DESIGN AND IMPLEMENTATION OF A MACHINE LEARNING FRAMEWORK FOR BLOCKED URL DETECTION

***V.NARESH,** *Lecturer in Computers,*
KIMS PG COLLEGE, KARIMNAGAR, TG.

ABSTRACT: People frequently host harmful content, such as phishing schemes, spam, malicious advertisements, and drive-by vulnerabilities, on malicious URLs, which are also known as malicious websites. Streamlining and clarifying the inquiry. Patience, please, for the following phase. Past research has made use of regular expressions, signature matching, and blacklisting. Neither previously discovered dangerous URLs nor variants of those URLs may be located using these methods. To resolve this matter, a machine learning-based solution could be suggested. This method relies on a large body of work in feature engineering and feature representation of security elements, such as URLs, which requires substantial investigation. Additionally, feature engineering and feature representation tools must be continuously updated to support both the old and new versions of URLs. With the use of deep learning, AI systems can already compete with humans in a wide range of tasks. They are so good at computer vision that they can outperform humans on certain tasks. They are capable of automatically extracting the most useful feature photos from provided raw data. In order to make raw URLs usable and translateable in cybersecurity apps, Deep URL Detect (DUD) use character embedding. A more sophisticated method of numerically representing letters is character-level embedding in natural language processing (NLP). After character-level embedding features are included, hidden layers in deep learning systems utilize a nonlinear activation function to determine if a URL might be hazardous. In order to identify fraudulent URLs, this study examines various deep learning-based advanced character-level embedding algorithms. The optimal deep learning character-level embedding model is determined via a battery of experiments. Every test is executed 500 times at a learning rate of 0.001, and it employs multiple deep learning-based character-level embedding models. DUD outperforms all other deep learning-based character-level embedding approaches in terms of speed and performance across all test scenarios. There was an improvement over n-gram representations when using deep learning systems based on character-level embedding models. This is because the URL's order and connections are preserved by the integration.

Keywords: *Cyber security, Cybercrime, Malicious URL, Machine learning, Deep learning, Character embedding.*

1. INTRODUCTION

Threat actors frequently employ malicious Uniform Resource Locators (URLs) to host and disseminate destructive content. Their functionality is crucial to many hacks. Unfortunately, malicious URLs are all too common on various social media platforms, in emails, and on platforms like Orkut, WhatsApp, and Facebook. Users acknowledge that this data has not been verified or validated. Malicious actors can potentially gain access to the hosting platform if an unauthorized user sees the website using the URL. Scammers who wish to steal personal information or pose as the user will find this user easy prey. Businesses lose billions of dollars annually due to rogue URLs. Having a mechanism to identify potentially dangerous URLs and notify the network administrator of them is more crucial than ever before. Many products on the market today use blacklisting. This strategy is based on a lengthy list of potentially harmful URLs. By conducting frequent checks and gathering additional user information, the antivirus team ensures that the blacklists are accurate. If you use the blacklisting method, you can locate a malicious URL that is already in the database. But they don't seem to be able to detect any newly malicious URLs or variants of problematic ones. The modern fraudster uses a method known as "mutation" to create new malware variants. To address this issue, machine learning techniques are employed.

In recent years, the most popular method for extracting lexical information from URLs has been to use machine learning models to enhance pre-existing domain expertise. A popular model in machine learning is the Support Vector Machine (SVM). The Bag-of-Words (BoW) technique is commonly employed in feature extraction. Machine learning algorithms have a lot of issues, yet they could be a viable alternative to blacklists. Normal methods of displaying URLs have the drawback of not adequately revealing the structure and assembly of the characters that comprise them.

Manually constructed attributes are the foundation of traditional machine learning models. Success in the widely regarded as extremely challenging sector of cybersecurity requires an in-depth familiarity of the industry as a whole.

Due to the system's confusion caused by the test results, private characteristics cannot be recovered. There are an excessive number of words that exceed the capacity of the machine learning system's memory, further complicating its learning process.

This research finds solutions to the previously mentioned issues using the Deep URL Detect (DUD) methodology. It achieves this by integrating deep learning, a state-of-the-art machine learning technique, with character embedding. Deep learning allows models to be formed at different levels of cognition by using many hidden layers to make nonlinear projections. This defensive strategy has multiple potential applications. Some of the project's most significant benefits are as follows:

This research aims to analyze and test several benchmark deep learning designs with the end goal of detecting malicious URLs.

