Universiteit Leiden

Master Computer Science

SNet : An Attention Mechanism Based Deep Neural Network For URL Classification

Name:	Saket Narendra
Student ID:	2759330
Date:	27/06/2022
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. E.M. Bakker
2nd supervisor:	Prof. Dr. M.S.K. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

" Parasparopagraho Jīvānām "

" Souls render service to one another "

Mahavira, 24th Tirthankara of Jainism

First and foremost, I would like to thank the Gods for their blessings and strength in every step of life.

I want to express my deep and sincere gratitude to my manager and mentors from the Advanced Technology Team at Irdeto, Tasnim Jemli, Shane O Meachair, and Jessica Alecci, for imparting invaluable knowledge, guidance and allowing me to conduct research. It was a great privilege and honour to work alongside you. I would also like to extend my gratitude to my thesis supervisors, Dr E.M. Bakker and Prof. Dr M.S.K Lew, for their critical remarks throughout the thesis.

Last but not least, I would like to thank my father, Air Vice Marshal B Narendra Kumar, my mother Smitha B.V, my brother Lieutenant Commander Ankit Narendra, my sister-in-law Sarvani Muppane, our beloved dog Casper, friends and well-wishers for always supporting me.

It would not have been possible without your support and guidance.

Thank You.

Abstract

Malicious URLs that lead to malware, phishing attacks or which contain pirated digital content are among the most common ways to deceive individuals into frauds, misdirect to acquire valuable data, and illegally obtain movies, games, and other recreational content. This work discusses differentiating between malicious and benign websites and can be extended to any URL binary classification task. As baselines, several experiments on large-scale datasets were conducted with two tree-based ensemble machine learning models, namely the Random Forest and XGBoost classifier and the deep learning model, URLNet. We also illustrate the importance of features present in the URL, which affect the models' predictions. Finally, we propose two novel methods involving an attention mechanism, SNet and SNet v2, which have shown certain improvements over the baseline methods.

Key Words: Malicious URLs, Machine Learning, Attention Mechanism, Cybersecurity

Contents

1 Introduction

Cybercriminals have become increasingly active in the past decade due to increased global internet usage. These actors develop malicious websites that can pilfer valuable data and use them in unethical or illegal ways. The malicious websites often masquerade as legitimate links, which deceive people into clicking on them, consequently giving access to the cybercriminals to their personal information. Websites that lead to phishing scams are among the most common methods to steal money and information from people and often occur in emails and URLs.

URL stands for Uniform Resource Locator[1], and it is the address of a particular source of information on the web. A URL comprises several parts, some of which are necessary to access the web page, such as the scheme, domain name and path to the file, whereas other parts capture features such as parameters and anchors. The structure of a URL is illustrated in Figure.1. An example of a malicious URL being 'http://www.scandals.co.nz/' and a benign URL being 'http://members.tripod.com/russiastation/'.

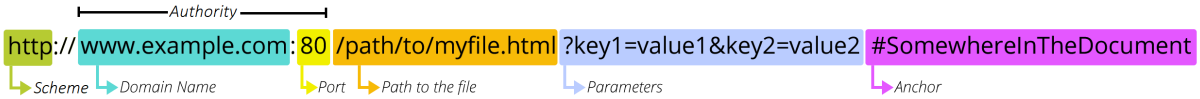

Figure 1: An example of a URL divided into multiple parts[1]

Cybercriminals may also use URL shortening tools to shorten the link and embed malware, include suspicious words and more special characters to masquerade malicious URLs as genuine or benign URLs that often lead to scams. The same goes for websites that host entertainment content. The resources are illegally obtained and posted on free-to-use websites where anyone can download the content without paying. The developers of these pirated websites make multiple mirror links to the same website, so even if one of the websites has been removed, the others will continue to provide the streaming content, which might also contain malware. Many resources are utilized for creating the content and should be safeguarded from illegal redistribution[2].

For a long time, the task of URL classification, whether containing malware, explicit content, or ones that lead to phishing scams, has been done manually. These approaches, however, are very labour intensive and can consume much time in a fast-changing digital world. Many URL classifiers[3][4] use classical machine learning methods such as the Naive-Bayes method, SVMs and Random Forests. These classical methods might be fast in classifying different web pages; however, they fail to capture the semantic meaning of the URL. These semantics, such as the number of special characters and the order of characters/words appearing together, may play an essential role in influencing the predictions made by the models. They require extensive feature engineering where the models might fail to perform well on unseen data.

The classical research methods aim to capture the URL's features, such as HTML content, geographical location and special characters present, with the help of TF-IDF methods. Deep neural networks for the URL classification problem that use CNNs, RNNs and LSTMs are some of the recent applications in Natural Language Processing[5][6][7][8][9].

The benchmark dataset for URL classification used in the works of URLNet[5] (state-of-the-art URL classifier) contains about 20M URLs, with roughly 94% of the dataset being 'benign' and 6% 'malicious'. It was collected from VirusTotal which is a pay-to-use database of malware and known malicious links. The chosen dataset for this work from MendeleyData[10] includes about 1.5M 'benign' URLs and 35k 'malicious' URLs (see Table.2), making this dataset a heavily imbalanced one yet, representative of the benchmark dataset. The data has been collected by crawling the internet using MalCrawler[11] and the labels have been verified using the Google Safe Browsing API[12]. In real-life scenarios, if one were to collect such URLs for the task of URL classification, quite often they might find this imbalance between the two labels. We also experimented with the balanced version of the Mendeley dataset by balancing the malicious and benign samples.

This work demonstrates the experiments conducted with two tree-based ensemble machine learning models, Random Forest and XGBoost and one deep neural network model, URLNet. These supervised machine learning models were used as a baseline (see Section 4) to demonstrate the performance on the selected dataset. The tree-based ensemble models though are fast to classify, have shown that they are incapable of effectively classifying these URLs, thus not being suitable for URL classification. On the other hand, deep neural network models have shown to perform well for the task of URL classification. The best F1-scores and confusion matrices for all the models with the highest true positive rate and true negative rate have been illustrated. Statistical analysis has also been conducted on the data which shows quantitative differences in the features present in the URL. The quantitative features such as the number of special characters (e.g. '/','@','.') present in the URL may affect the data quality, and hence we show that the predictions made by the machine learning models may be affected by these features. This paper also introduces two novel implementations of a malicious URL classification system called 'SNet' and 'SNet v2' that use an Attention mechanism[13]. The SNet is an extension over the current state-of-the-art URLNet, by Hung Le et al.[5]. SNet v2, a variant of SNet was implemwnted based on statistical analysis, which uses the attention mechanism directly onto the embeddings to give more weight to the characters and words occurring in a URL. Extensive testing shows that our model SNet and its variant, the SNet v2, perform competitively with the state-of-the-art baseline in URL classification by obtaining higher balanced accuracy and average precision.

The rest of the paper is organized as follows: Section 2 describes the different methods previously used by researchers in the domain of Natural Language Processing for URL classification. In Section 3, we describe the various metrics chosen to measure the performance of our work. Section 4 describes the methods chosen as baselines for the comparison to our proposed model (which is described in Section 5). In Section 6, we describe the URL dataset and the exploratory data analysis conducted. Section 7 describes the experiments and results for the various models chosen. Finally, we discuss the future scope and conclude in Section 8.

2 Related Work

Existing URL classification methods include classical methods such as the Naive Bayes and SVM. Recently, reinforcement learning and deep neural networks have been successfully applied to the problem. This section gives an overview of the most critical works on the subject.

Classical Machine Learning Methods

To demonstrate the importance of the attributes of a URL, Singh et al.[14] performed a holistic analysis by combining many attributes to classify malicious webpages. They combined HTML content, geographical location, and JavaScript-based attributes. Singh et al. worked on the benchmark dataset from Mendeley Data[10], which is also used in our work. Similar to the works of Singh et al., Weedon et al.[4] also showed the importance of lexical features (e.g. URL length, domain length, number of special characters) present in the URL by performing experiments on a small-scale dataset with the Random Forest classifier. Thirumoorthy et al.[3] worked on two classifiers, namely, Naive Bayes and SVM, for text classification. In their work, the feature selection is based on the TF-IDF method. By selecting features based on term frequency, they illustrated that such a feature selection method can improve the classification accuracy. Though TF-IDF methods can work well on smaller documents, they have some limitations, such as not considering the semantic similarities between the words and characters. Alejandro et al.[15] demonstrated the usage of Random Forests showing that feature engineering can be very beneficial for a more efficient analysis of phishing URLs. Data representation plays a vital role in any natural language processing task as high dimensional data may cause the model to perform poorly.

Reinforcement Learning (RL)

The works of Tianyang et al.[16] demonstrated a task-friendly representation that identifies important words or task-relevant structures without explicit structure annotations, which thus yields competitive performance towards learning a structured representation for text classification. Yin et al.[17] proposed two integration schemes named combination and hybridization, which are presented to attain synergy between different RL techniques. Shared long-term memory is used to accumulate the relevant knowledge exploited from multiple user experiences.

Deep Neural Network Models (DNN)

Convolutional Neural Networks have played a significant role in problems such as text classification in recent years. Le Cunn et al.[18] proposed Very Deep Convolutional Networks for Text Classification, where they showed that by increasing the depth of the CNN, the accuracy increased. On the other hand, Le et al.[6] demonstrated that a shallow-and-wide convolutional neural network at the word level is highly effective for text classification. However, increasing the depth of such convolutional models with character/word inputs might not bring significant improvements. It is necessary to consider the various conditions for CNNs to work well, such as feature parameter sharing and dimension reduction. In our proposed model SNet[5], we employ these conditions by reusing the weights and having the appropriate filter sizes. RNN-based deep learning architectures have also been widely used for malicious webpage classification, where Ebubekir et al.[7] worked towards finding meta tag information contained in the web page that can be used to classify a web page as malicious or benign. They illustrated that the meta tag information of a website, such as the words present on the website and the URL, can be used to classify webpages. This inspired us to include some meta tag information such as the top-level domain, 'https' tag and more for URL classification.

Multiple methods exist for the problem of URL classification, such as LSTMs, RNNs and

transformers. PhishingNet[8], proposed by Yongjie et al., makes use of CNN and RNN fused via a three-layered CNN to build accurate feature representations of URLs, on which the phishing URL classifier is trained to identify phishing websites. In PhishingNet, the attention is applied sequentially, first to the characters and then to words, hence limiting the capabilities of the attention mechanism in a way in which importance is not given to characters, words and character-level words individually. SNet (see 5) overcomes the limitations of PhishingNet by applying attention to the character-level branch and word-level branch, which is further divided into the character-level representation of words.

Shi et al.[9] proposed an Extensive Pyramid Network that makes use of the self-attention mechanism[19] to capture the semantic relationship between the words in the sequence. They also use recurrent layers to obtain the order information and convolutional layers to get the local contextual information. RNNs/LSTMs face the issue of a long-range-dependency problem which can negatively affect the model's performance by not allowing the model to remember important information. Since these structures encode the given text into a vector of fixed size, it is incapable of remembering long text sequences. We did not choose these models for our work since they face the problem of long-range dependency and take up a lot of training time. Hence, the attention mechanism is a solution to overcome this problem by giving importance to such essential features and being able to remember longer sequences. The attention mechanism has been quite successful for various NLP tasks. Its ability to selectively focus on essential features while ignoring the rest has made the attention mechanism the go-to method for our work.

Many classical machine learning models might produce good results; however, such methods may fail to capture the correlation of the characters and words, essential in classifying the URLs, resulting in poor URL classification performance. It is necessary to overcome some of the problems and limitations mentioned by combining several methods, including URL attributes as features, meta tag information, and performing feature engineering by using term frequency for supervised machine learning models. Inspired by the works of Yongjie et al.[8] and Shi et al.[9], we apply an attention mechanism to URLNet to have a well-performing classification system called the SNet and SNet v2.

3 Fundamentals

In this section, we discuss and define several well-known metrics used to measure the performance of the baselines and the proposed model. These metrics help to understand how well the models perform (These metrics are calculated using the python library scikit-learn[20]).

Confusion Matrix

A confusion matrix is used to evaluate the performance of a classification model. It aims at comparing the actual target value to the prediction made by the machine learning model. With the help of the confusion matrix, one can determine whether the model is getting confused in classifying the two classes.

A confusion matrix A is such that $A_{i,j}$ are the number of observations that are in class i

and predicted to be in class *j*. In our case we have only 2 classes (malicious and benign); we consider $A_{0,0}$ to be true positives, $A_{0,1}$ as false negatives, $A_{1,0}$ as false positives and $A_{1,1}$ as true negatives. An example of a confusion matrix is given below in Table.1

	Predicted Values		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	True Positive (TP)	False Negative (FN)
	Negative (-1)	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix

The threshold used in our work to calculate the confusion matrix is 0.5. When we call the predict method of an estimator instance from scikit-learn[20], 0.5 is used as the default threshold. This threshold ensures that the given probability of having 1 is greater than the probability of having 0.

Accuracy Score

Accuracy is defined as the ratio of the sum of the labels that have been correctly identified as positive and negative by the model to the sum of all the actual and predicted labels. It is given by Equation 1,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Accuracy score may not be an appropriate metric to evaluate the performance of the model on an imbalanced dataset as the model may easily predict the majority class (benign URLs) but not the minority class (malicious URLs).

Precision Score

The precision score is the ratio of true positives to that of all the positive labels. The intuition behind precision is to not classify a label that is negative as positive and vice versa. High precision indicates low number of false positives and vice-versa. It is given by Equation 2,

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall Score

The recall score is the ratio of true positives to that of the sum of true positives and false negatives. The intuition behind this metric is the model to be able to find all the positive labels. High recall indicates low number of false negatives and vice-versa. It is given by Equation 3

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1 Score

The F1 score which is also known as F-measure is the harmonic mean between precision and recall. The F1 score is a good metric of performance when the dataset is imbalanced. The relative contribution of precision and recall are equal in the F1 score. It can be calculated using Equation 4,

$$F1 = 2 \times \frac{Precision \times recall}{Precision + Recall} \tag{4}$$

Precision-Recall Curve

Class imbalance is a common problem in the binary prediction domain. This happens when the number of negative samples (benign URLs) outnumber the positive samples (malicious URLs). The Precision-Recall curve is specifically informative in imbalanced datasets because the precision and recall do not consider the true negatives and remain unaffected by the imbalance in the data.

According to the definition of precision[2] and recall[3], the higher the precision, the stronger the model is in classifying an observation as positive. If recall is higher, the more positive observations have been classified correctly as positive. The precision-recall curve shows the trade-off between the two which gives us an optimal value for multiple thresholds.

Average Precision

The Average Precision (AP) is a method of condensing the precision-recall curve into a single number that represents the average of all precisions. It is the weighted sum of precisions at each threshold where the weight is increasing in recall. It is given by Equation 5,

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{5}$$

where, R_n and P_n are the precision and recall at the n^{th} threshold respectively. The number of thresholds is *at most* equal to the number of samples as several samples may have the same underlying continuous value from the classifier. For imbalanced datasets, the average precision can be used as one of the primary evaluation metrics.

Sensitivity

Sensitivity is the true positive rate of a model or in other words the ability to correctly identify positive classes. It can be calculated with the help of a confusion matrix and is given by Equation 6,

$$Sensitivity = \frac{TP}{TP + FN} \tag{6}$$

Specificity

Specificity is the true negative rate of a model or in other words the ability to correctly identify

negative classes. It is given by Equation 7,

$$Specificity = \frac{TN}{TN + FP} \tag{7}$$

Balanced Accuracy Score

In cases of imbalanced datasets, the balanced accuracy score calculates the average of recalls obtained for each class in binary classification. This can be calculated by Equation 8,

$$Balanced\,Accuracy = \frac{Sensitivity + Specificity}{2} \tag{8}$$

Balanced Accuracy accounts for both the positive and negative outcome classes and does not mislead with imbalanced data. For this reason, balanced accuracy score can be considered as one of the primary metrics to evaluate the performance of a model.

Youden's J Statistic

The Youden's *J* statistic[21] is a threshold value which maximizes the precision and recall. It is given by Equation 9,

$$J = Precision + Recall - 1 \tag{9}$$

The index of maximum value of precision and recall corresponds to the index of the best value of threshold. In this work, we use the Youden's J score to indicate the best threshold value on the Precision-Recall curve.

In cases of balanced datasets, the classification report includes Accuracy, Precision, Recall and F1 score which determine the performance of the data. Though for imbalanced datasets, it is very common for the model to predict the majority class (in our case the benign URLs). Hence, it may lead to erroneous conclusions as the accuracy score might be too high. For such cases, the average precision and balanced accuracy score can be used. Since we want to find the positives in our classifier, Balanced Accuracy performs well. Both classes are given equal weight, hence the score is lower than what accuracy predicts. In order to maximize the precision and recall value and to obtain the optimal threshold, we can use Youden's J score.

4 Baseline Methods

This section describes the methods chosen as baseline for comparison to our proposed model. We have chosen two tree-based ensemble machine learning algorithms (implemented with scikit-learn[20]), and one based on deep neural networks, for URL classification, namely,

1. Random Forest
2. Xtreme Gradient Boosting (XGBoost) and
3. URLNet

The ensemble based methods were chosen as baselines because they are easy to use, understand and are extremely fast in classifying malicious and benign URLs. Though for NLP tasks, deep neural networks have proven to perform well. Hence, URLNet was chosen as one of the baselines as it is the state-of-the-art model for URL classification,.

