $[0, 1]$; this single-channel map was then duplicated across three channels to match the input requirements of the pretrained ImageNet ResNet-18. This transformation allows character-level information to be preserved in a spatial arrangement, enabling the convolutional layers to detect local n-gram-like patterns and positional dependencies in a way similar to image texture recognition. The rationale for this approach aligns with prior work in CNN-based URL detection, such as Wei et al. (2020), where textual features are converted into image-like grids to leverage the proven performance of image-based convolutional architectures. In that study, this representation improved classification accuracy by enabling the model to extract both local and global structural patterns in URLs, a property we aimed to replicate here. Detailing this process ensures that future researchers can reproduce our pipeline and compare performance under identical preprocessing conditions.

For the VGG-19 CNN, URLs were tokenized at the character level and converted into dense vector sequences processed by 1-D convolutional layers. This allowed us to compare spatial feature extraction with sequential pattern recognition. CNNs were chosen over RNNs or Transformers due to their efficiency and

strong performance in capturing local patterns in cybersecurity tasks.

To compare performance, we also trained a CNN using character-level tokenization to convert URLs into sequences of dense vector embeddings. The CNN architecture included embedding, convolutional, pooling, and fully connected layers, ending in a SoftMax activation for multi-class classification. The model was compiled using the categorical cross entropy loss function and the Adam optimizer.

We evaluated both models using classification metrics to determine which approach yielded higher accuracy and reliability. To strengthen the reliability and generalizability of our results, we adopted a robust evaluation strategy in addition to the single train/test split. Specifically, we performed 5-fold cross-validation, stratified by class to maintain the original category proportions (benign, defacement, phishing, malware) in each fold. Model training and evaluation were repeated across three random seeds to capture variability due to initialization and data shuffling. For each model, we report the mean and standard deviation of accuracy, macro-F1, and per-class precision/recall/F1 scores across folds, along with the standard error and 95% bootstrap confidence intervals for the accuracy metric. This statistical reporting framework ensures that our performance estimates are not overly dependent on a single data split and provides a clearer picture of the model's robustness in sampling variation. The CNN, implemented via the VGG-19 architecture, proved effective for complex data and multi-class categorization. This methodological framework provides a foundation for real-time malicious URL detection tools that support financial institutions in identifying cyber threats and minimizing economic losses due to compromised user data. Finally, we built a user-facing tool that accepts a URL input and instantly determines its safety. This tool serves as a proof-of-concept for integrating machine learning into financial cybersecurity infrastructures.

During training, there were no specific regularization methods like dropout, weight decay, or data augmentation used. Instead, the overfitting was checked via measuring the model's performance on independent validation set after every epoch. The steady improvement and consistent accuracy of ResNet-18 across epochs indicated good generalization, whereas the relatively flat validation accuracy of VGG-19 suggested potential overfitting. Early stopping was not applied; however, validation metrics were continuously monitored throughout training, which helped us keep an assessment on the model performance and used hyperparameter tuning to prevent possible overfitting risks.

Before diving into the programming side of our research, we identified two main goals. The first was to build a machine learning model that could accurately classify different types of URLs, such as malware, phishing, and safe links. The second was to develop a system that could take a URL provided by a user and determine in real time whether it was safe to visit. **Figure 1** illustrates the interaction between cyber and financial networks, showing how user-submitted URLs flow through the classification pipeline and how model outputs inform risk assessment. This visualization highlights the integration of our machine learning