The experimental study evaluates the models' generalizability across contexts by using multiple datasets. Experimental research distinguishes between time-split and random-split approaches to data splitting.

In this paper, we examine various deep learning models that are designed for n-gram encoding and character embedding.

2. RELATEDWORK

We take a look at what's already out there regarding using machine learning techniques to identify potentially dangerous URLs. This section provides a synopsis of the most popular methods for identifying potentially dangerous URLs. Some of the first approaches to detecting malicious URLs included signature matching, regular expression analysis, and blacklisting. We have only covered methods for finding static URLs up until now. Updating the signature library with the newest modifications is crucial for effectively dealing with new threats to URL patterns. Next, we identified potential dangers by using machine learning to discover new sets of uniform resource locators (URLs). In order to incorporate all pertinent URL information, ordinary machine learning algorithms occasionally require the feature engineering method. In cybersecurity, feature engineering is most effective when combined with extensive knowledge about URLs and a hand-picked set of high-quality features. Many published research identified and described hazardous URLs using various criteria. Among these characteristics are those that are content-, reputation-, and context-specific, as well as those that are host-, vocabulary-, and blacklist-dependent. We can learn more about a brick by checking at its Uniform Resource Locator (URL). With any luck, this tool can help you identify and report fraudulent URLs. Judging lexical qualities is done using data from URL strings, like how long they are and whether or not they contain special characters. The hostname component of the URL grants access to the host's in-built capabilities. Here, you might see details like the computer's position, WHOIS details, or IP address. When an unsuspecting user clicks on a malicious URL, the website's content is made using JavaScript and HTML. Among the many characteristics of the content are its standing, notoriety, and birthplace. For this study, the experts painstakingly compiled a list of all the traits and combinations of characteristics used in previous research.

It takes a long time to implement new features since people are worried about security. Acquiring situationally dependent traits, such as those mentioned before, might pose challenges and even dangers. Additionally, domain-specific knowledge and data are crucial throughout feature selection. It was usual practice to obtain this data via analyzing the URL's structure. It is easier to learn new words than other things, according to previously published research. Additional approaches based on frequency include n-grams, term-document matrix (TDM), and term-frequency and inverse document frequency (TF-IDF). It is not possible to deduce the meaning or proper arrangement of URLs from any of these attributes. Here, we disregard the opinions of the anonymous speakers. Remember that an attacker can simply bypass feature engineering and common machine learning techniques used to detect malicious URLs.

We now utilize deep learning algorithms to identify potentially hazardous URLs by assigning a value to each character. The effectiveness of deep learning, character-level embedding, conventional ML techniques, and feature engineering in detecting malicious and fraudulent URLs was evaluated. Deep learning systems outperformed more conventional approaches. Finding malicious URLs is a task for models with long short-term memory (LSTM) and recurrent neural networks (RNNs). Statistical URL analysis and linguistic features were compared using a random forest classifier. The LSTM model outperformed the competition when it came to machine learning. A convolutional neural network (CNN) was trained to identify potentially malicious website names, file paths, and registry keys using a character-level Keras embedding. This study demonstrated that several cybersecurity issues can be addressed by implementing a novel deep learning system. Current market conditions have resulted in a proliferation of deep learning benchmarking tools. The objective of this research is to compare the performance of various deep learning models in URL detection.

3. SYSTEM DESIGN

An Overview Of Uniform Resource Locator (Url)

The portion of a URI that directs a web service to access a specific file is called a Uniform Resource Locator (URL). Figure 1 shows the three components that make up a Uniform Resource Locator (URL). First, select the protocol, which can be "http" or "https." Then, pick an IP address or a domain name. The final section details the necessary steps to reach the desired website, along with any other prerequisites. Following a sequence of forward slashes is the domain name, and following that is the protocol. The route is divided into its component pieces by a single forward slash. Figure 1 shows one possible version of a URL.

Using the URL as the primary source opens the door for an attacker to potentially assist with and conduct harmful acts. Sadly, many people send dangerous URLs in emails and on social media. Unwary users run the risk of hacking the system if they click on malicious URLs. Thus, knowing what a URL is and where it needs to go is crucial. The CNN-LSTM model integrates CNN and LSTM, two systems that mimic the brain, to create a single, cohesive whole. Keras use an embedding technique that processes text character by character.