4.1 Random Forest Classifier

A Random Forest[22] model is an ensemble learning strategy that utilizes the advantage of 'majority voting'. It is made up of several trees that have been trained using bagging, also known as bootstrap aggregation. *Bagging* randomly selects a sample of data from the entire set. Each model is created using row sampling to replace the samples (Bootstrap Samples) supplied by the original data. Bootstrapping refers to the stage of row sampling with replacement. Each model is trained individually, producing results. The Random Forest model produces an output based on the average of individual tree outcomes, called *aggregation*. As a result, increasing the number of trees in a forest can increase the Random Forest's performance. If the decision trees are unstable, then there is a good chance the models might overfit the data since the trees used by Random Forest are shallow 'weak-learners', and hence not likely to overfit individually. This occurs because even a minor modification in the tree's structure might result in severe or unsatisfactory effects. A Random Forest model can produce good results even with the default hyperparameters.

Though the Random Forest classifier works well, as mentioned above, it also has some drawbacks. Random Forest models are not very interpretable, and if the dataset is vast, it can create many trees that may consume more memory, thus slowing down the training process.

4.2 XGBoost Classifier

XGBoost, or "eXtreme Gradient Boosting"[23] is a machine learning ensemble algorithm. The XGBoost classifier is widely used due to its ability to handle missing data, being computationally as well as cache efficient. The model prunes the trees backwards; in other words, it applies a 'depth-first' approach, increasing the computational performance. It is a model that minimizes overfitting thanks to built-in regularization. Boosting outperforms bagging, and Gradient Boosting is a strong boosting ensemble technique. XGBoost is less sensitive to overfitting and learns from past mistakes because it is regularized. It has been shown to work well even without hyper-parameter adjustment. The models created by gradient boosting make predictions on the residual terms of the previous models, which are then combined. Gradient boosting uses the gradient descent algorithm that minimizes the loss when the newer models are added. The sequential addition of the models may make the model more robust and efficient.

The XGBoost model is efficient, accurate, feasible, and performs parallel computations making it a widely used classifier. It works well with large datasets and can be simple to use. For these reasons, we chose the XGBoost as one of our baselines.

4.3 Data Pre-Processing for Random Forest and XGBoost

URLs can be split into multiple components as shown in Figure.1. We split the URL into its different components such as the address, scheme, netloc, path and file with the help of a URL parser (python module urllib.parse), which helps split the URLs. We check whether the URLs contain IP addresses by checking their hexadecimal form and string matching. If they are present in the URL, it will return 1; else, 0. These are checked for each URL present in the dataset and assigned a representative value for those URLs. The dataset used for classification with supervised machine learning models is obtained from Mendeley Data (see 6). It contains URLs in their raw format and some extracted attributes such as the geographic location, the length of JavaScript code, and class labels for benign and malicious webpages.

Cybercriminals may perform several actions to generate malicious URLs and can be described as follows,

1. URL Shortening: Cybercriminals can use URL shortening tools that shorten the links and embed malicious content and malware into the URLs. These shortened URLs can redirect a person to the websites that the cybercriminal would want them to click (e.g. bit.ly, tinyurl.com/23fdd).

2. Suspicious Words: Most malicious URLs contain suspicious words that can often be easily recognised. Links that lead to scams include words such as 'urgent', 'lucky', 'free', 'pay', 'bank', and 'won' may be considered suspicious and hence tagged as malicious.

3. Directory Mimicking: The number of directories in the URL is usually delimited by a '/', which shows the number of directories in that particular URL. If a URL is long-lived or has been in existence for a while, it would most likely contain more directories, and perhaps more special characters present in them as well. Cybercriminals in the modern age may mimic these benign URLs by embedding malware and luring people by uploading explicit content in clickable links, leading to phishing in malicious URLs.

4. Special Characters: The cybercriminal may also include multiple special characters in the URL, which can mimic the characteristics of a benign URL. The number of special characters present in the URL was taken into consideration. These special characters include '@','#','%','/','.',':'. Along with the special characters, other characteristics include the length of the URLs, length of top-level domain and number of digits. All the characteristics are combined to give us features used by the machine learning models.

We include the features mentioned in the works of Singh et al.[14], Ebubekir et al.[7], Weedon et al.[4] and Le et al.[5] such as different attributes of a URL for classification. This has shown that qualitative features (e.g. presence of top-level domain, https, etc.) and quantitative features (e.g. length of URLs, number of special characters, etc.) present in the URL may help classify the URLs. These features are pre-processed by converting them into vectors using the scikit-learn's[20] TF-IDF vectorizer. The data is then label-encoded to encode the target variables as 0 or 1. Finally, a pipeline is implemented which applies column transform, with whose help the generated features can be concatenated to form a single feature space, and the final estimator can be sequentially applied. These features are then passed to our chosen

baselines of ensemble machine learning models to classify URLs as malicious or benign. The architecture showing the flow of information is depicted in Figure.2

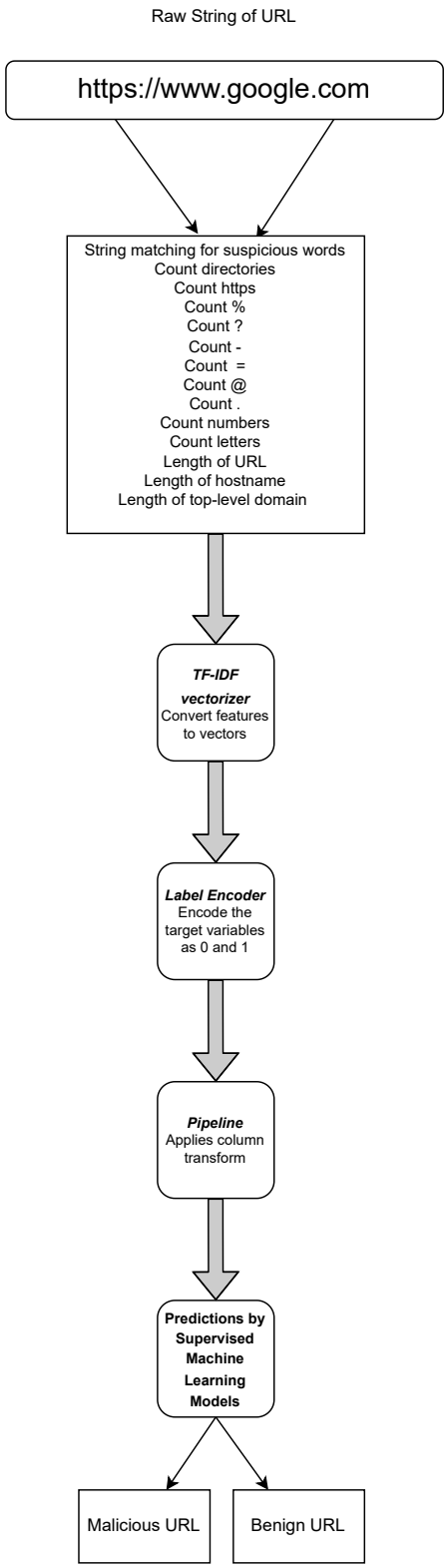

Figure 2: Architecture showing the preprocessing steps leading to the features used by the classifiers

4.4 URLNet

URLNet has been introduced by Hung Le et al.[5]. The URLNet architecture is one of the baselines used in this work as a state-of-the-art model in the problem of URL classification. Our proposed model, the SNet[5], is a modification on the URLNet architecture. In this work, URLNet is used on the Mendeley Data[10] which is a large scale dataset containing many URLs(see 6).

4.4.1 Architectural Setup

The input to the model is a raw string of URL. Consider there to be a set of *T* URLs, $(u_1, y_1), ..., (u_T, y_T)$, where $u_T = 1, ..., T$ represents the URL and $y_T \in \{-1, +1\}$ represents the label of the URLs with $+1$ being a 'malicious' URL and -1 denoting a 'benign' URL.

The first stage in the classification process is to create a feature representation $u_t \rightarrow x_t$, where $x_t \in R^n$ is the n-dimensional feature vector that represents the URL u_t. The next stage is to create a prediction function(CNN), $f : R^n \rightarrow R$, which is a rating that predicts a URL instance's class assignment denoted by *x*. The method's prediction is symbolized by $\hat{y_t} = sign(f(x_t))$. The goal is to discover a function which can reduce the number of errors across the entire dataset. This is accomplished by minimizing a loss function.

To use the lexical features, we obtain a feature vector *x* from URL *u*. The special characters present in a URL help to act as a delimiter for the URL. All characters and words are combined to give a dictionary, and hence all the unique characters and words become features. If there are M distinct features present in the URL, each URL is mapped to a vector $x_t \in R^M$. Many quantitative features such as the number of '/','@','.',':' have also been made use of as we discuss its importance in Section 6

The URLNet architecture is depicted in Figure.3 and consists of the following modules :

Char-level branch

In URLNet, Convolutional Neural Networks have been used at both character-level as well as word-level branches. Since a URL u consists of sequence of characters (delimited by special characters), we represent this by a matrix $u \rightarrow x \in R^{L \times k}$ where instance *x* is made of continuous components x_i, i=1,...,L in a sequence, where the component can be a character or word present in the URL. The individual component is a k-dimensional vector which is randomly initialized and learned by the model. Each such component is represented by an embedding $x_i \in R^k$. The sequences are padded to the same length L.

With the help of a convolutional operator, over the instance $x \in R^{L_1 \times k}$ where k = 32 and L_1 = 200, a CNN would convolve and therefore generate new features.

Every segment of the input is divided by a pre-defined stride value, from which the output of this convolution layer applies a filter *W* with a nonlinear activation.

There are 4 different convolutional filters $W \in R^{k \times h}$ where h=3,4,5,6. Hence, the model learns the sequential patterns in a sequence of 3, 4, 5, 6. We employ 256 filters for each filter size.

Subsequently, after the convolution layer, a max-pooling layer is applied that selects the maximum elements from the region of the feature map that is covered by the filter. These features are then passed to a fully connected layer regularized by dropout followed by four more fully connected layers which ultimately leads to the output classifier.

Word-level branch

The word-level branch has a similar structure and flow of data like the char-level branch where both make use of CNNs. The word-level CNN takes a series of input URLs from which the order of words appearing together are considered. The training corpus's unique words, which can comprise alphabets and numerals, are extracted. The $< PAD >$ token is used to make the length of the words consistent. The number of words occurring only once in the training corpus can be very high as the dataset increases. This can cause memory constraints in the model leading to slower training. To tackle this issue, all the words which occur only once are replaced with an $< UNK >$ token. This can help reduce memory usage and speed up the training process.

Since many special characters occur in the URL, it is vital to consider their importance. The special characters that occur in the URLs are considered unique words. To take these special characters and words into consideration, the word embeddings can be considered the sum of the word embeddings and the character embeddings of that word. There are two matrices involved here: an embedding matrix EM_w for words and another embedding matrix for characters EM_c. The character embedding matrix is different from the embedding matrix used for the character-level representation of the URL as the character embedding matrix for words aims to represent the words at a local level.

We acquire the representation $URL_w \in R^{L_2 \times k}$ based on EM_w while getting the URL Matrix representation. Then, using EM_c, we get a matrix representation $L3 \times k$ of each word in a URL, where each word is padded or shortened to become a sequence of $L3 = 20$ characters. This matrix added together gives the word a vector embedding of $1 * k$. Then we obtain the *Character-level Word embedding*, which has the URL matrix representation $URL_{cw} \in R^{L_2 \times k}$. The sum of these two matrices, $URL_w + URL_{cw}$, yields the final URL matrix representation.

Once the matrix representation is obtained for word-level branch (or char-level word representation), it is passed to the Convolutional Neural Networks which then performs the same steps as mentioned for the char-level branch. For combined features of the char-level as well as word-level branches, once the convolution is applied to both branches and is passed to the fully connected layer, the output for both branches are concatenated. The features from this concatenation are then passed on to four fully connected layers ultimately leading to the output classifier.

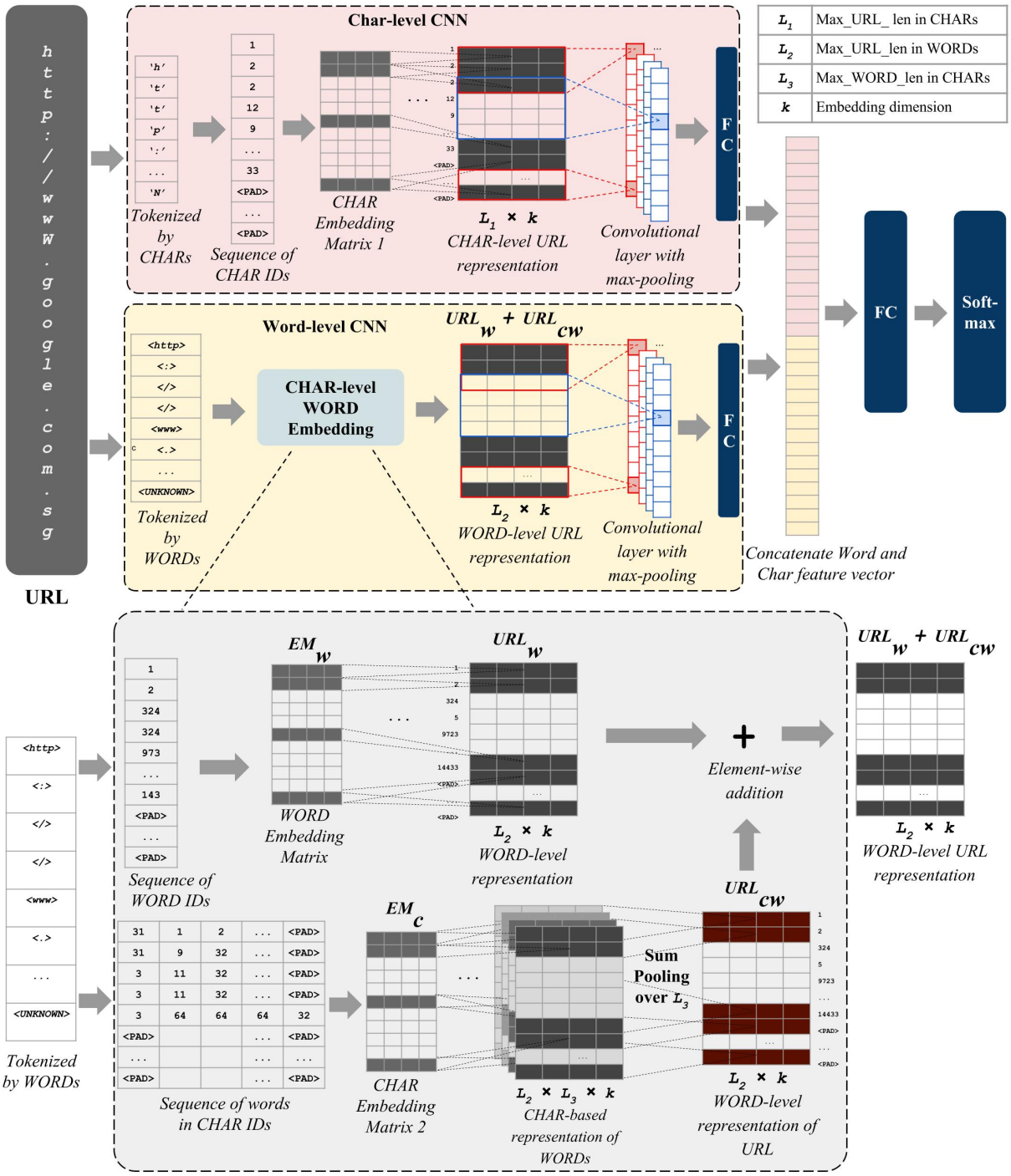

Figure 3: Architecture of URLNet[5]

5 SNet

The SNet architecture is a modification of the URLNet architecture[5]. This work describes how the attention layer applied to the features improves the performance of the URLNet as it is an endeavour to duplicate the action of selectively focusing on few relevant features such as special characters and words appearing together. We added an attention mechanism to the URLNet architecture in order to give more emphasis on the features that are passed by the CNN. By doing so, the model quickly learns which features (e.g. number of '/', '.', etc.) to give importance to while neglecting others. We make use of the attention mechanism inspired by Vaswani et al.[19], Bahdanau et al.[13] and Yongjie et al.[8]. The entire network is depicted in Figure.4

5.1 Attention Mechanism

Neural machine translations were previously based on encoder-decoder RNNs and LSTMs until Bahdanau et al.[13] introduced the Attention mechanism in 2014. The problem with the encoder-decoder structure is encoding the input sequences to a fixed vector. This can be a bigger problem in cases where the URL string is very long. This problem is called the 'long-range dependency problem' of the RNNs and LSTMs (suffering from vanishing gradients). When Cho et al.[24] proposed the encoder-decoder structure in 2014, they demonstrated that as the text length increases, the model's performance decreases rapidly. In natural language processing, the attention mechanism appeared as an improvement over the encoder decoder-based neural machine translation system (NLP). The attention mechanism has been widely used in applications such as computer vision, speech processing and other NLP tasks.