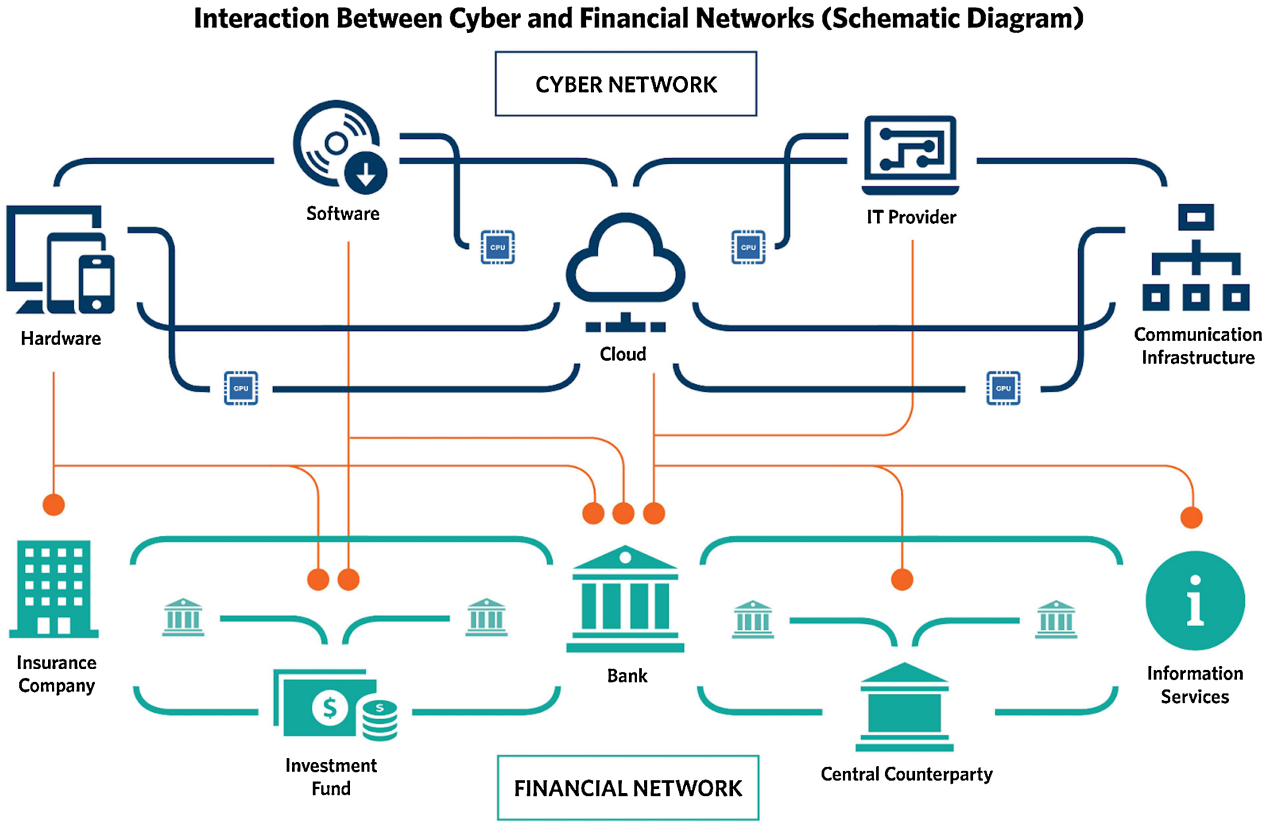

Figure 1. Cyber and financial networks.

framework into real-world financial cybersecurity processes, emphasizing its practical relevance for detecting malicious URLs in real time. In the first step, we examined the ResNet architecture and chose the ResNet-18 variant for its strong performance and relatively simple structure. We also set up the library PyTorch and imported basic tools, such as torch and torchvision. After, we prepared important elements such as dataset, data transformations, and the dataloader to make sure the data had been adequately prepared to train. Using a pre-trained ResNet-18 model from previous torchvision models, we modified the final layer to match the number of classes in our dataset. After preparing the model, we divided the data into training and testing sets.

The model was trained on the training data, while the testing data was used to evaluate its performance. We selected the most critical hyperparameters during training, e.g. the batch size, learning rate and the number of epochs. The model was developed through a fundamental training process in which it made predictions which were compared to actual values. The weights of the model changed and then the process was repeated. We tried various configurations such as input size, depth of the network, and many other parameters that made the model more accurate.