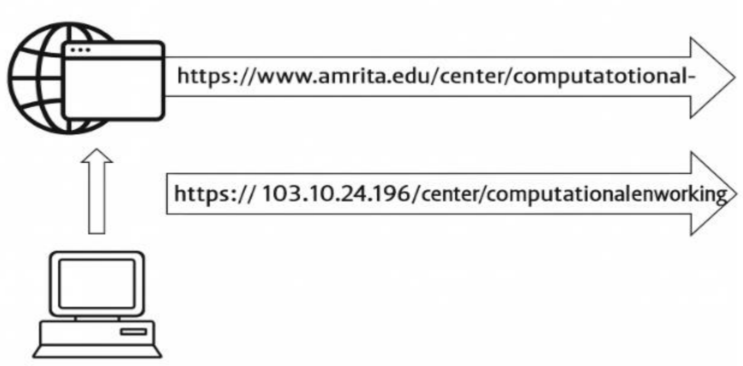

https://www.amrita.edu/center/computatotional-
https:// 103.10.24.196/center/computationalenworking

Figure 1: Uniform resource locator (URL) components

Background details of Deep learning Models

Hybrid architecture –

In order to learn from data, both Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs) employ a convolutional approach.

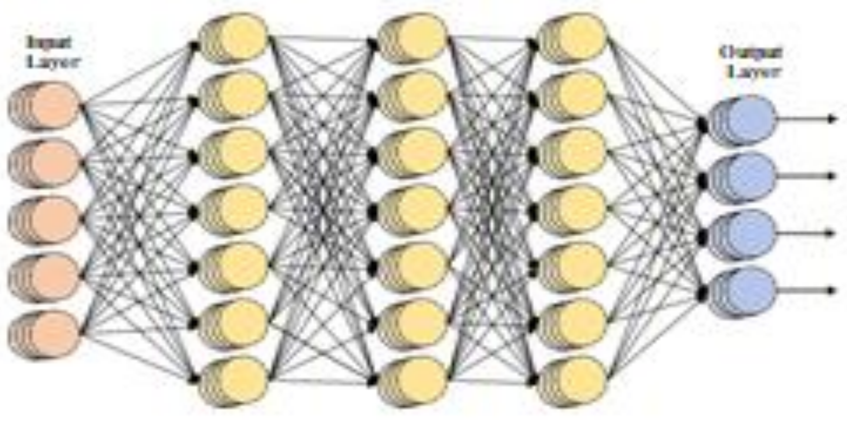


Figure 2: Pictures 2 and 3 show examples of convolutional neural network (CNN) and deep neural network (DNN) designs. C-N-G was a shortened version of CNN. Often referred to as temporal convolution and temporal pooling, CNN-C primarily employs

one-dimensional convolution and pooling methods. The CNN-C model cleverly employs the most advantageous aspects of URL character encoding. The character-level Keras embedding model is utilized for this purpose. To begin, you'll need to know the length of the character vector, the maximum size of the dictionary, and the length of the embedding vector. Keras allows you to initialize the weights of character-level embeddings using a hyperparameter. Back propagation is used to optimize the weights. In order to train the LSTM, data is used from the CNN. This expedites the process of teaching sequence labeling at the character level.