Since a URL includes a mixture of alphanumeric values rather than a simple string of letters, it seems important to emphasize the sequence of characters or words that appeared together in the URL. We undertook various tests with encoder-decoder structures only to find that they still have the long-range dependency problem and need a lot of training time, making them unsuitable for the SNet architecture. If we dig deeper into CNNs, we may find certain limits. For example, consider a 4x4 convolution, in which the convolution filter comprises 16 char/words, and the value of the destination char/word may be computed by referencing to the 15 char/words around it and itself. As a result, only local information may be utilized to determine the destination char/word, which might lead to bias because the global information is hidden. There are ways around this, such as building deeper networks or adding more filters, which adds to the computational load. Hence, to address the issues faced by URLNet, we applied an attention mechanism to the CNN, which creates a global reference for each char/word-level prediction that may provide the char/words with additional significance. The attention mechanism seemed to be a suitable choice for this particular architecture as it might capture the semantic meaning behind the URLs and understand better which features are essential.

The SNet architecture, a modification to the URLNet, has the same flow of information throughout its structure. The difference is with the addition of the attention mechanism after the CNNs. To include an attention mechanism in the URLNet, the output of the convolutional layers from either Char-level CNN or Word-level CNN are passed to their respective attention

mechanisms by reusing the same weights. It computes the attention along the URLs. We take the dot product of the weights and inputs along with the bias terms. The context vector c_i for output y_i can be generated using the weighted sum of annotations which maintains the relative importance of the inputs. After which, we apply the *tanh* activation function followed by a *softmax* layer. With the help of the *softmax* layer, we can obtain the alignment scores. We use the attention mechanism to maximize the number of computations in parallel, which minimizes the total complexity. The attention mechanism allows the path between the input and output sequences to be as minimum as possible, which helps solve the long-range dependency problem by giving importance to those selected features. This kind of attention acts as a self-attention within the architecture where the mechanism allows inputs to interact with each other. The features obtained from the attention mechanism are then added to the max-pooling layer, followed by a fully connected layer regularized by dropout. The results are concatenated and followed by four fully connected layers, leading to the output classifier.

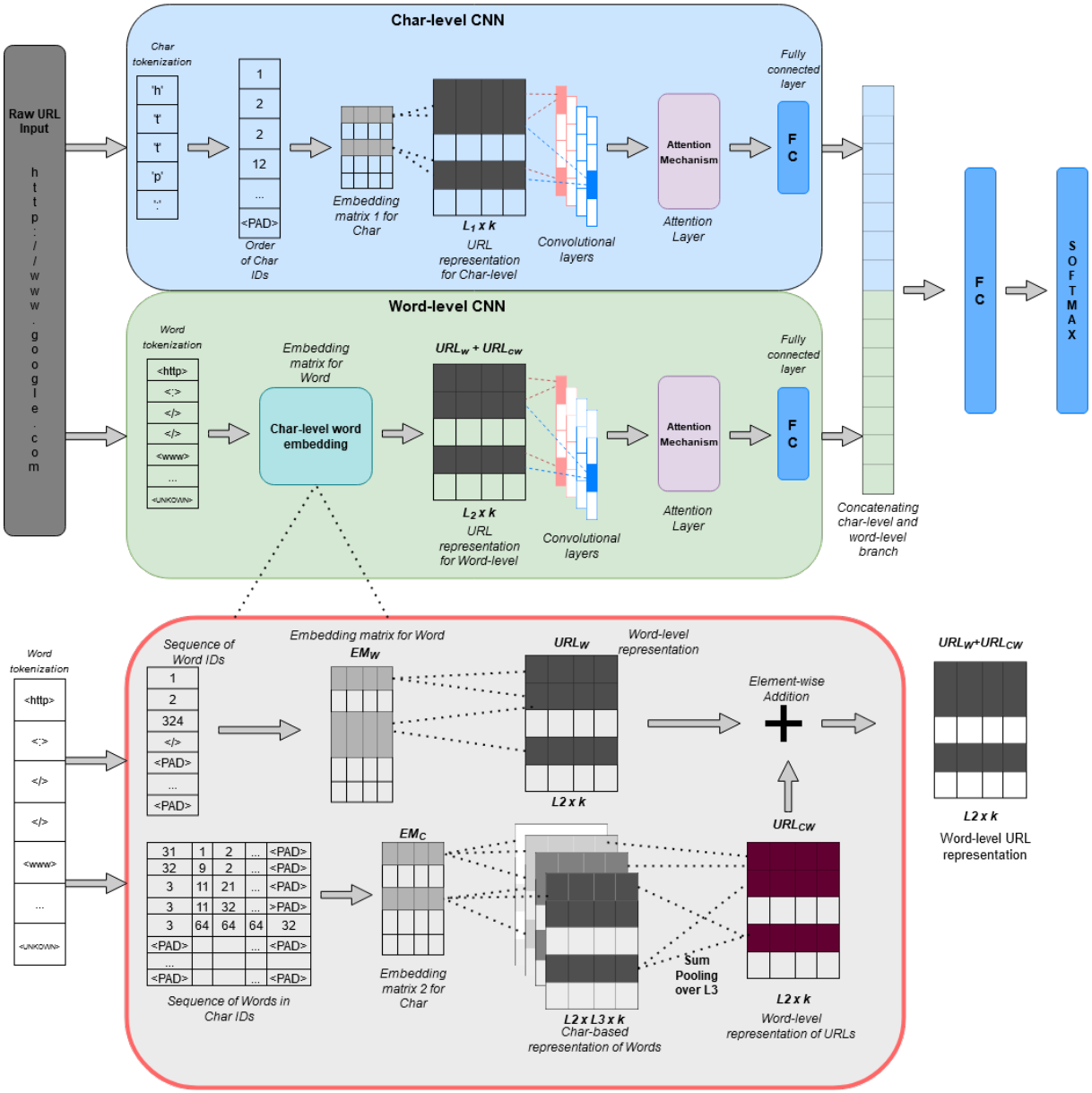

Figure 4: Architecture of SNet involving usage of Attention Mechanism

5.2 SNet v2

The SNet v2 was implemented in order to discover further improvements to SNet. We assume that the lexical features or the special characters present in the URL might play a significant role in the binary classification problem, specifically in cases where the dataset is highly imbalanced.

The variation of SNet implements the same architecture, with the difference being the placement of the attention mechanism. Since the embeddings capture the semantics of the URL by placing semantically similar characters and words closer to each other in the embedding space, it seemed necessary for attention to be applied directly to the embeddings. The output of the attention layer is forwarded to the convolutional layers, after which the structure of the SNet v2 remains similar to the SNet architecture. This helps the model understand better to give more importance to the quantitative features such as the number of '@',':','/' present in the URL, thus capturing the semantic meaning of the special characters. This variant of the SNet is illustrated in Figure.5

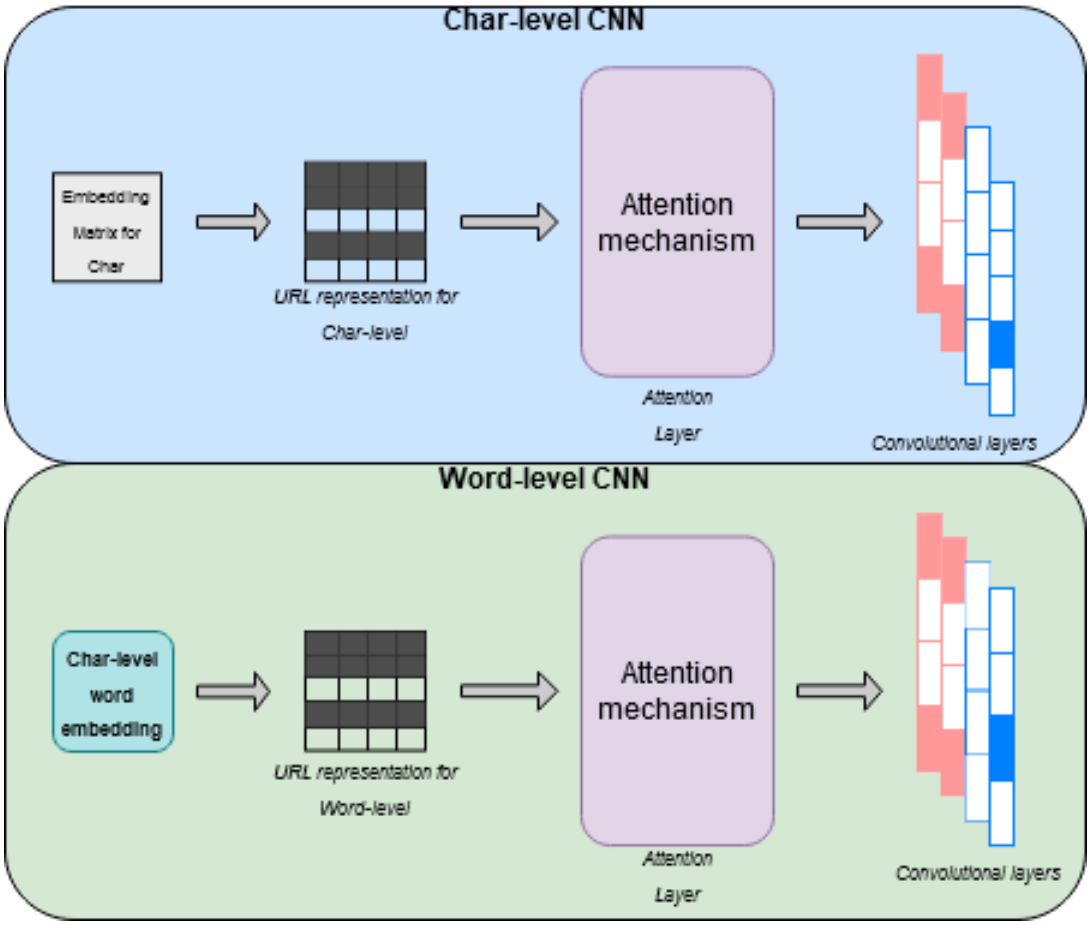

Figure 5: Architecture of SNet v2 involving usage of Attention Mechanism on the embeddings

6 URL Dataset and Analysis

In this section, we describe the Mendeley dataset[10] used for our experiments and conduct an exploratory data analysis to gain more insights into the statistical characteristics of the data.

6.1 Dataset

The Mendeley dataset has been used in many URL classification studies, and one such example is the work of Singh et al.[14] where they describe the importance of URL attributes such as HTML content, geographical location and JavaScript-based attributes. We use the Mendeley dataset as a benchmark dataset for our work. The dataset is imbalanced, consisting of URLs, and it is used for binary classification of URLs labelled as malicious or benign. The labels of the dataset are converted as '-1' for benign URLs and '+1' for malicious URLs throughout this work. Imbalance in the dataset may cause the models to be biased towards the majority class and fail to capture the minority class. Due to this, the model might return high accuracy by predicting the majority class. Hence, accuracy is not an appropriate metric to evaluate imbalanced datasets. In such cases, average precision and balanced accuracy can be used to evaluate the model's performance. Experiments were also conducted on the balanced version of the Mendeley dataset to investigate the models' performance on a balanced set. To create the balanced training set, random over-sampling was performed with the help of imblearn[25] on the default Mendeley training set. The samples are randomly duplicated from the minority class (malicious URLs). By doing so, more training samples are included in the dataset, which becomes more balanced. We performed data analysis to understand the characteristics of the data in depth.

Distribution of URLs

Distribution of the imbalanced dataset(Mendeley data) and the balanced dataset is illustrated in Table. 2 and Table. 3.

Mendeley Dataset	**Malicious URLs**	**Benign URLs**	**Total**
Training Set	27253	1172747	1200000
Testing Set	8062	353872	361934
Total	35315	1526619	1561934

Table 2: Distribution of Imbalanced(Mendeley) Dataset

Balanced Dataset	**Malicious URLs**	**Benign URLs**	**Total**
Training Set	1172747	1172747	2345494
Testing Set	8062	353872	361934
Total	1180809	1526619	2707428

Table 3: Distribution of Balanced Dataset

The distribution of the imbalanced dataset is visualized in Figure.6. We perform analysis on the

imbalanced dataset because, in many URL classification problems, the dataset is imbalanced, with the benign URLs being the majority class and malicious URLs being in the minority. This is often the trait since the number of malicious URLs found on the internet may be much less (but constantly growing) than benign URLs. From the distribution visualization(Figure.6), it is observed that in the Mendeley Dataset, the number of benign URLs is much higher than that of malicious URLs, making the task of identifying malicious URLs quite challenging.

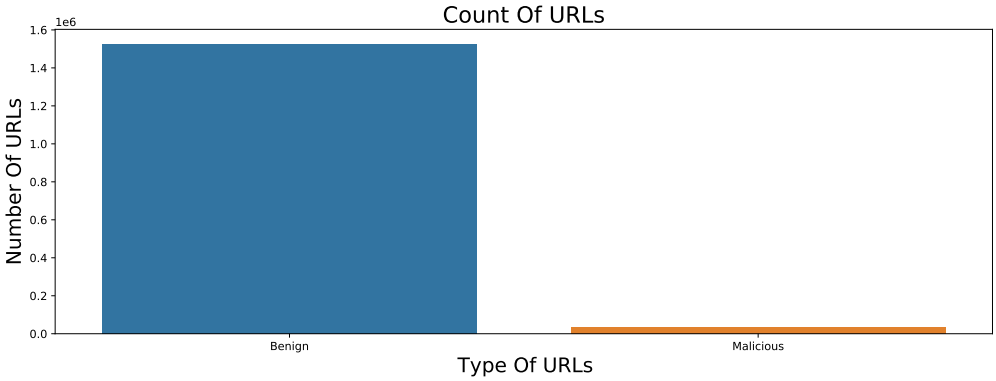

Figure 6: Distribution of Benign and Malicious URLs in the dataset

Quantitative Features : URL Length

The quantitative features present in the URL, such as the length of URLs, may play an essential role in the classification of the URL. From Figure.7, we observe that most of the URLs have less than 500 characters. From Figure.8 and Figure.9, we observe that malicious and benign URLs have an almost similar length of approximately 20-100 characters each, making malicious URLs harder to classify as they display similar characteristics to benign URLs.

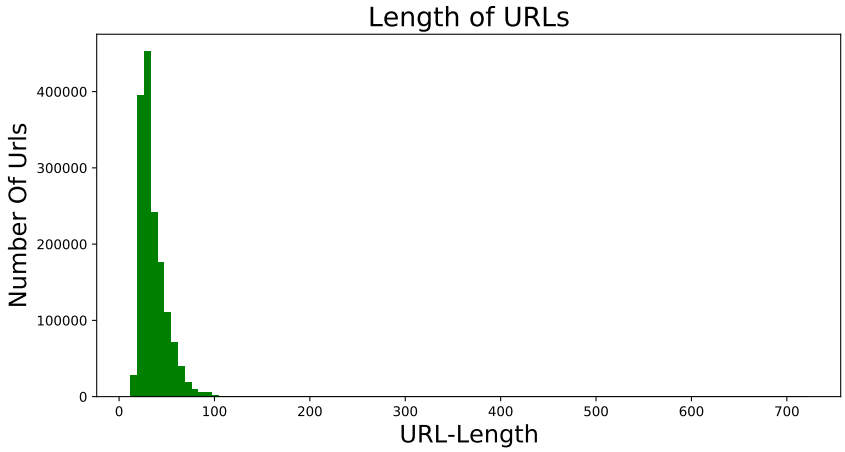

Figure 7: Length of all URLs in the Mendeley Dataset

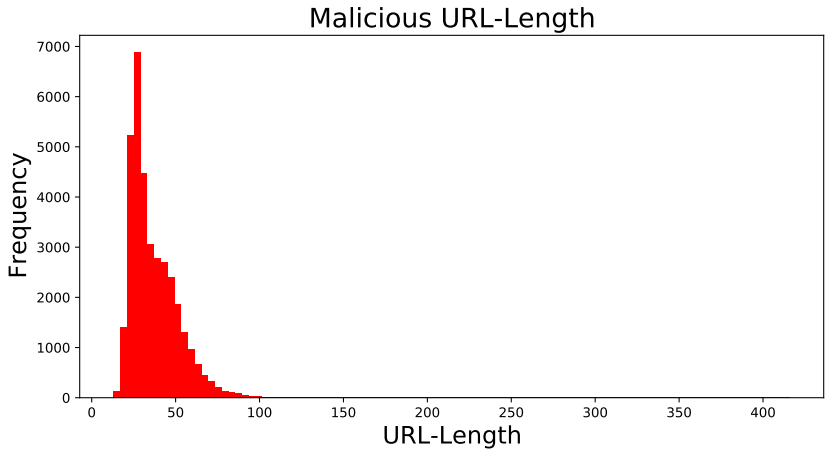

Figure 8: Length of Malicious URLs in the Mendeley Dataset

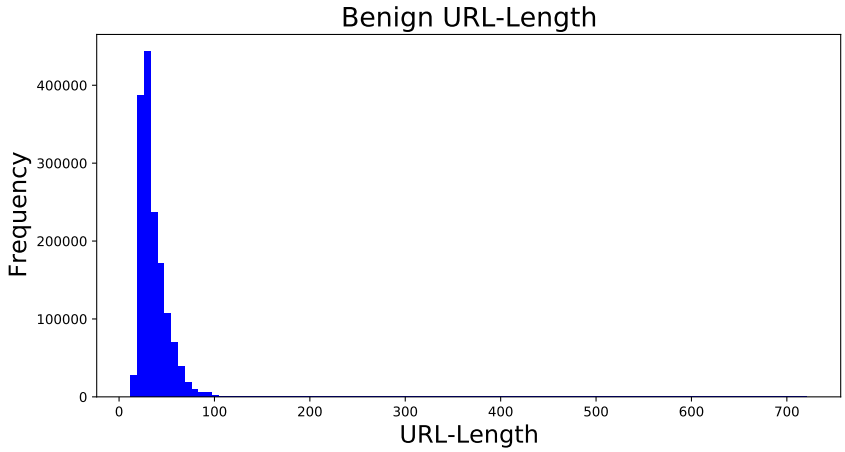

Figure 9: Length of Benign URLs in the Mendeley Dataset

Quantitative Features : Count 'www'

The 'www' domain acts as a hostname and can contain multiple subdomains. Many top-level domains (TLDs) have been added in recent years, and if a URL contains a TLD, which is rare, the usage of 'www' can help identify a legitimate website[26]. Many malicious URLs may masquerade themselves to appear as benign websites by containing the 'www' tag. As seen from Figure.10, it can be observed that the 'www' tag is present in most benign and malicious URLs with a count of 0 or 1. This makes it hard to classify a malicious URL with this feature alone. However, count 'www' of 2-4 is present mostly in benign URLs. Therefore, a malicious URL containing a count 'www' of more than two could be classified as benign. The normalized graph for count 'www' is illustrated in Figure.11.