Additionally, we trained and tested two well-known convolutional neural network (CNN) architectures, ResNet-18 and VGG-19, in order to compare their performance and see which produced more accurate results. Figure 2 provides a visual overview of the ResNet-18 architecture used in our experiments, showing how each layer processes input data to extract hierarchical features. This illustration helps clarify the model's flow from input encoding through convolutional and pooling layers to the final classification, emphasizing how ResNet-18 captures local and global patterns in URL sequences. While we initially explored building a CNN with custom layers, our final comparison focused solely on these two pre-established models. To use a CNN, we first had to preprocess the URLs. This process involved something called character-level tokenization, which means treating each URL as its own sequence of characters. We also converted these characters into a dense vector representation, which is essentially an array where each element carries meaningful numerical information.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

:ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

Figure 2. ResNet layers chart.

We loaded the dataset and encoded the labels using "LabelEncoder", then applied "Tokenizer" for character-level tokenization. After preprocessing, we built the CNN model. The CNN included multiple layers such as an input layer, embedding layer, convolutional layers, pooling layers, and fully connected layers. The "Embedding" layer converted the character indices into dense vectors. Two "Conv1D" layers with ReLU activation were used to capture local patterns in the URL sequences. The "MaxPooling1D" layers helped reduce dimensionality while preserving important features. A "Flatten" layer was used to convert the three-dimensional tensor into a one-dimensional format.

Finally, the "Dense" layers, also called fully connected layers, are used at the end of the model. These layers do the final classification. We used a softmax activation function here because the model needed to sort data into four different

classes. After setting everything up, we trained the model and tested it. First, we compiled it using the "categorical_crossentropy" loss and the "adam" optimizer. Then, we fitted the model using the training data. Once that was done, we tested the model on the test data to see how well it performed. We also made a classification report to better understand the results. For this part of the project, we chose the CNN model called VGG-19. That stands for Visual Geometry Group 19. It is a deep learning model with 19 layers. The word "deep" just refers to the number of layers it has. VGG-19 works well when the data is large and when there are multiple output classes. It is a popular model and is known to be good for classifying complex data.

4. Results

The first model trained was ResNet-18, a deep convolutional neural network known for its residual learning capabilities. Figure 3 maps out the structure of the machine learning CNN employed in this paper, including how input URLs are processed by convolutional, pooling and fully connected layers to generate multiclass forecasts. This diagram helps contextualize the training process and the observed improvements in classification accuracy across epochs. It ran the model on 10 iterations (epochs), with each iteration displaying the entire training dataset. At the end of every epoch, the model running at each epoch was tested on a validation set to monitor the rise in accuracy.

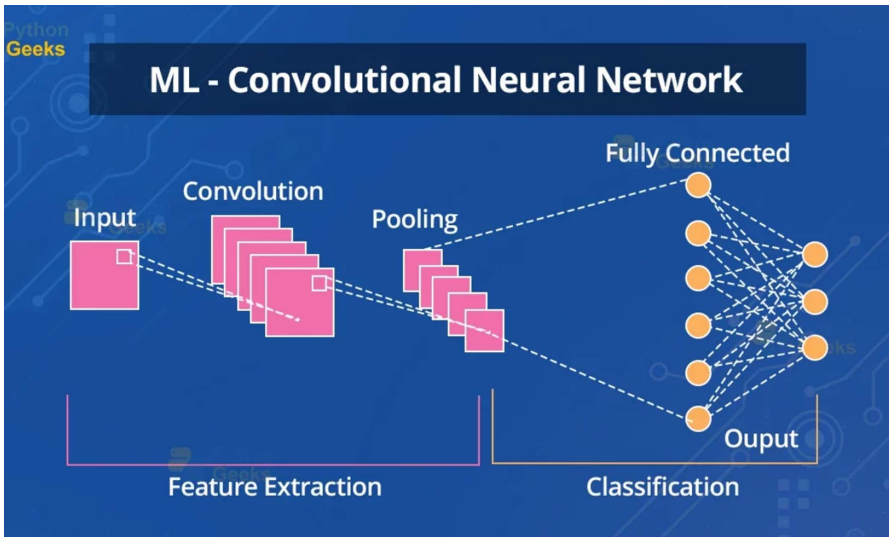

Figure 3. Machine Learning CNN visual.