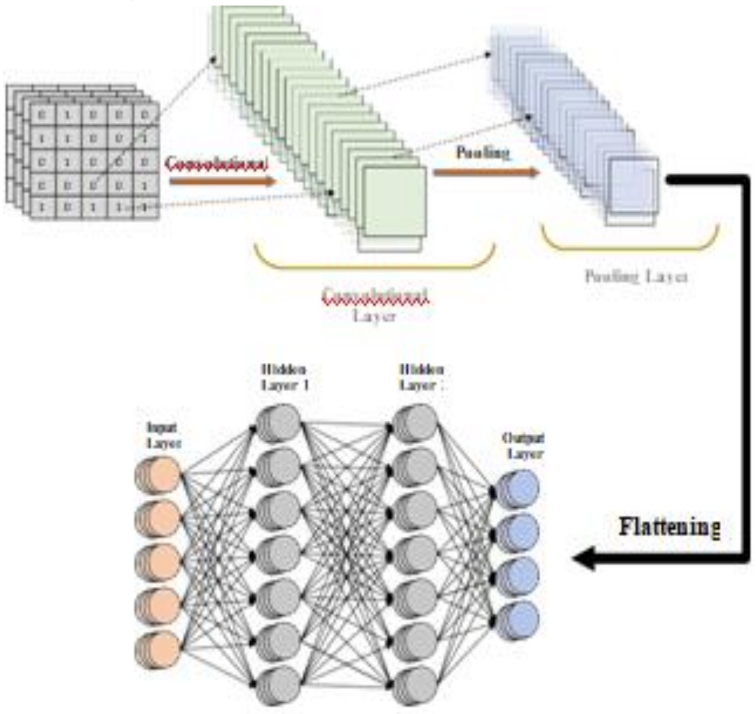


Figure 3: Convolution Neural Network architecture

Character based models

In order to obtain features, character-based models just require character patterns as input. Text classification is one application of the aforementioned characteristics. In order to detect harmful URLs in security software, multiple character-based natural language processing techniques are now under evaluation. Embedding transforms URLs into numerical vectors in the initial layer of every model.

Character level models based on RNN

Endgame Architecture:

This study finds and sorts domain names generated by DGAs using a combination of character-level Keras embeddings and LSTM. Domain names are made more memorable and easier to type with character-level Keras embeddings. Feature engineering, which is a crucial component of conventional machine learning techniques, is also completely ignored. Feature engineering, hidden Markov models, and the use of bigrams with conventional ML techniques are all inferior to the aforementioned approach. This research makes use of a three-layer LSTM network, with an LSTM layer for feature extraction, a logistic regression layer for data classification, and an embedding layer for URL storage. For feature extraction, the embedding layer provides an LSTM with 128-dimensional character representations. The next step is to assign a probability score to each domain name using logistic regression.

CMU Architecture:

Carnegie Mellon University is responsible for the architectural structure known as Tweet2vec. Its primary function is to classify and organize social media data, primarily tweets, through the use of tags. Use of Bidirectional Gated Recurrent Units (BGRUs) allows for the extraction of feature models from Twitter data. This technique converts tweets into "tokens," which are collections of letters. The next step is to define each token using the one-hot character encoding approach. The one-time models are incorporated into the BGRU design after being transformed into a character space. Through the utilization of both forward and backward Gated Recurrent Units (GRUs), the model facilitates comprehension of the domain name's series. Hashtag predictions are made using the tweet categorization model's forward and backward GRU layers. To achieve this, a fully connected layer and the softmax nonlinear activation function can be utilized. The effectiveness of Tweet2vec in comparing different word forms on Twitter is evaluated.

Models constructed at the character level using convolutional neural networks and supervised learning.

Character level models based on CNN

NYU Architecture:

Convolutional neural networks (CNNs) are now the industry standard for this purpose. A new method for text categorization using one-dimensional convolutional neural networks (CNNs) has emerged. The professionals in this area utilized Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The CNN show has a lot of fans at New York University. CNN describes text using a variety of ways, such as search tables, embeddings, and taught embeddings. Text is described using LSTM and pre-trained word embeddings. A vast number of files were examined. The character-level CNN model outperformed the more conventional and deep learning approaches. Common text representation methods used to evaluate deep learning models are Bag-of-Words (BoW) and n-grams with Term Frequency-Inverse Document Frequency (TF-IDF).

Invincea Architecture:

File paths, URLs, and registry entries are all examples of short strings of characters that are good candidates for encoding in convolutional neural networks (CNNs). The architecture of CNN consists of three interconnected layers. This is accomplished on two levels: one using a Keras modeling layer and the other using a parallel convolutional neural network. The activation function is based on Rectified Linear Units (ReLUs), and each of the three completely connected layers contains 1,024 units. To avoid overfitting and speed up training, model developers utilize batch normalization and dropout regularization. The Convolutional Neural Network (CNN) employs a sigmoid nonlinear activation function and a fully linked layer with a single unit to determine the goodness or badness of a given string of characters.