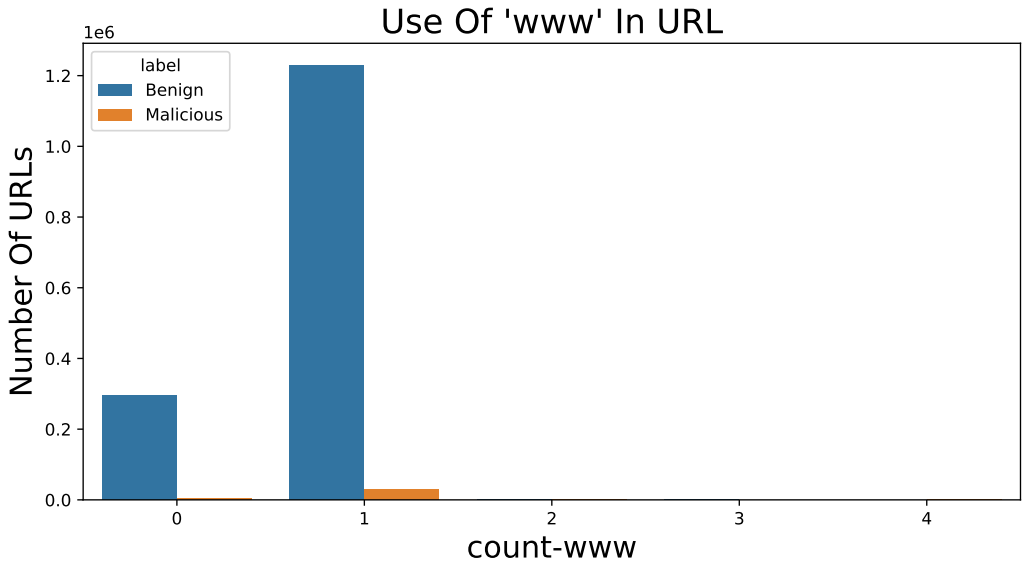

Figure 10: Count 'www' in the URLs

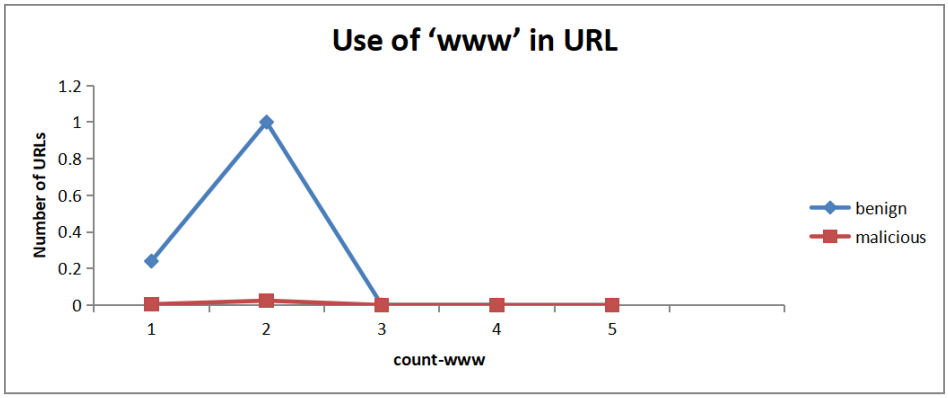

Figure 11: Normalized graph of Count 'www' in the URLs

Quantitative Features : Count 'Directories'

From Figure.12, it can be observed that the number of directories in malicious URLs is comparable to benign URLs. Malicious URLs are often short-lived (since they are created and taken down very often) and may not contain much information/content present in them (apart from malware). In the Mendeley dataset, the benign URLs have up to 15 directories, while malicious URLs have only up to 7 directories. If a malicious URL contains similar number of directories as a benign URL, it can be challenging to classify them. Genuine and constructed malicious URLs can look the same in that respect. These directories are delimited by '/', which indicates that the number of special characters present in the URL, such as '/','@','.', may impact the classification of imbalanced datasets. The normalized graph for the number of directories is illustrated in Figure.13.

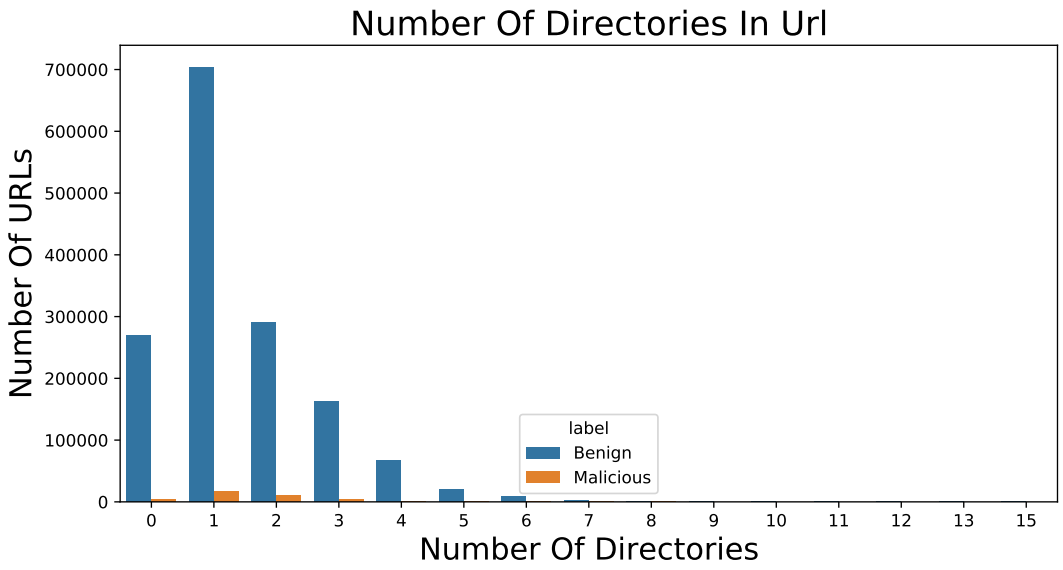

Figure 12: Number of directories present in the URL

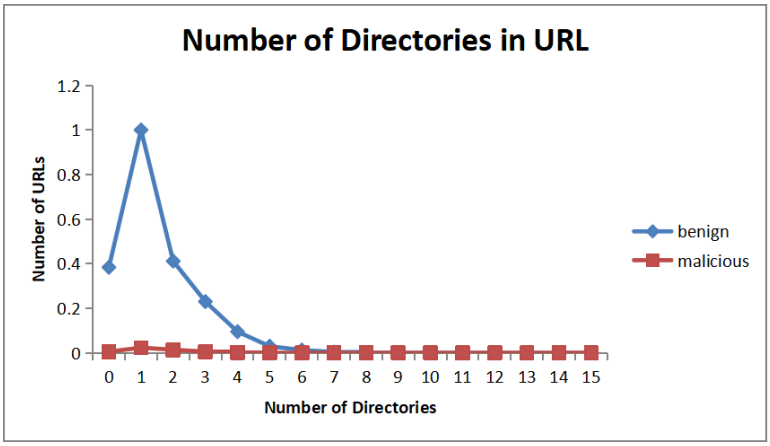

Figure 13: Normalized graph of number of directories present in the URL

6.2 Error Analysis

To illustrate the impact of lexical features present in the URL on the predictions (especially for an imbalanced dataset), we conducted an error analysis on the correctly and incorrectly predicted URLs. The predictions from URLNet's *Char-based CNN* on the Mendeley test set (imbalanced dataset) were used as it achieved the highest average precision. We considered several features such as count '-', '@', '/', '.' for the quantitative analysis of the predictions. We illustrate the counts for '/' and '.' specifically as we assume these characters are more often present in a URL than '-' and '@'. This analysis helps to understand the importance of these features in URL classification for an imbalanced dataset.

	Correct Predictions			Incorrect Predictions		
	Max	Min	Average	Max	Min	Average
Length of URLs	620	13	35	121	14	30.46
Count '-'	18	0	0.16	10	0	0.20
Count '@'	5	0	0.0003	1	0	0.0009
Count '/'	16	0	0.46	8	2	3.17
Count '.'	12	1	2.34	5	1	2.05

Table 4: Statistical Analysis of correctly and incorrectly predicted URLs

From Table.4 it can be observed that, in the correctly predicted URLs, the maximum length of the URL is 620 characters long, with the average being 35 characters. In the incorrectly predicted URLs, the maximum length of the URL was found to be 121 characters long, with the average URL length being 30 characters long. These statistical features were found to be much higher in benign URLs than malicious URLs for correctly predicted URLs. The benign URLs that depict malicious URL characteristics (having lesser or no lexical features present) may be misclassified and vice-versa. This makes the problem of URL classification challenging as benign and malicious URLs may display similar characteristics. Hence, it is essential to examine the different features of a URL altogether to be able to classify it better.

Quantitative Features for Correct Predictions: Count '/' and Count '.'

Figure.14 illustrates the distribution of '/' for both the class labels in correctly predicted URLs where the feature count in benign URLs is much higher than in malicious URLs. The higher count of '/' shows that most benign URLs have more directories, which is also an indication of a well-structured website that can be long-lived. These benign URLs contain between 2-16 counts of '/' (with most of them containing 2-7), and malicious URLs contain between 2-4 counts of '/'. Though, if a malicious URL displays similar properties to a benign URL (such as having count '/' more than 7), it might lead to misclassification. Sometimes, there may also be a dependence on the misspellings of words and special characters appearing together, which may cause misclassification. The normalized graph for count '/' for correctly predicted URLs is illustrated in Figure.15

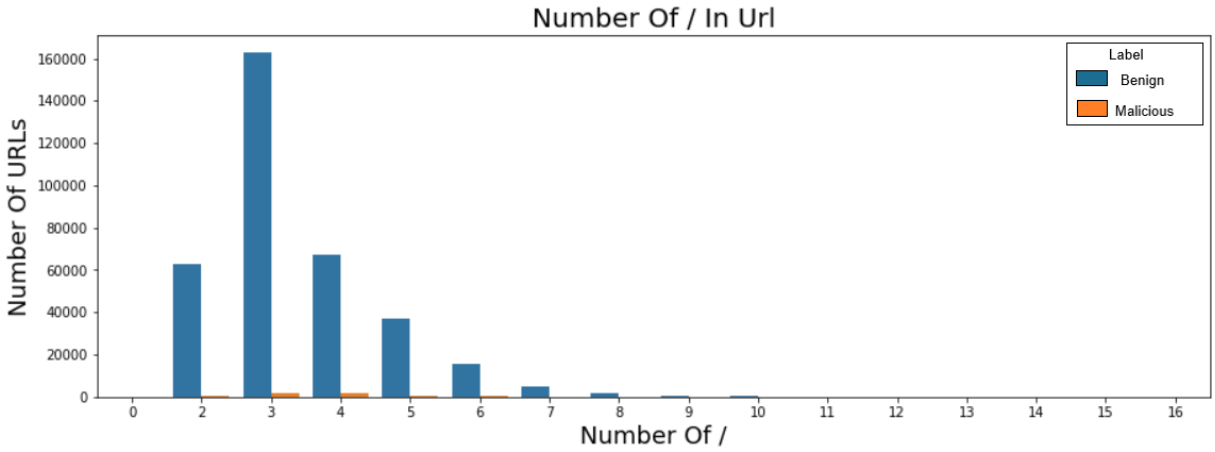

Figure 14: Count '/' in correctly predicted URLs

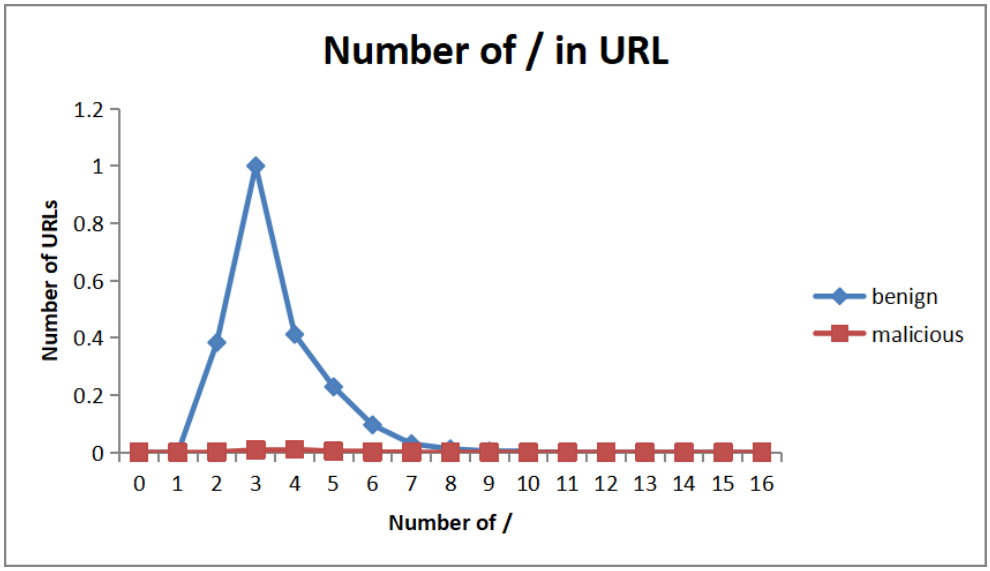

Figure 15: Normalized graph of Count '/' in correctly predicted URLs

Figure.16 shows the distribution of '.' (dots) for both class labels in correctly predicted URLs, the feature count in benign URLs is much higher than in malicious URLs. It was observed that malicious URLs either did not contain any '.' present in the URLs or ranged between 1-4. Most of the benign URLs that were correctly predicted contained 1-12 counts of '.'. This indicates that if a malicious URL contains more than four counts of '.' (displaying characteristics of a benign URL) it may be classified as a benign URL. The normalized graph for count '.' for correctly predicted URLs is illustrated in Figure.17

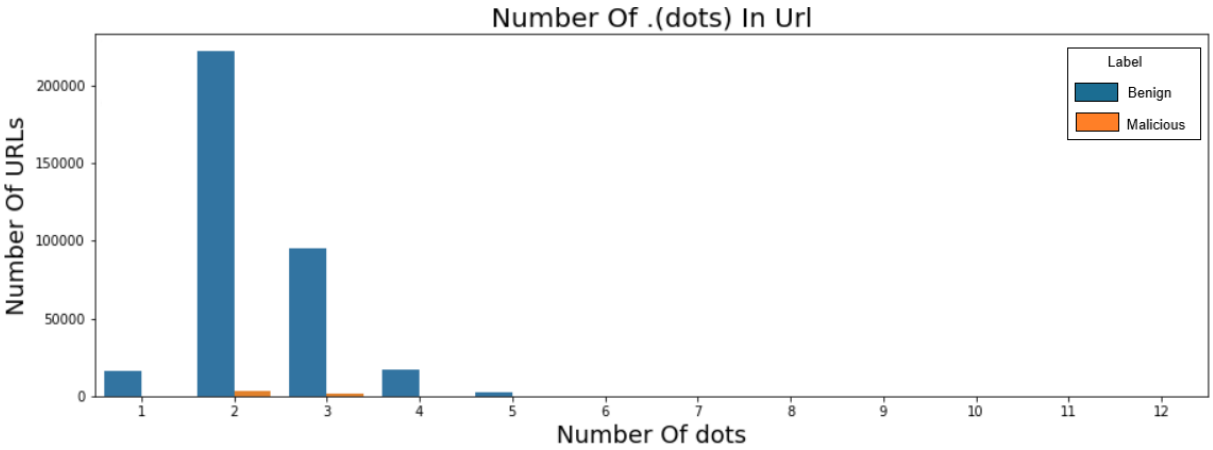

Figure 16: Count '.' in correctly predicted URLs

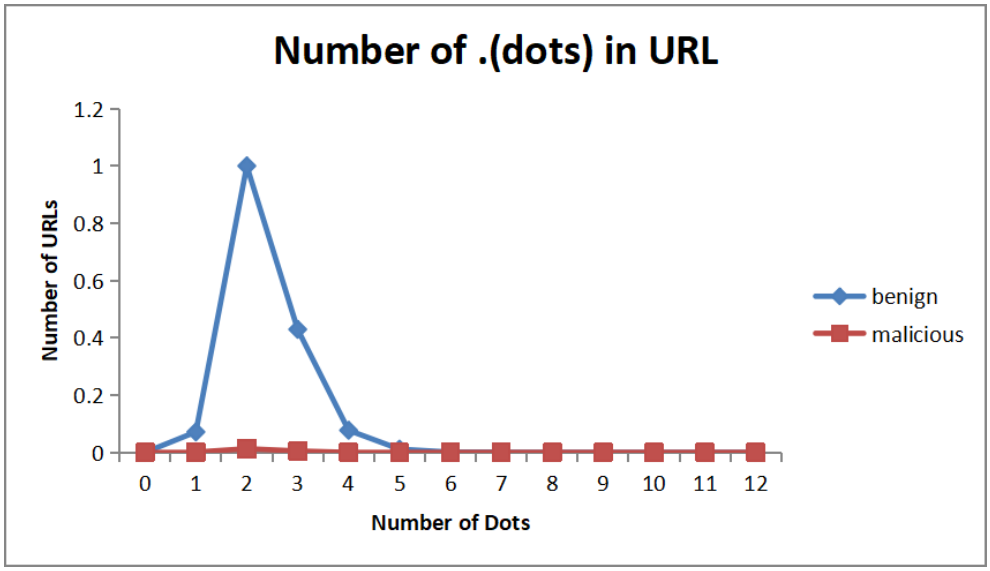

Figure 17: Normalized graph of Count '.' in correctly predicted URLs

Quantitative Features for Incorrect Predictions: Count '/' and Count '.'