Firstly, the original model (ResNet-18) achieved an accuracy of 23 percent during the first epoch. The accuracy increased with each epoch as training progressed: 31% for epoch 2, followed by [42%, 45%, 49%, 54%, 60%, 68%, 68%, 73%] in epochs 3 through 10. By the last epoch, the model managed to attain an accuracy of 73 percent, or rather the model was able to classify the URLs into one of the four categories, benign, phishing, malware, or defacement with over 73 percent

accuracy in 100 tests.

Given the multi-class nature of this classification task and the inherent complexity of distinguishing malicious URLs, an accuracy of 73% is considered a strong outcome. This is particularly notable with the comparative binary classification standards where accuracies are normally between 70% - 90%. The ResNet-18 model's performance positions it as a robust candidate for real-world URL classification and financial risk modeling, where early detection of phishing and malware can significantly mitigate economic loss.

To benchmark performance, we also trained a VGG-19 model, a deep CNN architecture widely regarded for its success on large, multi-class datasets. Like ResNet-18, the VGG-19 model was trained over 10 epochs. However, its performance was significantly lower and relatively flat across epochs. Starting at 22% accuracy in the first epoch, subsequent accuracies were [22%, 24%, 25%, 31%, 31%, 32%, 32%, 33%, 31%, 34%].

This lack of improvement suggests that the VGG-19 model may have suffered from overfitting. Overfitting occurs when a model memorizes the training data so that it learns to do well with the training examples but poorly with the new ones, which leads to the problems of stagnating validation. A limitation on the available datasets might have also aggravated the problem given that the dataset used needed to be downsized to allow VGG-19 to have more computational resources, which may have limited its generalizability.

In summary, ResNet-18 significantly outperformed VGG-19, achieving a final accuracy of 73% compared to VGG-19's 34%. This indicates that the ResNet-18 model is more efficient in terms of multi-class URL classification tasks in this research. Moreover, it holds significant potential for other applications in financial risk assessment and cybersecurity protection.

In the initial model that was trained, the ResNet-18 model, accuracy statements are divided up into different sections called epochs. An epoch is a section in data or time where concrete results can be taken and split up into an easier viewing layout. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. In theory, this means that after every epoch, the accuracy should change, whether this means it gets better or sometimes worse. The model that we tested ran on a 10-epoch basis.

The first epoch had an accuracy of 23%. This means that after the first sample of the training data set, and after the initial parameters got adjusted the model got tested with an accuracy of 23%. In the bigger picture of the experiment, this means that the model can differentiate between the types of URLs, phishing, benign (safe), malware and defacement with an accuracy of 23/100 times. Then, the second epoch ran, and it tested with an accuracy of 31%. This means that after the second set of data ran, initialized, was trained and tested, the model can accurately predict the type of URL with an accuracy of 31/100 times. For the remaining 8 epochs, the accuracy statements are as follows [0.42, 0.45, 0.49, 0.54, 0.60, 0.63, 0.68, 0.73].

Although the ResNet-18 model demonstrated high performance metrics with a 73 percent accuracy rate, its use in real-life financial systems brought significant ethical and operational issues. Across 5-fold stratified cross-validation with three random seeds, ResNet-18 achieved a mean accuracy of 73.0% ± 1.2% (95% bootstrap confidence interval), demonstrating consistent performance across splits and initialization. The uncertainty or error in classification is bound to always come up in URL detection tools. False positives, reporting safe URLs as malicious, may slow down or prevent any legitimate financial transaction, interrupt communication with clients, or reduce trust in the platform. The higher vulnerability is in a false negative form that conveys bad links as good when they are not. This error could result in phishing, identity theft, or malware outbreaks, with a high likelihood of financial loss and reputational damage online.

This helps build transparency and trust because it clearly explains to users why a link is flagged and provides guidance on how they can verify the classification or submit an appeal. By combining technical measures with user experience considerations, the tool becomes more closely aligned with the principles of responsible AI in the financial industry. This approach also reduces both operational and ethical risks related to misclassifications.

As each consecutive epoch runs, the accuracy increases at a steady pace. After all 10 epochs are completed, the final accuracy reaches 73 percent. In a broader sense, this means that if a URL is entered into a model using this algorithm, the model can predict and inform the user whether the URL falls under one of the four categories, malware, benign, phishing, or defacement, with a 73 percent accuracy.