Character level models based on hybrid CNN and RNN

MIT Architecture:

The present approach borrows heavily from the NYU methodology for categorizing tweets. An LSTM layer and layered convolutional neural networks (CNNs) make up the architecture. When using a multilayer convolutional neural network (CNN), overfitting might occur. To simplify this process, a handful of tools are utilized.

Problem Formulation

Classifying a URL as safe or dangerous is the main objective of this project. One of the most challenging tasks is organizing content into meaningful categories. Here are some URLs you should be familiar with: U is a set of tuples, and each tuple has a URL (u) and a label (y) that indicates the level of danger. If the value is zero, then the URL is legitimate. You need to figure out how to characterize features and make predictions well in the first two steps of categorization. An n-dimensional vector representing the features (represented by xn) and a prediction function (yn = sign(f(xn)) are both possible. The primary objective is to lessen the occurrence of individuals being incorrectly classified. One approach would be to eliminate the loss function. To improve the performance of this loss, a regularization term could be useful. In this study, we display the variable f using deep learning models.

Shortcomings in Malicious URL Detection

There is currently a lack of publicly accessible standard datasets dedicated solely to research on malicious URL detection. Up till now, the majority of studies have tested various deep learning frameworks and typical machine learning algorithms for detecting bogus URLs using private datasets. The information used to compile these private databases came from a variety of sources, including DMOZ, Phishtank, Open Phish, Malware Domains, Malware Domain-List, Alexa, and many more. Keep in mind that these approaches do not constitute "generic procedures" since they draw from distinct data sources. The majority of published research fail to detail the data splitting procedure that utilized the time-split method. Recent research has examined the significance of data partitioning into training and testing sets using the "temporal split" technique. When looking for zero-day malware, temporal data splitting is a crucial technique. The lack of widespread deployment of security solutions based on machine learning has recently been the subject of expert investigation. A great deal of detail is provided regarding the many test scenarios that should be considered when studying for the test. The availability of a large number of test cases can facilitate the process of determining the reliability of solutions that rely on machine learning. Another topic that came up was the difficulty of applying data science methodologies to military applications.

4. SYSTEM ANALYSIS

Model Configuration of Malicious URL Detection Engine

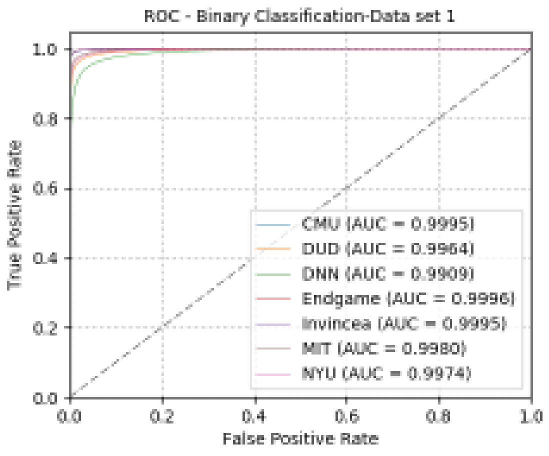
```
Algorithm 1: Malicious URL Detection Engine
  Input: A set of URLs $U_1, U_2, ..., U_n$.
  Output: Deep URL Detect Model Labels $y_1, y_2, .., y_n$ (0: legitimate or 1: Malicious)
          and BenchMarkModels prediction $p$.
1 VectorizedURLs = Dataprocessing(Ui)  // URLs into numerical vectors using
   text representation method
2 Predicions p = BenchMarkModels(VectorizedURL)      // Invincea, NYU, MIT,
   CMU, Endgame
3 for each URLs $U_i$ do
4     $lcURL$= lowerCase($U_i$)
5     ZeroPaddedURL $Z_i$ = Padding($lcURL$)
6     $E$ = Character level Keras Embedding ($Z_i$)
7     $C$ = CNN($E$)
8     $L$ = LSTM($C$)
9     Compute $y_i$ = Sigmoid($L$)
10 end for
```