Figure.18 shows the distribution of '/' for both class labels in the incorrectly predicted URLs. We observed that the count of '/' in benign URLs(maximum count of '/' being three and ranging from 2-7) is much higher than in malicious URLs (maximum count of '/' being three and ranging from 2-4). Here, the malicious URLs having a similar structure to benign URLs have been misclassified due to the similar structure. Due to this, the model might confuse itself to predict the malicious URLs as benign and vice versa. We also illustrate the normalized graph for count '/' in incorrectly predicted URLs in Figure.19

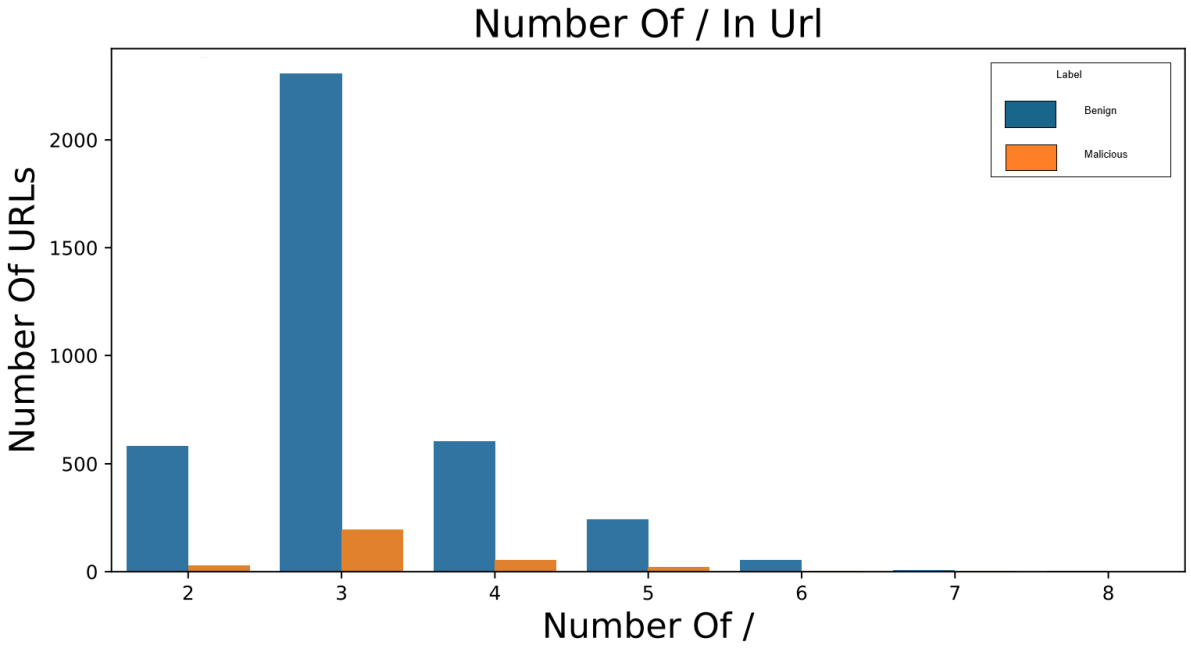

Figure 18: Count '/' in incorrectly predicted URLs

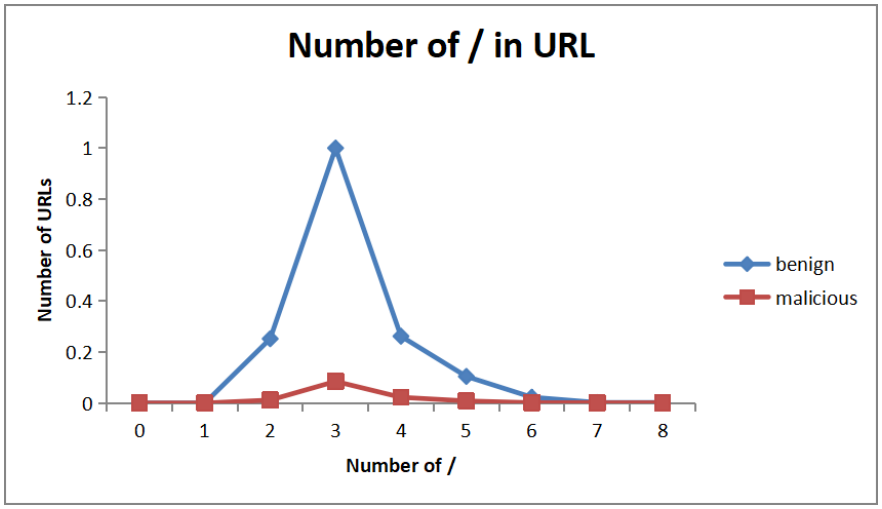

Figure 19: Normalized graph of Count '/' in incorrectly predicted URLs

Figure.20 shows the distribution of '.' for both class labels in the incorrectly predicted URLs where the count of the feature in malicious URLs is much lesser than in benign URLs. Malicious URLs that were incorrectly predicted contained count '.' ranging between 1-4 (maximum at 2), whereas the misclassified benign URLs contained count '.' ranging from 1-5 (maximum at 2), indicating that malicious and benign URLs that were misclassified were showing characteristics of each other. The normalized graph of count '.' in incorrectly predicted URLs is illustrated in Figure.21

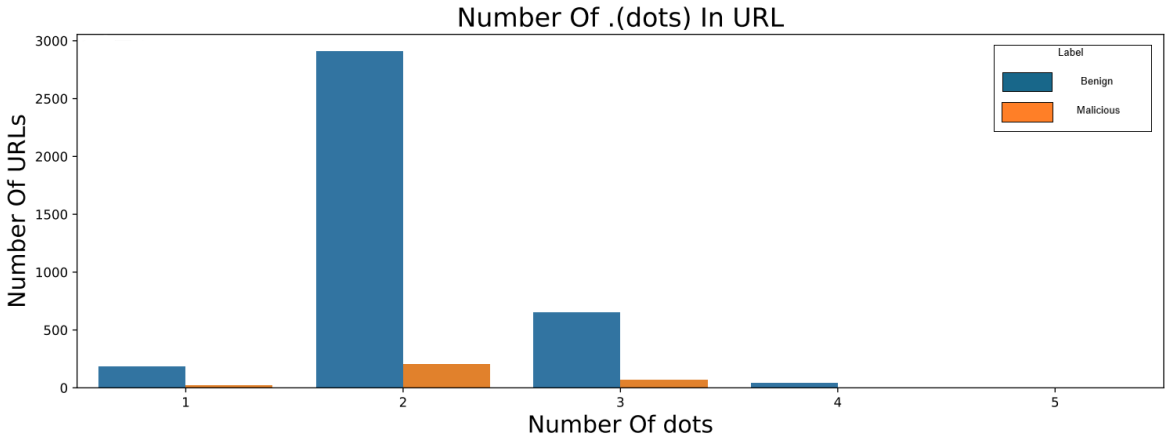

Figure 20: Count '.' in incorrectly predicted URLs

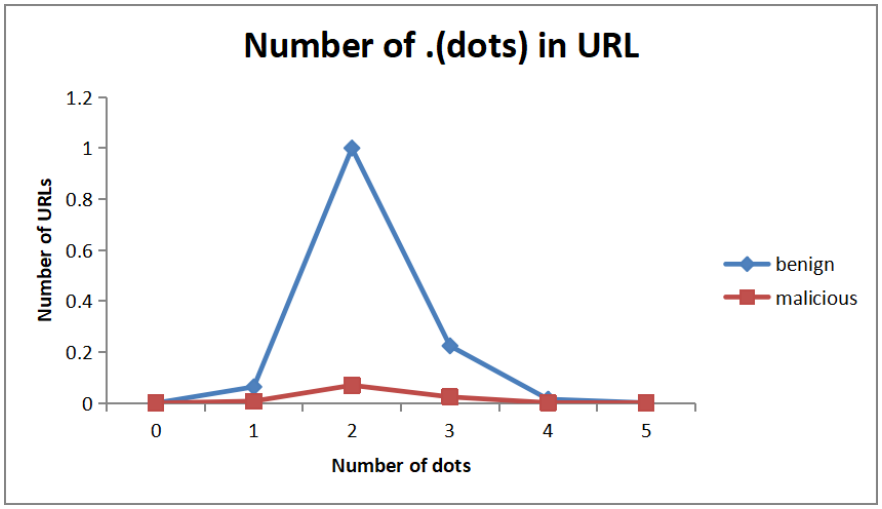

Figure 21: Count '.' in incorrectly predicted URLs

We observed that when URLs are correctly predicted, malicious URLs have a shorter length, contain fewer features (such as '/', '.') and can have incomplete URL domain names. After analysing the predictions, the number of features in incorrect predictions closely resembled that of correct predictions (having more quantitative features present) and hence were misclassified. Misclassification can also occur if there are grammatical errors in the URL (since the order of words appearing together plays an essential role in URL classification). Since they bear such a close resemblance, the models may classify malicious URLs as benign and vice versa. This may expect to have an impact on the model's performance.

7 Experiments and Results

In this section, we discuss the experiments conducted and their results obtained. When training the models with an imbalanced dataset, the models can be biased towards the majority class (in this case, benign URLs) and may produce erroneous results (i.e, results that show the model has overfit). For such reasons, we consider two metrics mainly to evaluate our work; average precision and balanced accuracy. Since our benchmark dataset is heavily imbalanced, these metrics prove to be the most informative and relevant for classification of the URLs and understanding the performance of the models. We also illustrate the best F1-scores and conduct experiments on a balanced dataset for all the discussed models.

7.1 Results : Tree-Based Ensemble Machine Learning Models

The two ensemble based machine learning models undergo the same pre-processing methods (see Figure.2). The Mendeley dataset (imbalanced) used to conduct the experiments is split by default into training and testing sets. 77% of all the URLs present in the Mendeley dataset are used for training and the rest for testing. Similarly, 87% of the balanced dataset is used for training and the rest for testing. The default hyperparameters were used for training of the ensemble methods. The results obtained from these models were recorded using the tool MLFlow[27], which helps to keep track of the successes or failures of the experiments.

The classification reports and the confusion matrices for both the ensemble based supervised machine learning models can be found in Appendix A.1. We present the average precision and balanced accuracy obtained on the Mendeley dataset and the balanced version of the Mendeley dataset in Table.5 and Table.6 respectively.

Mendeley Dataset (Imbalanced)	***Average Precision***	***Balanced Accuracy***
Random Forest	0.017	**0.649**
XGBoost	**0.018**	0.620

Table 5: Results for average precision and balanced accuracy of supervised ML models on imbalanced dataset

Balanced Dataset	***Average Precision***	***Balanced Accuracy***
Random Forest	0.017	0.726
XGBoost	0.017	**0.770**

Table 6: Results for average precision and balanced accuracy of supervised ML models on balanced dataset

From the experiments conducted for two ensemble models, we observed that the average precision scores were poor. This may suggest that the models predict many false positives, resulting in low average precision scores. The Random Forest classifier proved to perform the best on the imbalanced dataset as it achieved a balanced accuracy of 0.649. XGBoost performed the best on the balanced dataset by obtaining a balanced accuracy score of 0.770. Some of the advantages of using these models are that they are fast to train, easy to understand and implement.

The experiments conducted show that the Random Forest model performs better than the XGBoost model on the imbalanced dataset, and XGBoost performs better than the Random Forest on the balanced dataset in terms of balanced accuracy. It has the advantage of making use of 'majority voting' (see 4.1). XGBoost, on the other hand, is an efficient implementation of the stochastic gradient boosting algorithm and is successful due to its scalability. These models perhaps cannot capture the semantic meaning of the URLs and differentiate well between malicious and benign URLs since they obtain very low average precision scores. Ensemble models require extensive feature engineering, which can be time-consuming. Hence, from the results we obtained for these models, we can assume that such models may not be appropriate for URL classification as they may fail to capture the semantic meaning behind the URLs.

We calculate the confusion matrices for the models with the help of scikit-learn[20]. The best confusion matrix obtained on the imbalanced dataset among both the supervised machine learning models was found to be of Random forest (see Table.7). For the balanced dataset, the best confusion matrix was found to be for XGBoost (see Table.8). We define the 'best' confusion matrices to have the highest sensitivity (true positive rate) and highest specificity (true negative rate).

	Predicted Values		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	2421	5641
	Negative (-1)	631	353241

Table 7: Best confusion matrix belonging to Random Forest on the imbalanced dataset

	Predicted Values		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	5342	2720
	Negative (-1)	42735	31137

Table 8: Best confusion matrix belonging to XGBoost on the balanced dataset

7.2 Results : Deep Neural Network Models

This section compares the performances of URLNet, SNet and SNet v2. We conduct experiments on both the Mendeley dataset as well as the balanced dataset that we created. The average precision and balanced accuracy curves are calculated and compared for all the benchmark models. We also show the best confusion matrices and F1-scores obtained for all of the models.

All of the URLs in the dataset are delimted by special characters and each of the special characters are considered as words. The dataset by default was split into training and testing where 77% of all the URLs present in the dataset were used for training and rest for testing. Similarly on the balanced dataset, 87% of the URLs in the dataset were used for training and rest for testing. The models were evaluated on the validation set after every 500 batches where each batch contains 128 URLs. This way the models have a lesser chance of overfitting. The models were trained with the Adam optimizer for 5 epochs with L2 regularization, a learning rate of 0.001, and a Dropout rate of 0.5.

In order to find the best threshold value on the precision-recall curve, we indicate the Youden's Score (see 9) on the curve by finding the maximum index of precision and recall. A low precision indicates a high number of false positives and a low recall indicates high number of false negatives and vice-versa for high precision and recall. We find the confusion matrices for all the models. The classification report, the Precision-Recall curve, Validation loss curve, and Validation accuracy curve for the benchmark models have been plotted. These can be found under Appendix A.2.

The results for the benchmark models on the imbalanced and balanced dataset are discussed below.

Results for Imbalanced Dataset

Mendeley Data (Imbalanced)	**Average Precision**			**Balanced Accuracy**		
	URLNet	**SNet**	**SNet v2**	**URLNet**	**SNet**	**SNet v2**
Char-based CNN	**0.698**	0.684	0.685	0.769	**0.772**	0.754
Word-based CNN	0.418	**0.423**	**0.423**	0.623	0.627	**0.632**
Char and Word-based CNN	**0.698**	0.690	0.689	**0.769**	0.763	0.753
Char-level Word CNN	0.542	**0.543**	0.536	0.671	**0.681**	0.679
Char and Char-level Word CNN	**0.698**	0.693	0.691	0.749	**0.763**	0.758

Table 9: Average precision and balanced accuracy of benchmark models for imbalanced data

Table.9 illustrates the results on the imbalanced dataset. The average precision and balanced accuracy score for all methods of URLNet, SNet and SNet v2 have been calculated.

Our proposed model, SNet, obtains higher average precision scores than the benchmark model URLNet for *'Word-based CNN'* and *'Char-level Word CNN'* by achieving scores of 0.423 and 0.543 respectively. Similarly, for SNet, the *'Char-based CNN'*, *'Char-level Word CNN'* and *'Char and Char-level Word CNN'* obtains higher balanced accuracy score than the benchmark model, URLNet, by achieving scores of 0.772, 0.681 and 0.763 respectively.

SNet v2 obtains a higher average precision compared to the benchmark URLNet for *'Word-based CNN'* (similar to the *'Word-based CNN'* for SNet) by achieving an average precision of 0.423. SNet v2 obtains a better balanced accuracy score than URLNet and SNet on the *'Word-based CNN'* by achieving a score of 0.632.