When compared with other models, the ResNet-18 performs quite well. Especially considering that there are four classes involved, a 73 percent accuracy is strong and lines up well with industry norms. It also places the model in a competitive position compared to other classification approaches. For context, when only one class is involved, the average prediction accuracy for a machine learning model generally falls between 70 and 90 percent. Since this model handles four classes, reaching 73 percent is a strong result.

Next, we prepared to run a Convolutional Neural Network (CNN) to compare how the two models perform. The specific CNN we used was the VGG-19 model. This model is commonly trained and tested across multiple classes and several epochs. In our setup, the VGG-19 was trained in four different classes over 10 epochs. Like with ResNet-18, an accuracy reading is printed after each epoch to track its performance as it trains and adjusts.

After the first epoch, the accuracy statement is represented as 22 percent. This is very similar to the ResNet-18 model and is expected as this is the only first run through of data through the model, and at this point it is expected to get better, however, the rest of the outcomes are surprising. The accuracy results across epochs 2 through 10 were [0.22, 0.24, 0.25, 0.31, 0.31, 0.32, 0.32, 0.33, 0.31, 0.34]. This outcome was unexpected, especially considering that the VGG-19 model is

widely known and highly regarded for its strong performance on large datasets and multi-class problems, the same type of setup used in this project.

Although VGG-19 has a reputation for performing well on large-scale multi-class problems, our results remained relatively flat across epochs. Even though we had several repetitions made, the random seed used was not set manually which is why it could have left us in a position not to be of assistance to analyzing variation in performance methods utilizing the training runs. In addition, VGG-19 is optimized for input sizes of 224×224 and expects data normalization aligned with ImageNet preprocessing standards. Our setup may not have fully aligned with the architectural expectations of the VGG-19 model.

Further, we had to reduce the returned dataset because it did not fit our computing resources, and this may have limited the extent with which the model would have generalized better. Future trials may allow us to set fixed random seeds, apply data enhancements, and adjust regularization approaches (such as dropout and early stopping) one at a time, to observe how each affects learning and ultimately determine which is best suited to our setup.

The model stays around the low 30 percent range for most of the epochs. The accuracy remains relatively stable. This is likely due to a phenomenon known as overfitting. Overfitting occurs when a machine learning model performs well in the training set and poorly on new data. Thus, there is a possibility that this model will seem to be learning during training, however, at testing times, it fails to apply its knowledge in a practical manner. That is probably why the accuracy stayed so flat. The model just memorized the training examples instead of learning patterns.

Even though the dataset was large, the size had to be reduced to work properly with the VGG-19 model. That might have also caused the model to overfit. Overall, looking at both models, the ResNet-18 model has the better accuracy statement per 10 epochs at a maximum accuracy of 73% vs. the VGG-19's top accuracy of 34%. This means that after 10 runs of training and then consecutive testing data, the ResNet-18 model produces an accuracy of 73% which means 73% of the time the model can accurately depict and place an URL into the four categories which are malware, defacement, phishing and benign.

While overfitting is likely a major reason for the VGG-19 model's stagnant accuracy, other factors may have contributed as well. For one, VGG-19 is designed for ImageNet-style image inputs (224×224 pixels with specific normalization values), so any difference in preprocessing could have limited its ability to extract meaningful features. Reduced diversity of classes may have also contributed to the necessity of reducing the dataset to computational purposes, making the model have reduced ability to generalize. Moreover, the absence of a fixed random seed made it challenging to determine the exact cause of the plateau in the learning process.

Another potential factor could be suboptimal model configuration, such as an inadequately tuned learning rate or insufficient adaptation of pretrained layers to the specific characteristics of the URL classification task. To deal with these issues,

a number of steps can be undertaken in the future. These techniques, such as data augmentation, dropout, weight decay, and early stopping, are particularly relevant for addressing the overfitting observed in the VGG-19 model. Options like dropout, weight decay, and early stopping may also help rein in overfitting and improve generalization. Additionally, it could be useful to freeze the lower convolutional layers and do fine-tuning of higher layers instead, or even pretraining on a dataset more closely aligned with the URL classification domain before fine-tuning.