Table 1: Detailed configuration parameter information of DUD for Data set 1

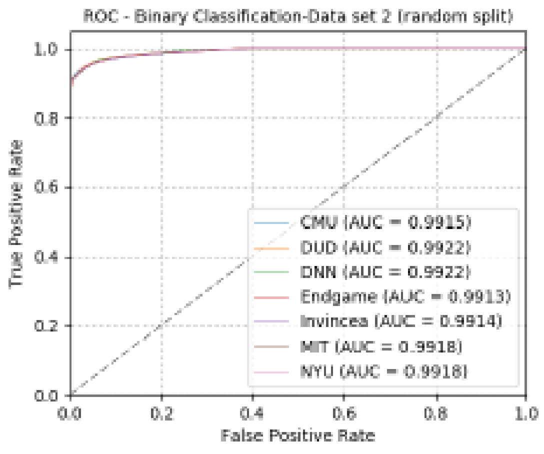
Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 2307, 128)	19200
conv1d_1 (Conv1D)	(None, 2306, 128)	32896
max_pooling1d_1 (MaxPooling1	(None, 1153, 128)	0
lstm_1 (LSTM)	(None, 70)	55720
dense_1 (Dense)	(None, 1)	71
activation_1 (Activation)	(None, 1)	0
Total params: 107,887 Trainable params: 107,887 Non-trainable params: 0		

Table 2: Detailed configuration parameter information of DUD for Data set 2 random-split and time-split

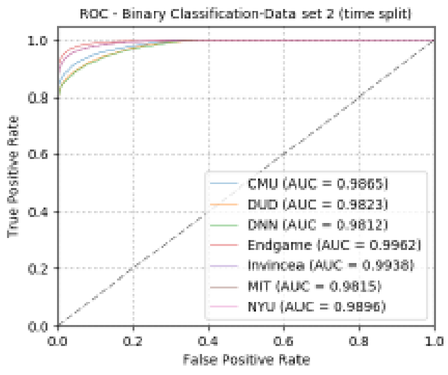
Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 246, 128)	5376
conv1d_1 (Conv1D)	(None, 245, 128)	32896
max_pooling1d_1 (MaxPooling1	(None, 122, 128)	0
lstm_1 (LSTM)	(None, 70)	55720
dense_1 (Dense)	(None, 1)	71
activation_1 (Activation)	(None, 1)	0
Total params: 94,063 Trainable params: 94,063 Non-trainable params: 0		



(a)

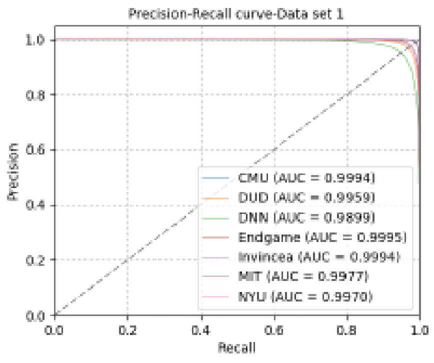


(b)

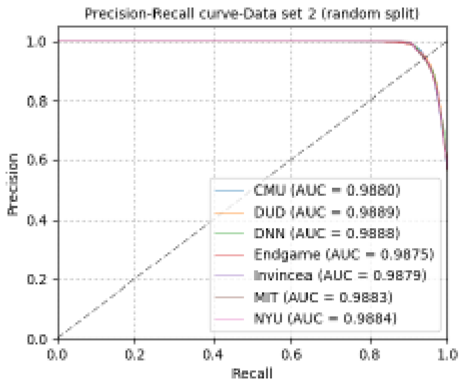


(c)

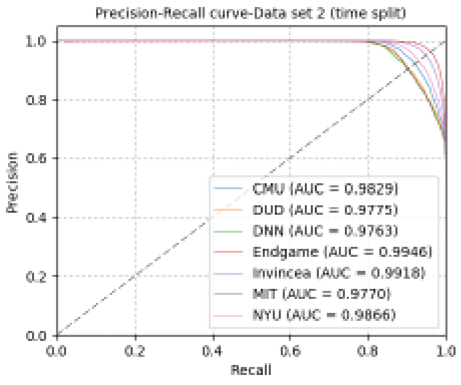
Figure 4: ROC curve for (a) Data set 1, (b) Data set 2 (random-split) (c) Data set 2 (time-split)



(a)



(b)



(c)

Figure 5: ROC curve for (a) Data set 1, (b) Data set 2 (random-split) (c) Data set 2 (time-split)