The benchmark model URLNet obtains a higher average precision compared to SNet and SNet v2 for *'Char-based CNN'*, *'Char and Word-based CNN'*, and *'Char and Char-level Word CNN'* by achieving scores of 0.698, 0.698 and 0.698 respectively. It also obtains higher balanced accuracy score than SNet and SNet v2 for *'Char and Word-based CNN'* by achieving a score of 0.769.

By comparing the results of URLNet, SNet and SNet v2, we conclude that the SNet has proven to produce the best results among the models for an imbalanced dataset.

The best confusion matrices for each of the benchmark models are illustrated where the true positive rate and true negative rate are the highest on the imbalanced dataset. All the confusion matrices for the other methods of the benchmark models can be found under Appendix. A.2

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	4347	3715
	Negative (-1)	328	353544

Table 10: Best confusion matrix of URLNet on imbalanced dataset observed for 'Char and Word-based CNN'

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	4409	3653
	Negative (-1)	570	353302

Table 11: Best confusion matrix of SNet on imbalanced dataset observed for 'Char-based CNN'

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	4176	3886
	Negative (-1)	248	353624

Table 12: Best confusion matrix of SNet v2 on imbalanced dataset observed for 'Char and Char-level Word CNN'

Results for Balanced Dataset

Mendeley Data (Balanced)	**Average Precision**			**Balanced Accuracy**		
	URLNet	**SNet**	**SNet v2**	**URLNet**	**SNet**	**SNet v2**
Char-based CNN	**0.724**	0.692	0.723	**0.861**	0.860	0.854
Word-based CNN	0.423	0.421	**0.428**	0.690	0.689	**0.691**
Char and Word-based CNN	0.598	0.596	**0.600**	**0.728**	0.715	0.716
Char-level Word CNN	**0.383**	0.380	0.359	**0.662**	0.652	0.653
Char and Char-level Word CNN	**0.464**	0.418	0.388	**0.693**	0.673	0.656

Table 13: Results for average precision and balanced accuracy of benchmark models for balanced data

Table.13 illustrates the results on the balanced Mendeley dataset. The average precision and balanced accuracy score for all the benchmark models have been calculated.

The benchmark model, URLNet obtains higher average precision scores compared to SNet and SNet v2 for *'Char-based CNN'*, *'Char-level Word CNN'* and *'Char and Char-level Word CNN'* by achieving scores of 0.724, 0.383 and 0.464 respectively. URLNet also obtains a higher balanced accuracy score for *'Char-based CNN'*, *'Char and Word-based CNN'*, *'Char-level Word CNN'* and *'Char and Char-level Word CNN'* on the balanced dataset by achieving scores of 0.861, 0.728, 0.662 and 0.693 respectively.

SNet v2 obtains higher average precision scores for *'Word-based CNN'* and *'Char and Word-based CNN'* compared to URLNet and SNet by achieving scores of 0.428 and 0.600 respectively. SNet v2 also obtains higher balanced accuracy scores on *'Word-based CNN'* by achieving a score of 0.691. SNet performs competitively to URLNet and SNet v2 for the balanced Mendeley dataset.

The best confusion matrices for the benchmark models on the balanced dataset have been illustrated below where the true positive rate and true negative rate are the highest. All

the confusion matrices for the other methods of the benchmark models can be found under Appendix. A.2

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	6014	2048
	Negative (-1)	8118	345754

Table 14: Best confusion matrix of URLNet on balanced dataset observed for 'Char-based CNN'

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	6167	1895
	Negative (-1)	15632	338240

Table 15: Best confusion matrix of SNet on balanced dataset observed for 'Char-based CNN'

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	5871	2191
	Negative (-1)	7000	346872

Table 16: Best confusion matrix of SNet v2 on balanced dataset observed for 'Char-based CNN'

In Table.17, we illustrate the F1 scores (see 3) across all the models. This metric takes both False Positives (FP) and False Negatives (FN) into account. Higher the values of precision and recall, higher would be the F1-score as well. The perfect F1-score is 1, and the closer the F1-score of the model is to 1, the better it is. From the experiments conducted, we observe that the best F1-score of 0.68 belonged to both SNet and URLNet indicating that both the models perform similar in accounting for false positives and false negatives. For URLNet, this best score was obtained for '*Char-based CNN and Char and Word-based CNN*', whereas for SNet, this score was obtained for '*Char-based CNN*'.

F1-Score	***Imbalanced Dataset***		***Balanced Dataset***	
	Malicious	**Benign**	**Malicious**	**Benign**
Random Forest	0.44	0.99	0.17	0.93
XGBoost	0.38	0.99	0.19	0.93
URLNet	***Char-based CNN, Char and Word-based CNN***		***Char and Word-based CNN***	
	0.68	0.99	0.61	0.99
SNet	***Char-based CNN***		***Char and Word-based CNN***	
	0.68	0.99	0.59	0.99
SNet v2	***Char and Char-level Word CNN***		***Char and Word-based CNN***	
	0.67	0.99	0.59	0.99

Table 17: Table illustrating the best F1 scores across all the models on imbalanced and balanced Mendeley Data

In real-world scenarios, the number of malicious websites may be lesser than benign websites, making it harder to classify malicious websites. SNet and the SNet v2 provide emphasis on the features present in the 'malicious' URLs, which allows the models to perform well and hence be able to capture the semantic meaning of the URLs better. From the experiments conducted, SNet has performed better than all the other models when performing URL classification on an imbalanced dataset. URLNet performed better than the other models on the balanced dataset, with SNet v2 and SNet performing competitively.

8 Conclusion and Future Scope

In this paper, we proposed two novel methods SNet and its variant SNet v2 for classifying URLs using an attention mechanism. We also experimented with two of the most popular tree-based ensemble machine learning models, Random Forest and XGBoost, as a baseline to compare to our proposed methods. Compared to the deep neural network models, the tree-based ensemble machine learning models did not perform competitively as the average precision proved to be much lesser, indicating that the ensemble models might be classifying most of the positive labels as false positives and vice versa. Even though the training time is much shorter compared to the neural network models, these models perhaps fail to perform well as they are based on bag-of-words and URLs consist sequence based terms. This shows that the tree-based ensemble machine learning models might not be the best choice for such URL classification tasks. In order to further improve the performance of the ensemble methods, we can perform any one of the hyperparameter tuning methods such as bayesian optimization to obtain the best hyperparameters and perform cross-validation that may increase the performance of these models.

The importance of special characters in the URL inspired us to apply attention to the char/word embeddings directly. SNet performed better compared to all the baseline models when performing URL classification on the imbalanced dataset. URLNet achieved the best results on the balanced dataset with SNet and SNet v2 performing competitively. In real-world scenarios, where the number of benign URLs most likely outnumber malicious URLs, SNet can perform the best compared to all other methods holistically on the imbalanced dataset. This conclusion was drawn based on the total number of outcomes where SNet achieved higher average precision and balanced accuracy scores compared to URLNet on the imbalanced dataset. This makes SNet a better URL classifier than URLNet when dealing with imbalanced datasets as it was able to achieve the highest balanced accuracy score of 0.772 when compared to the highest balanced accuracy score of 0.769 of URLNet. The downsides of using the attention mechanism is that it adds more weights to the model, slowing down the training process and making it hard to parallelize processes.

An interesting approach to further classify malicious URLs can be to set up a system that visits the websites and checks for the content present in them for misclassified URLs. If the page is empty, has grammatical errors, does not contain a trust seal or contains many suspicious words with links, such sites may be deemed as malicious.

Pan et al. proposed a paradigm for understanding transfer learning in their paper *A Survey on Transfer Learning*[28], which uses domain, task, and marginal probability. The concept

of transfer learning may be well-made use of in URL classification. With the help of transfer learning, we can leverage the features, weights and other learned representations from previously trained models (such as the attention mechanism in SNet and SNet v2) and use them for training on newer models. This may help reduce training time and the amount of data needed for those who do not have the required computational resources. In terms of Natural Language Processing, the word embeddings from pre-trained models can be used for feature extraction. These character or word embeddings may provide the relevant information regarding the URL and can perhaps help capture the semantic meaning behind them. There can be scenarios where transfer learning might not add value to the model and instead reduce its performance, called negative transfer[29]. Therefore, based on the use case, transfer learning can be used to improve the target task performance.

In conclusion, one may give more importance to the letters and special characters appearing together in the URL while performing a classification task since a URL is unlike a regular text string containing alphabetical characters alone and varying text lengths. URLs can be made of alphanumeric characters, and 'attention' may be provided to these characters to get a better semantic meaning of them. These characters may influence a model's performance as their quantitative values may enrich the data. Since many malicious websites masquerade as legitimate websites and steal information, they must must be detected and taken down at the earliest as it can affect systems and causes considerable losses to the economy.

References

[1] MDN Contributors. What is a url? https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL. Accessed: 2022-06-04.

[2] Damjan Jugovic Spajic. Piracy is back: Piracy statistics for 2022. https://dataprot.net/statistics/piracy-statistics/, note = Accessed: 2022-06-18.

[3] K Thirumoorthy and K Muneeswaran. Feature selection for text classification using machine learning approaches. *National Academy Science Letters*, pages 1–6, 2021.

[4] Martyn Weedon, Dimitris Tsaptsinos, and James Denholm-Price. Random forest explorations for url classification. In *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–4. IEEE, 2017.

[5] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. Urlnet: Learning a url representation with deep learning for malicious url detection. *arXiv preprint arXiv:1802.03162*, 2018.

[6] Hoa T Le, Christophe Cerisara, and Alexandre Denis. Do convolutional networks need to be deep for text classification? In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] Ebubekir Buber and Banu Diri. Web page classification using rnn. *Procedia Computer Science*, 154:62–72, 2019.

[8] Yongjie Huang, Qiping Yang, Jinghui Qin, and Wushao Wen. Phishing url detection via cnn and attention-based hierarchical rnn. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 112–119. IEEE, 2019.

[9] Linqing Shi, Zeping Yu, and Gongshen Liu. Extensive pyramid networks for text classification. *Aust. J. Intell. Inf. Process. Syst.*, 17(1):17–23, 2019.

[10] AMIT KUMAR (2020) SINGH. "dataset of malicious and benign webpages". *Mendeley Data, V2*. doi:10.17632/gdx3pkwp47.2.

[11] MalCrawler. https://www.malcrawler.com/. Accessed: 2022-06-18.

[12] Google. Safe browsing api. https://developers.google.com/safe-browsing. Accessed: 2022-06-18.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[14] AK Singh and Navneet Goyal. A comparison of machine learning attributes for detecting malicious websites. In *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, pages 352–358. IEEE, 2019.

[15] Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, and Fabio A González. Classifying phishing urls using recurrent neural networks. In *2017 APWG symposium on electronic crime research (eCrime)*, pages 1–8. IEEE, 2017.

[16] Tianyang Zhang, Minlie Huang, and Li Zhao. Learning structured representation for text classification via reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] Peng-Yeng Yin, Kuang-Cheng Chang, et al. Reinforcement learning for combining relevance feedback techniques. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 510–515. IEEE, 2003.

[18] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] William J Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

[22] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[23] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. URL: `http://doi.acm.org/10.1145/2939672.2939785`, `doi:10.1145/2939672.2939785`.

[24] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[25] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL: `http://jmlr.org/papers/v18/16-365.html`.

[26] Zac Harris. Www vs non www: Which is right from seo perspective. `https://www.benchmarkdesign.net/index.php/announcements/13/WWW-vs-Non-WWW-Which-is-Right-from-SEO-Perspective.html`. Accessed: 2022-06-10.

[27] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018.

[28] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[29] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

A Appendix

In this section, we list the confusion matrices for all the models along with the precision-recall curve with the optimum threshold (obtained from Youden's J score), the validation loss and accuracy curves for URLNet, SNet and SNet v2. For all the ensemble based supervised machine learning models, we illustrate the classification report along with confusion matrices. The classification report consists of precision, recall, F1, and accuracy which has been discussed in Section. 3. The samples present in the dataset are labelled as 'malicious' and 'benign'. Malicious URLs represent websites which may lead to malware, phishing attacks or even contain links to pirated digital content. Benign URLs are genuine and safe websites, where, upon clicking on them they would not lead to any harm to systems or would not steal personal information unlike malicious URLs.

Accuracy displays the sum of true positives and true negatives divided by the total number of samples. Accuracy although is not an appropriate measure of performance as the models may easily be able to predict the majority class but unable to predict the minority class. The macro average score is a score calculated against precision, recall and F1 where it is the unweighted mean (regardless of number of samples per class) of per-class. The weighted average considers the weighted mean (while considering number of samples per class) of per-class precision, recall and F1 scores.

A.1 Tree-Based Ensemble Machine Learning Models

The classification reports, confusion matrices and results obtained for Random Forest and XGBoost are illustrated in this section. The results for the balanced (weighted) versions of Random Forest and XGBoost were obtained on the Mendeley Dataset. By doing so, we control the balance of positive and negative weights, which is useful for unbalanced classes. This value used for the balanced version of the ensemble methods was calculated by taking the sum (negative instances) / sum (positive instances). In our case, this value was approximated to 44. For the weighted versions of the ensemble models, we illustrate the average precision and balanced accuracy.

1. Random Forest

In this section, we illustrate the classification reports and confusion matrices for the Random Forest Classifier.

1.1 Imbalanced Dataset

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.79	0.30	0.44	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.89	0.65	0.71	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	2421	5641
	Negative (-1)	631	353241

1.2 Balanced Dataset

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.10	0.58	0.17	8062
Benign	0.99	0.88	0.93	353872
Accuracy			0.87	361934
Macro Average	0.54	0.73	0.55	361934
Weighted Average	0.97	0.87	0.91	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	4637	3425
	Negative (-1)	43102	310770

1.3 Weighted Version of Random Forest

Average Precision	0.017
Balanced Accuracy	0.640

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.83	0.28	0.42	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.91	0.64	0.71	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	2273	5789
	Negative (-1)	451	353421

2. XGBoost

In this section, we illustrate the classification reports and confusion matrices for the XGBoost Classifier.

2.1 Imbalanced Dataset

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.89	0.24	0.38	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.94	0.62	0.69	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	1947	6115
	Negative (-1)	235	353637

2.2 Balanced Dataset

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.11	0.66	0.19	8062
Benign	0.99	0.88	0.93	353872
Accuracy			0.97	361934
Macro Average	0.55	0.77	0.56	361934
Weighted Average	0.97	0.87	0.92	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	5342	2720
	Negative (-1)	42735	311137

2.3 Weighted Version of XGBoost

Average Precision	0.019
Balanced Accuracy	0.589

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.99	0.18	0.30	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.99	0.59	0.65	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	1437	6625
	Negative (-1)	12	353860

A.2 Deep Neural Network Models

In this section, we illustrate the classification reports for URLNet, SNet and SNet v2. Along with the classification report, we also illustrate the precision-recall curve with Youden's J score marked on the curve (indicating the point at which the precision and recall is maximum), the validation and accuracy curves.

1. URLNet

In this section, we illustrate the classification reports, confusion matrices and curves on the imbalanced dataset (Mendeley Dataset) and the balanced dataset for URLNet.