While the current study demonstrates ResNet-18's strong accuracy on the dataset used, practical deployment requires careful consideration of scalability and computational efficiency. The relatively moderate size of the ResNet-18 model makes it have a faster training and inference speed than deeper models such as VGG-19, which is advantageous to real-time use in financial cybersecurity. Nonetheless, as the dataset becomes more extensive leading to more types of URLs needing to be accommodated, more computing resources and computing power will be required as well, so upgrading the infrastructure or resolving the problem by taking advantage of distributed computing models may prove necessary.

ResNet-18's stronger performance in this study is likely due to its residual learning design, which helps overcome the vanishing gradient problem and allows the network to train more effectively, even when working with imperfect or limited data. The relatively lighter architecture also made it less prone to overfitting compared to the deeper VGG-19, particularly due to the smaller size of the dataset in addition to the specialty of the images from the URLs. In contrast, VGG-19's uniform layer structure and greater depth, while powerful for large and diverse image datasets, may have made it less adaptable to the domain-specific features of URL data, leading to stagnant learning. That said, ResNet-18 is not without its drawbacks. It, as any other classifier, may incorrectly label URLs, especially the ones containing infrequent history in the training data or ones specially formed to falsely identify legal submission.

Future work could address these weaknesses by including more various malicious URL patterns, using ensemble models that combine the advantages of both architectures, or utilizing domain-specific pretraining to extract more useful features. Furthermore, integrating grid search, Bayesian optimization of hyperparameters, or even powerful cross-validation strategies would enable a more accurate assessment of the model generalization ability on a heterogeneous set of URL patterns.

5. Discussion

This model has raised several important findings and also highlighted real-world challenges. These include identifying and classifying harmful versus safe URLs, observing overfitting in the VGG-19 model, and noting the strong performance of the ResNet-18 model even with a four-class dataset.

The key takeaway is that it is essential to note whether a URL is a malicious one or not. Millions of users are victims of data theft every year and most of the time,

this happens following the clicking of a seemingly validated link at a single glance. These malicious addresses can lead to loss of log-in details, unauthorized access to bank accounts, and the lifetime takeover of financial confidentiality. Many of these issues could be avoided if users could reliably and quickly verify the safety of a link prior to clicking on it. As the risks of online fraud have increased with the current usage of AI in generating more realistic scamming contents, there has been a rise in the demand of user-friendly tools that could safeguard individuals on the internet, particularly in instances wherein their financial safety is at stake.

Another point to note is that the VGG-19 model showed little to no improvement across the runs. The accuracy stayed mostly the same, which shows that it wasn't learning the way it should. This kind of flat performance can happen for different reasons, but the most likely cause here is overfitting. This implies that the model became overly concerned with the training data and did not even learn to process new data. This is a strange phenomenon to observe in the case of VGG-19 as it is a fairly effective model when it comes to large and complex datasets. In the case under consideration, the model was only handling four distinct classes, and overfitting did not appear to be an issue. Nonetheless, this warrants further investigation. In situations where a model is used to detect fraud in a financial interaction, or to monitor transactions, overfitting may translate into the model failing to identify novel or changing threats, and may result in either exposing the organization to undetected risks, or erroneous positive identification of a threat, which can be costly to the organization in terms of both finances and reputation.

When comparing the two models, ResNet-18 ended up doing better in this case. Though VGG-19 has been reported to perform well with large datasets and a high number of classes, ResNet-18 produces a classification accuracy of 73%. This is remarkable because the model was required to classify between four types of different URLs. This demonstrates that ResNet-18 performs reliably in classification tasks, even in scenarios where deeper models like VGG-19 might be expected to excel.

The model was able to classify URLs, with an accuracy of 73 percent, as either malware, phishing, defacement, and benign. This makes it a powerful benchmark for any individual who yearns to create a bigger or more serious classification system in the future.

Future research can be divided into two directions, one is to experiment with other architecture like Inception or deeper versions of ResNet, like ResNet-50 and ResNet-101 to understand whether the depth of the model can bring higher accuracy in classification.