5. CONCLUSION

In order to identify bogus URLs, this research compares and contrasts deep learning-based character-level embedding algorithms. Various deep learning architectures have proven time and time again how effective they are. Two of the models are RNN-based, two are CNN-based, and one combines features from the two architectures to create a hybrid model. With an impressive false positive rate of only 0.001%, all of the models performed admirably in identifying harmful URLs 93% to 98% of the time. Deep neural networks (DNNs) and n-grams are compared. In every metric that has been used for evaluation, character models based on deep learning have consistently outperformed other models. By embedding characters at the level of the URL, deep learning algorithms may be able to handle a variety of malicious alterations effectively. In spite of deep learning's impressive advancements, character-level embedding models trained using these techniques may not necessarily outperform more conventional approaches. Standard approaches include regular expression blacklists, identification based on signatures, and elementary machine learning techniques. The DeepURLDetect (DUD) model can be improved by include other components such as server services, website content, registry keys, file paths, and network reputation. In my opinion, this road is crucial to the realization of any and all future objectives.

REFERENCES

1. Tran, K. N., Alazab, M., & Broadhurst, R. (2013, November). Towards a feature rich model for predicting spam emails containing malicious attachments and urls. In 11th Australasian Data Mining Conference, Canberra.
   Alazab, M., & Broadhurst, R. (2015). Spam and criminal activity.
2. Alazab, M., Layton, R., Broadhurst, R., & Bouhours, B. (2013, November). Malicious spam emails developments and authorship attribution. In Cybercrime and TrustworthyComputing Workshop (CTC), 2013 Fourth (pp. 58-68). IEEE.
3. Broadhurst, R., Grabosky, P., Alazab, M., Bouhours, B., & Chon, S. (2014). An analysisof the nature of groups engaged in cyber crime.
4. Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2011, December). Zero-day malware detection based on supervised learning algorithms of API call signatures. In Proceedings of the Ninth Australasian Data Mining Conference-Volume 121 (pp. 171- 182). Australian Computer Society, Inc..
5. Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q. V., Padannayil, S. K., & Simran, K. (2020). A Visualized Botnet Detection System based Deep Learning for the Internet of Things Networks of Smart Cities. IEEE Transactions on Industry Applications.
6. Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection sys-tem. IEEE Access, 7, 41525-41550.
7. Vinayakumar, R., Alazab, M., Jolfaei, A., Soman, K. P., & Poornachandran, P. (2019,May). Ransomware triage using deep learning: twitter as a case study. In 2019 Cyber- security and Cyberforensics Conference (CCC) (pp. 67-73). IEEE.
8. Srinivasan, S., Ravi, V., Sowmya, V., Krichen, M., Noureddine, D. B., Anivilla,

S., & Kp, S. (2020, March). Deep convolutional neural network based image spam classifi- cation. In 2020 6th Conference on Data Science and Machine Learning Applications (CDMA) (pp. 112-117). IEEE.

9. Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179.
10. Felegyhazi, M., Kreibich, C., & Paxson, V. (2010). On the Potential of Proactive Domain Blacklisting. LEET, 10, 6-6.
11. Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009, June). Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1245-1254). ACM.
12. Kolari, P., Finin, T., & Joshi, A. (2006, March). SVMs for the Blogosphere: Blog Iden- tification and Splog Detection. In AAAI spring symposium: Computational approaches to analyzing weblogs (pp. 92-99).
13. Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009, June). Identifying suspicious URLs: an application of large-scale online learning. In Proceedings of the 26th annual international conference on machine learning (pp. 681-688). ACM.
14. Chiba, D., Tobe, K., Mori, T., & Goto, S. (2012, July). Detecting malicious web- sites by learning IP address features. In Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on (pp. 29-39). IEEE.
15. Chiba, D., Tobe, K., Mori, T., & Goto, S. (2012, July). Detecting malicious web- sites by learning IP address features. In Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on (pp. 29-39). IEEE.
16. McGrath, D. K., & Gupta, M. (2008). Behind Phishing: An Examination of Phisher Modi Operandi. LEET, 8, 4.
17. Cao, J., Li, Q., Ji, Y., He, Y., & Guo, D. (2016). Detection of forwarding-based malicious urls in online social networks. International Journal of Parallel Programming, 44(1), 163-180.
18. Choi, H., Zhu, B. B., & Lee, H. (2011). Detecting Malicious Web Links and Identifying Their Attack Types. WebApps, 11, 11-11.