1.1 Imbalanced Data

Char-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.53	0.68	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.76	0.84	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4267	3795
	Negative (-1)	299	353573

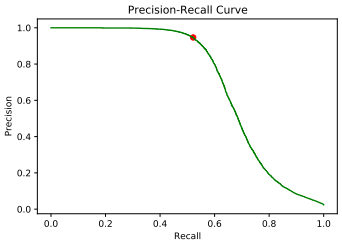

Figure A.1: Precision-Recall curve with Youden's J score of 0.548

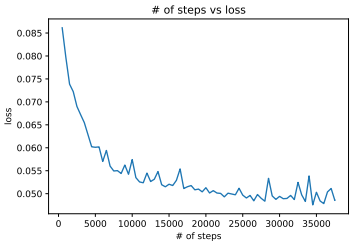

Figure A.2: Validation loss curve

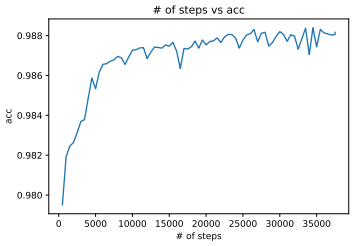

Figure A.3: Validation Accuracy Curve

Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.94	0.25	0.39	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.96	0.62	0.69	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	1988	6074
	Negative (-1)	118	3535754

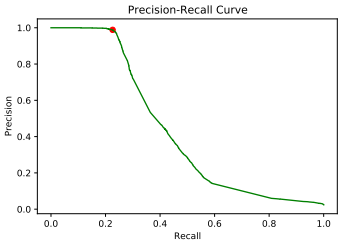

Figure A.4: Precision-Recall curve with Youden's J score of 0.800

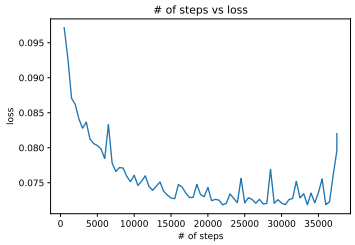

Figure A.5: Validation loss curve

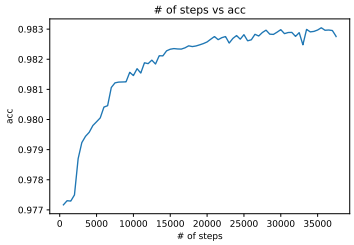

Figure A.6: Validation Accuracy Curve

Char and Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.54	0.68	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.77	0.84	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4347	3715
	Negative (-1)	328	353544

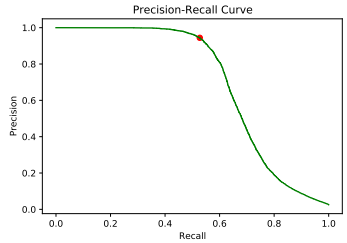

Figure A.7: Precision-Recall curve with Youden's J score of 0.557

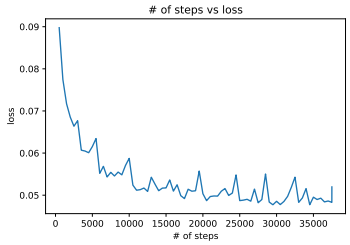

Figure A.8: Validation loss curve

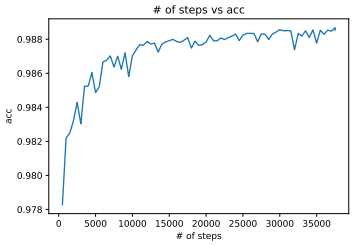

Figure A.9: Validation Accuracy Curve

Character-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.97	0.34	0.51	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.98	0.67	0.75	361934
Weighted Average	0.98	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2762	5300
	Negative (-1)	97	353775

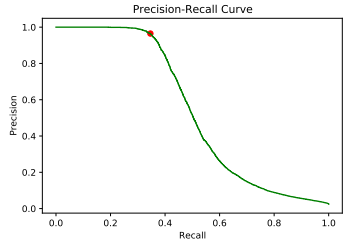

Figure A.10: Precision-Recall curve with Youden's J score of 0.474

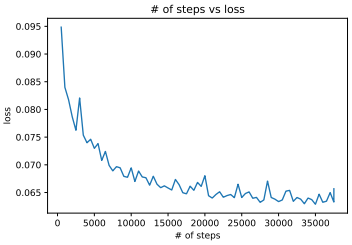

Figure A.11: Validation loss curve

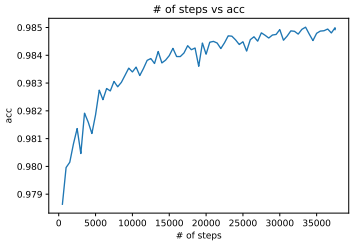

Figure A.12: Validation Accuracy Curve

Char and Char-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.97	0.50	0.66	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.98	0.75	0.83	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4323	3739
	Negative (-1)	277	353595

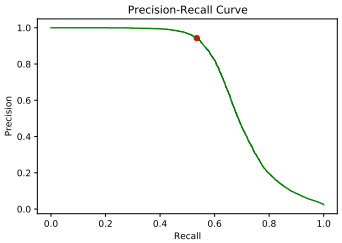

Figure A.13: Precision-Recall curve with Youden's J score of 0.432

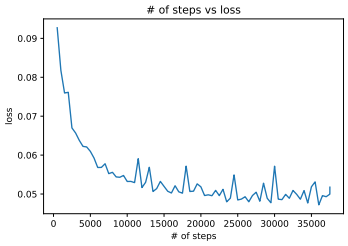

Figure A.14: Validation loss curve

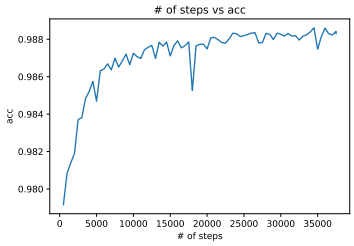

Figure A.15: Validation Accuracy Curve

1.2 Balanced Data

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.43	0.75	0.54	8062
Benign	0.99	0.98	0.99	353872
Accuracy			0.97	361934
Macro Average	0.71	0.86	0.76	361934
Weighted Average	0.98	0.97	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	6014	2048
	Negative (-1)	8118	345754

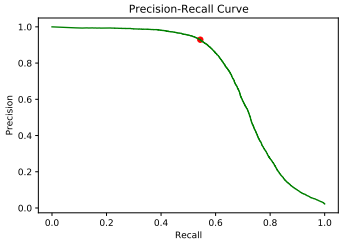

Figure A.16: Precision-Recall curve with Youden's J score of 0.988

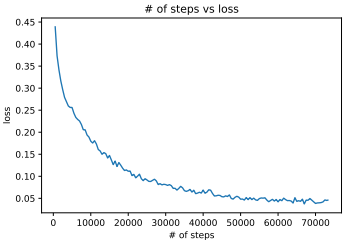

Figure A.17: Validation loss curve

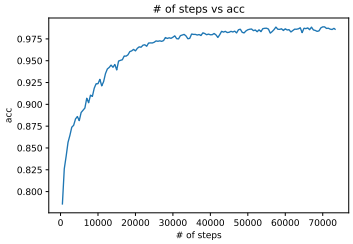

Figure A.18: Validation Accuracy Curve

Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.50	0.39	0.44	8062
Benign	0.99	0.99	0.99	353872
Accuracy			0.98	361934
Macro Average	0.74	0.69	0.71	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	3142	4920
	Negative (-1)	3150	350722

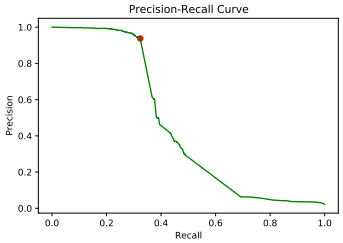

Figure A.19: Precision-Recall curve with Youden's J score of 0.868

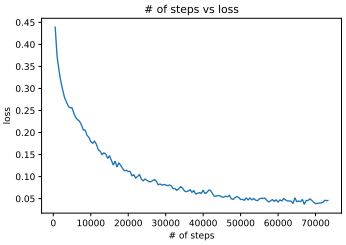

Figure A.20: Validation loss curve

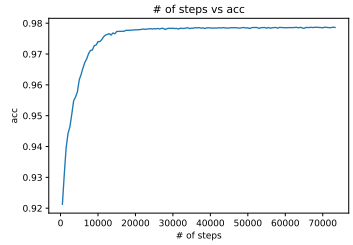

Figure A.21: Validation Accuracy Curve

Char and Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.46	0.61	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.73	0.80	361934
Weighted Average	0.99	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	3684	4378
	Negative (-1)	287	353585

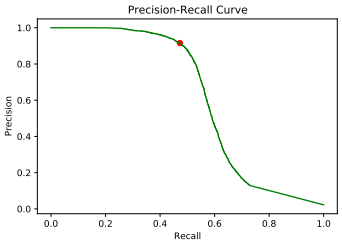

Figure A.22: Precision-Recall curve with Youden's J score of 0.314

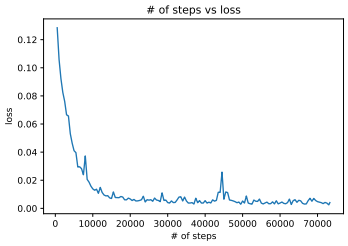

Figure A.23: Validation loss curve

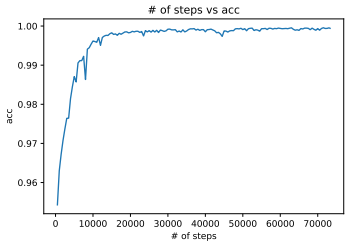

Figure A.24: Validation Accuracy Curve

Character-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.78	0.33	0.46	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.88	0.66	0.73	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2632	5430
	Negative (-1)	731	353141

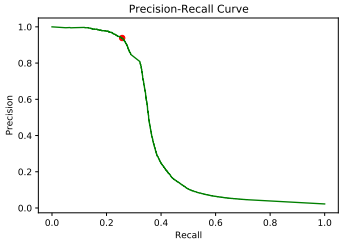

Figure A.25: Precision-Recall curve with Youden's J score of 0.970

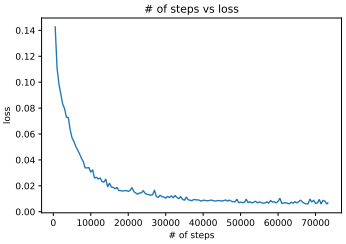

Figure A.26: Validation loss curve

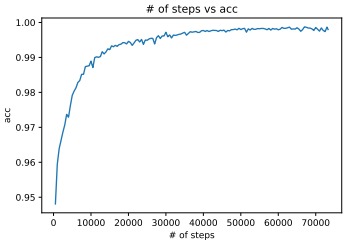

Figure A.27: Validation Accuracy Curve

Char and Char-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.89	0.39	0.54	8062
Benign	0.89	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.94	0.69	0.77	361934
Weighted Average	0.98	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	3122	4940
	Negative (-1)	399	353473

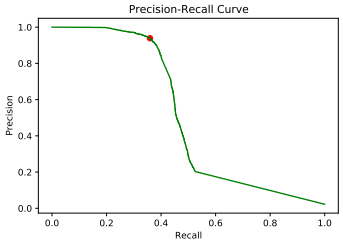

Figure A.28: Precision-Recall curve with Youden's J score of 0.903

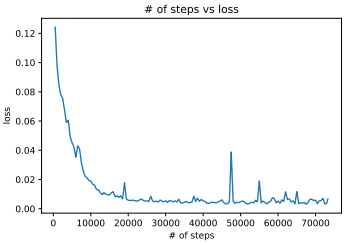

Figure A.29: Validation loss curve

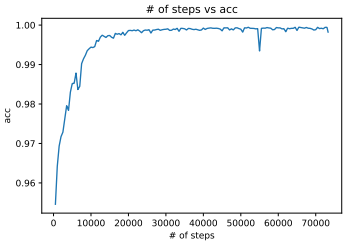

Figure A.30: Validation Accuracy Curve

2. SNet

In this section, we illustrate the classification reports, confusion matrices and the curves on the imbalanced dataset (Mendeley Dataset) and the balanced dataset for SNet.

2.1 Imbalanced Data

Char-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.89	0.55	0.68	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.94	0.77	0.84	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4409	3653
	Negative (-1)	570	353302

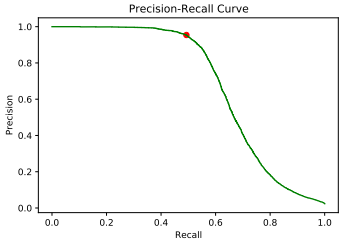

Figure A.31: Precision-Recall curve with Youden's J score of 0.800

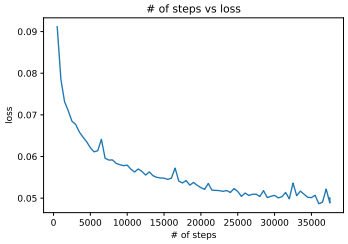

Figure A.32: Validation loss curve

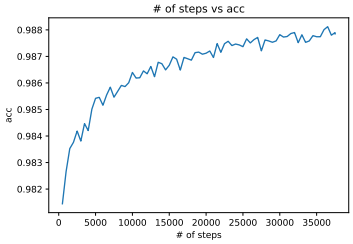

Figure A.33: Validation Accuracy Curve

Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.95	0.26	0.40	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.97	0.63	0.70	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4409	3653
	Negative (-1)	570	353302

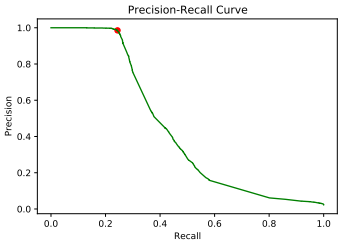

Figure A.34: Precision-Recall curve with Youden's J score of 0.699

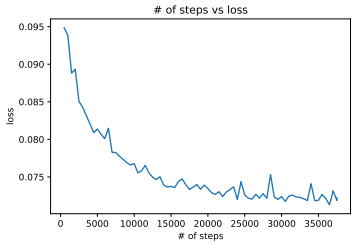

Figure A.35: Validation loss curve

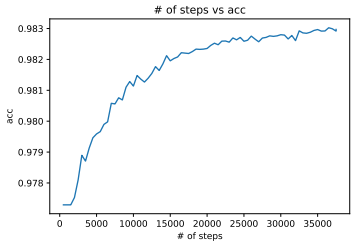

Figure A.36: Validation Accuracy Curve

Char and Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.92	0.53	0.67	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.76	0.83	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4255	3807
	Negative (-1)	360	353512

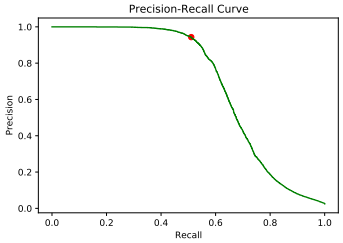

Figure A.37: Precision-Recall curve with Youden's J score of 0.572

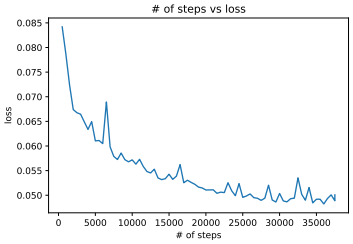

Figure A.38: Validation loss curve

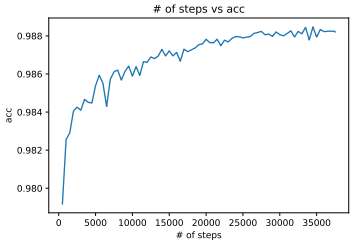

Figure A.39: Validation Accuracy Curve

Character-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.94	0.36	0.52	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.68	0.76	361934
Weighted Average	0.98	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4255	3807
	Negative (-1)	360	353512

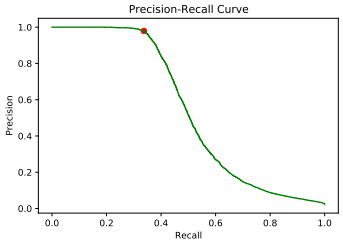

Figure A.40: Precision-Recall curve with Youden's J score of 0.668

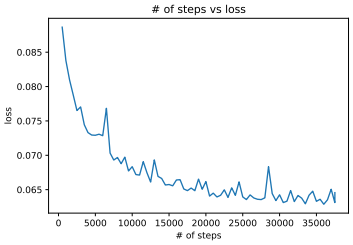

Figure A.41: Validation loss curve

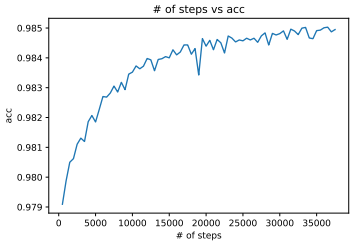

Figure A.42: Validation Accuracy Curve

Char and Char-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.53	0.67	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.76	0.83	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4253	3809
	Negative (-1)	318	353554

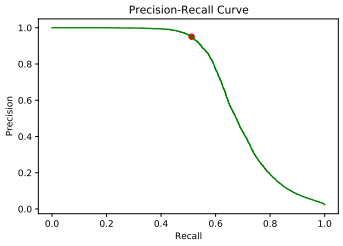

Figure A.43: Precision-Recall curve with Youden's J score of 0.571

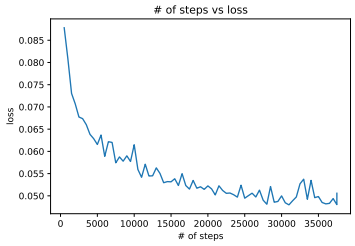

Figure A.44: Validation loss curve

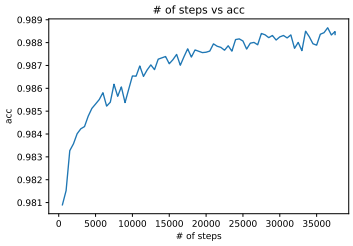

Figure A.45: Validation Accuracy Curve

2.2 Balanced Data

Char-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.28	0.76	0.41	8062
Benign	0.99	0.96	0.97	353872
Accuracy			0.95	361934
Macro Average	0.64	0.86	0.69	361934
Weighted Average	0.98	0.95	0.96	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	6167	1895
	Negative (-1)	15632	338240

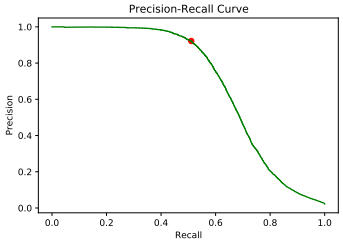

Figure A.46: Precision-Recall curve with Youden's J score of 0.973

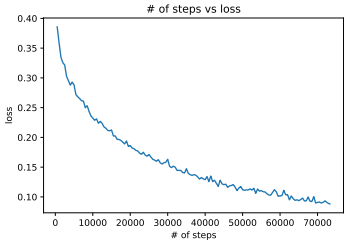

Figure A.47: Validation loss curve

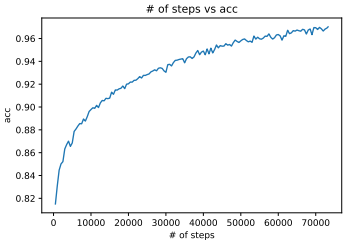

Figure A.48: Validation Accuracy Curve

Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.49	0.39	0.44	8062
Benign	0.99	0.99	0.99	353872
Accuracy			0.98	361934
Macro Average	0.74	0.69	0.71	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
Actual Values		*Positive (1)*	*Negative (-1)*
	Positive (1)	3219	4933
	Negative (-1)	3195	350677