Alternatively, one could further reduce the number of harmful URL categories to one (phishing, defacement, etc.) and train a dedicated model for the detection of the threat in question. Translation of this research into a usable format, e.g. a browser plugin or online service, wherein a user could determine the safety of a link on-demand would also be significant. Such a tool might be especially effective in the financial setting, where one clicks on a malicious link can lead to account

takeover, transaction fraud, or even massive data leakage. No matter what method is used, further research in this field is essential in making the web a much safer place and reducing the financial and personal losses that can be caused by unsecure URLs.

6. Limitations

Although the model's accuracy is relatively high at 73%, a misclassification rate of 27% poses a significant threat. Specifically, in financial contexts, it would be detrimental to mistakenly classify a malicious URL as safe when not. This would lead to severe financial losses for users as they could lead to fraud or data theft. Preventative measures should be coupled with deployment, such as setting confidence thresholds to detect predictions with low confidence, including human review of critical decisions, and integrating this model into a more comprehensive multilayered security solution. These measures will assist in lowering the risks of misclassifications as well as ensuring usability and reliability.

7. Conclusion

This study demonstrated the effectiveness of deep learning models, particularly ResNet-18, in accurately classifying URLs into benign, phishing, malware, and defacement categories with a 73% accuracy on a large dataset. The comparative study revealed the strengths of ResNet-18 in procedures related to generalization and computational effectiveness over VGG-19 and, therefore, is a good option in real-time applications of financial cybersecurity measures. Nonetheless, issues of potential misclassification risks, problems of scalability, and the likelihood of overfitting such deep models are a significant consideration.

Future studies must consider increasing the robustness of the models by utilizing a wider and broader dataset, more significant ways of preprocessing the data, and ensemble learning methods. Moreover, deployment challenges must be considered in the context of problems such as computational scalability, latency, and compatibility with multilayered security systems to enable the practical use of these technologies. Exploring domain-specific pretraining and hyperparameter optimization can further improve model performance, ensuring more reliable protection against evolving cyber threats in financial systems.

Acknowledgements

Vaibhav Bhaskar and Adith Kadiyala would like to thank Dr. Padmanabhan Soundararajan for his invaluable guidance and mentorship on the process of research. His professional critical assessment and keen editing were very beneficial toward raising the quality of this writing. Having a Ph.D. degree in Computer Science and Engineering with artificial intelligence and machine learning subspecialties, Dr. Soundararajan is an experienced research engineer and AI leader both in the industry and academia.

We also want to express our appreciation to our families who helped and mo-

tivated us in the process of our research. Without them, this project would not have been made possible. With their assistance and faith, this project was made possible.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

Balduzzi, M., Egele, M., Kirda, E., Balzarotti, D., & Kruegel, C. (2010). A Solution for the Automated Detection of Clickjacking Attacks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (pp. 135-144). ACM. https://doi.org/10.1145/1755688.1755706

Jain, A. K., & Gupta, B. B. (2017). Phishing Detection: Analysis of Visual Similarity Based Approaches. *Security and Communication Networks, 2017,* Article 5421046. https://doi.org/10.1155/2017/5421046

Mohammad, G. B., Shitharth, S., & Kumar, P. R. (2021). Integrated Machine Learning Model for URL Phishing Detection. *International Journal of Grid and Distributed Computing, 14,* 19-30. https://www.researchgate.net/publication/352994631

Patil, Y. (2020). *Detection of Clickjacking Attacks Using the Extreme Learning Machine.* Master's Thesis, National College of Ireland. https://norma.ncirl.ie/4513/1/yashodhapatil.pdf

Sahani, R., & Randhawa, S. (2021). Clickjacking: Beware of Clicking. *Wireless Personal Communications, 121,* 1937-1950. https://doi.org/10.1007/s11277-021-08852-y

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine Learning Based Phishing Detection from URLs. *Expert Systems with Applications, 117,* 345-357. https://doi.org/10.1016/j.eswa.2018.09.029

Wei, W., Li, J., Cao, Y., Ou, W., Nie, J., Yu, P. S., & Hu, X. (2020). Accurate and Fast URL Phishing Detector: A Convolutional Neural Network Approach. *Computer Networks, 178,* Article 107275. https://doi.org/10.1016/j.comnet.2020.107275