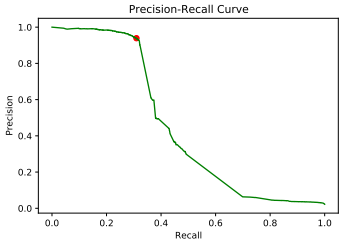

Figure A.49: Precision-Recall curve with Youden's J score of 0.909

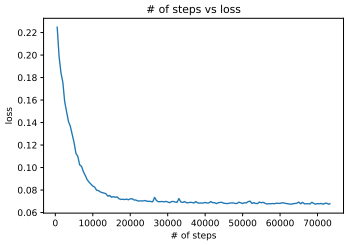

Figure A.50: Validation loss curve

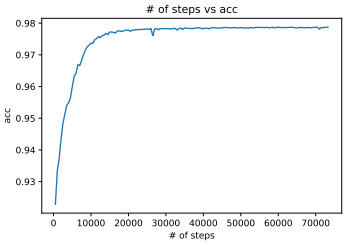

Figure A.51: Validation Accuracy Curve

Char and Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.92	0.43	0.59	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.95	0.72	0.79	361934
Weighted Average	0.99	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	3487	4575
	Negative (-1)	302	353570

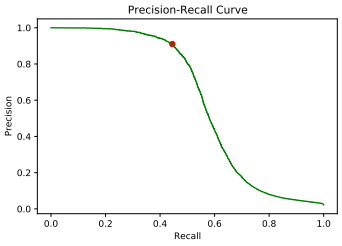

Figure A.52: Precision-Recall curve with Youden's J score of 0.274

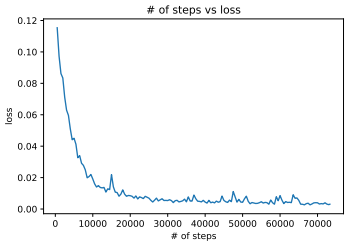

Figure A.53: Validation loss curve

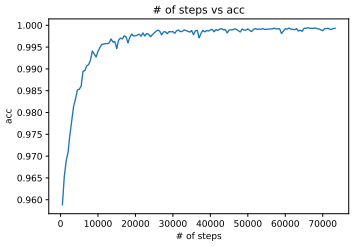

Figure A.54: Validation Accuracy Curve

Character-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.76	0.31	0.44	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.87	0.65	0.71	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2472	5590
	Negative (-1)	790	353082

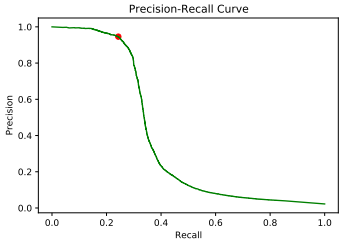

Figure A.55: Precision-Recall curve with Youden's J score of 0.987

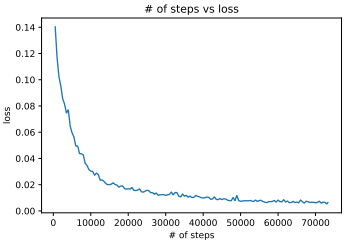

Figure A.56: Validation loss curve

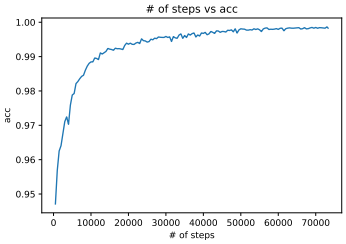

Figure A.57: Validation Accuracy Curve

Char and Char-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.87	0.35	0.50	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.93	0.67	0.74	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2805	5257
	Negative (-1)	418	353454

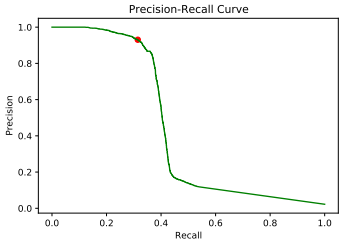

Figure A.58: Precision-Recall curve with Youden's J score of 0.939

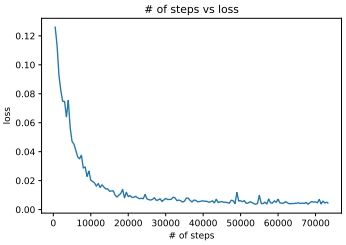

Figure A.59: Validation loss curve

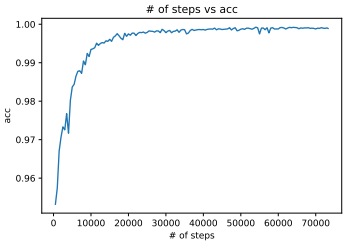

Figure A.60: Validation Accuracy Curve

3. SNet v2

In this section, we illustrate the classification reports, confusion matrices and the curves on the imbalanced dataset (Mendeley Dataset) and the balanced dataset for SNet v2.

3.1 Imbalanced Data

Char-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.57	0.66	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.75	0.83	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4113	3949
	Negative (-1)	292	353580

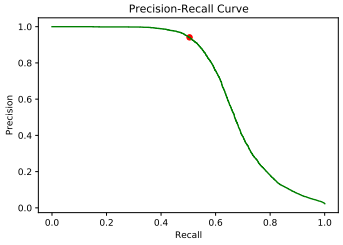

Figure A.61: Precision-Recall curve with Youden's J score of 0.529

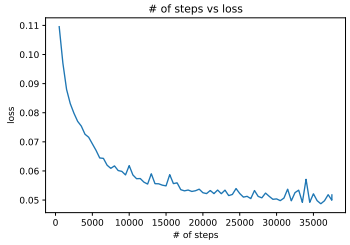

Figure A.62: Validation loss curve

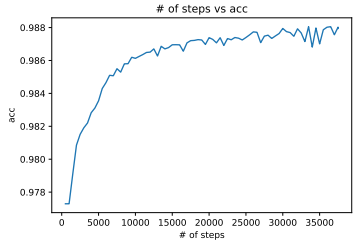

Figure A.63: Validation Accuracy Curve

Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.92	0.26	0.41	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.95	0.63	0.70	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2136	5926
	Negative (-1)	185	353687

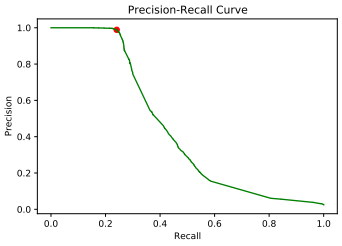

Figure A.64: Precision-Recall curve with Youden's J score of 0.823

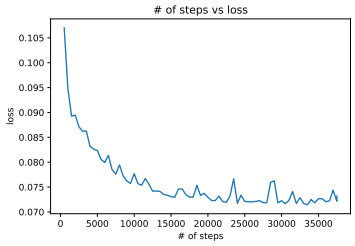

Figure A.65: Validation loss curve

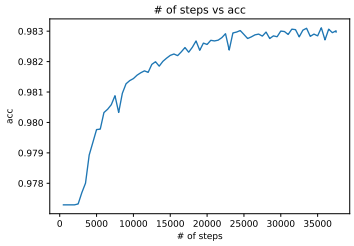

Figure A.66: Validation Accuracy Curve

Char and Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.96	0.57	0.66	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.98	0.75	0.83	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4089	3973
	Negative (-1)	155	353717

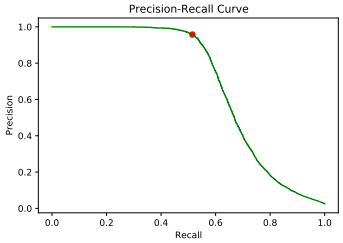

Figure A.67: Precision-Recall curve with Youden's J score of 0.460

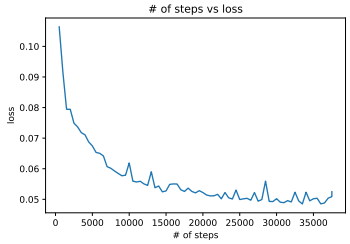

Figure A.68: Validation loss curve

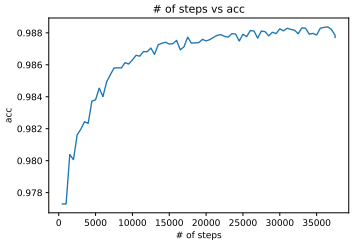

Figure A.69: Validation Accuracy Curve

Character-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.36	0.52	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.68	0.75	361934
Weighted Average	0.98	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2893	5169
	Negative (-1)	234	353638

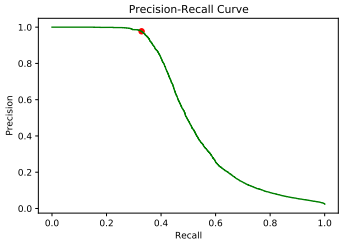

Figure A.70: Precision-Recall curve with Youden's J score of 0.661

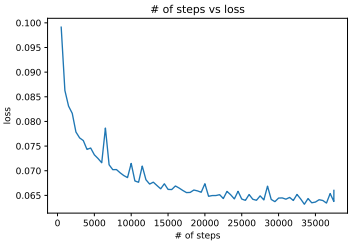

Figure A.71: Validation loss curve

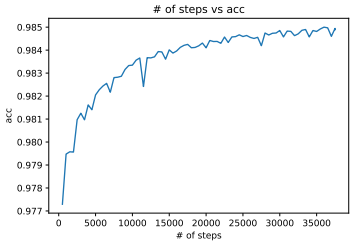

Figure A.72: Validation Accuracy Curve

Char and Char-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.94	0.52	0.67	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.97	0.76	0.83	361934
Weighted Average	0.99	0.99	0.99	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	4176	3886
	Negative (-1)	248	353624

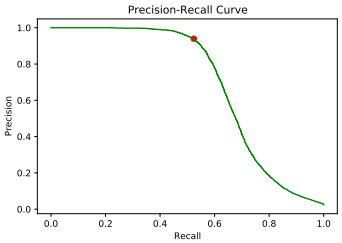

Figure A.73: Precision-Recall curve with Youden's J score of 0.485

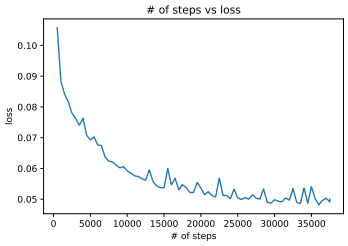

Figure A.74: Validation loss curve

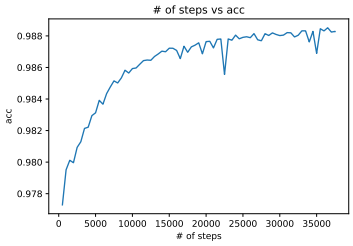

Figure A.75: Validation Accuracy Curve

3.2 Balanced Data

Char-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.48	0.73	0.56	8062
Benign	0.99	0.98	0.99	353872
Accuracy			0.97	361934
Macro Average	0.72	0.85	0.77	361934
Weighted Average	0.98	0.97	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	5871	2191
	Negative (-1)	7000	346872

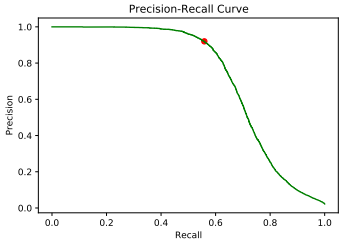

Figure A.76: Precision-Recall curve with Youden's J score of 0.976

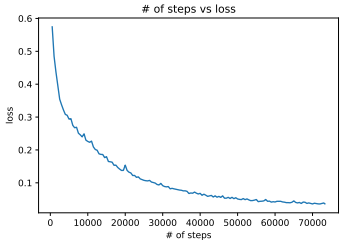

Figure A.77: Validation loss curve

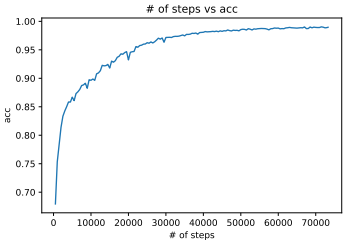

Figure A.78: Validation Accuracy Curve

Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.50	0.39	0.44	8062
Benign	0.99	0.99	0.99	353872
Accuracy			0.98	361934
Macro Average	0.74	0.69	0.71	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	3158	4904
	Negative (-1)	3221	350651

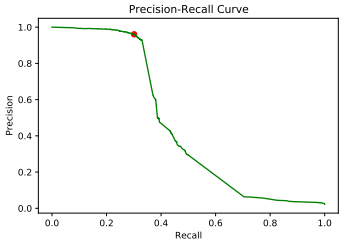

Figure A.79: Precision-Recall curve with Youden's J score of 0.991

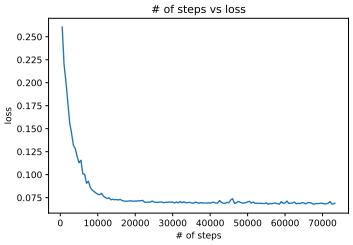

Figure A.80: Validation loss curve

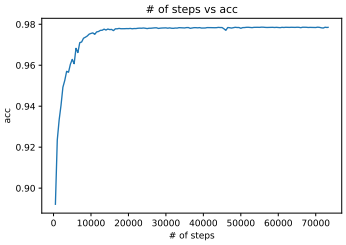

Figure A.81: Validation Accuracy Curve

Char and Word-based CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.93	0.43	0.59	8062
Benign	0.99	1.00	0.99	353872
Accuracy			0.99	361934
Macro Average	0.96	0.72	0.79	361934
Weighted Average	0.99	0.99	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	3504	4558
	Negative (-1)	248	353624

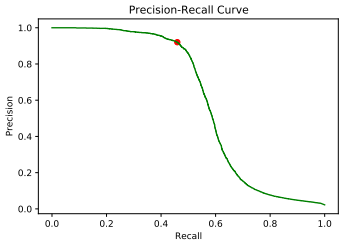

Figure A.82: Precision-Recall curve with Youden's J score of 0.185

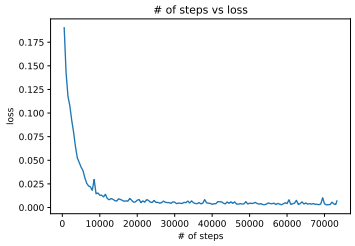

Figure A.83: Validation loss curve

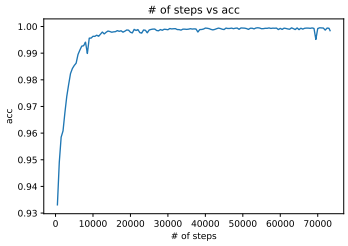

Figure A.84: Validation Accuracy Curve

Character-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.58	0.31	0.41	8062
Benign	0.98	0.99	0.99	353872
Accuracy			0.98	361934
Macro Average	0.78	0.65	0.70	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2524	5538
	Negative (-1)	1838	352034

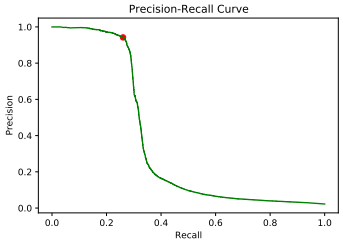

Figure A.85: Precision-Recall curve with Youden's J score of 0.987

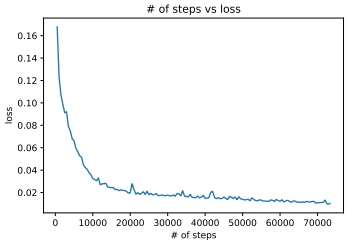

Figure A.86: Validation loss curve

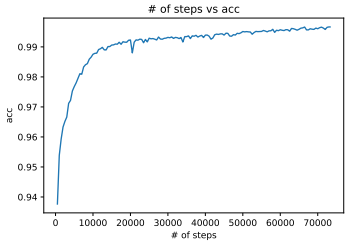

Figure A.87: Validation Accuracy Curve

Char and Char-level Word CNN

Classification Report				
	Precision	Recall	F1	Support
Malicious	0.90	0.31	0.46	8062
Benign	0.98	1.00	0.99	353872
Accuracy			0.98	361934
Macro Average	0.94	0.66	0.73	361934
Weighted Average	0.98	0.98	0.98	361934

Confusion Matrix	**Predicted Values**		
		Positive (1)	*Negative (-1)*
Actual Values	*Positive (1)*	2524	5538
	Negative (-1)	273	353599

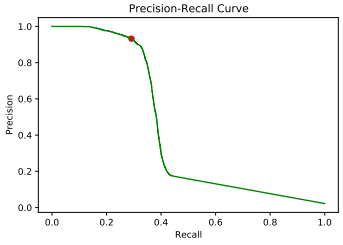

Figure A.88: Precision-Recall curve with Youden's J score of 0.935

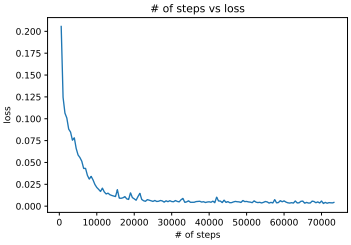

Figure A.89: Validation loss curve

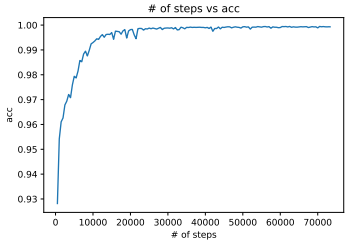

Figure A.90: Validation Accuracy Curve