An Empirical Analysis of SMS Scam Detection Systems

Muhammad Salman  
Macquarie University  
Sydney, Australia  
muhammad.salman2@students.mq.edu.au

Muhammad Ikram  
Macquarie University  
Sydney, Australia  
muhammad.ikram@mq.edu.au

Mohamed Ali Kaafar  
Macquarie University  
Sydney, Australia  
dali.kaafar@mq.edu.au

ABSTRACT
The short message service (SMS) was introduced a generation ago to the mobile phone users. They make up the world’s oldest large-scale network, with billions of users and therefore attracts a lot of fraud. Due to the convergence of mobile network with internet, SMS based scams can potentially compromise the security of internet services as well. In this study, we present a new SMS scam dataset consisting of 153,551 SMSes. This dataset that we will release publicly for research purposes represents the largest publicly-available SMS scam dataset. We evaluate and compare the performance achieved by several established machine learning methods on the new dataset, ranging from shallow machine learning approaches to deep neural networks to syntactic and semantic feature models. We then study the existing models from an adversarial viewpoint by assessing its robustness against different level of adversarial manipulation. This perspective consolidates the current state of the art in SMS Spam filtering, highlights the limitations and the opportunities to improve the existing approaches.

1 INTRODUCTION
SMS scams have reached alarming volumes in recent years. In the US in 2021, an estimated USD 86 million was lost to SMS scammers (reported by Federal Trade Commission\(^1\)). Similarly, the Australian Competition and Consumer Commission (ACCC)’s ScamWatch body\(^2\) reported a near-doubling of annual losses between 2020 (AUD 175 millions) and 2021 (AUD 323 millions). In 2021, there were 67,180 SMS fraud reports, up from 32,337 reported in 2020: more than 8,835 SMS scams were reported only in February 2022, the highest among all scam delivery methods. Despite almost two decades of research on SMS Spam detection [1–8], it continues to be a challenging and serious issue to our modern digital societies.

Researchers have developed various techniques to combat SMS Spam, and there are several surveys of these attempts [8–10]. However, unlike email spam detection, it is difficult to identify the appropriate system and directly compare the proposed systems in literature, since most of them used highly outdated and different datasets for evaluation. Although, traditional ML and deep learning (DL) methods have substantially improved Spam detection in SMS; however, scammers’ dynamic nature frequently defeats security barriers. Recently, studies have revealed different adversarial attacks in the text domain, which could effectively evade various ML-based text analyzers [8, 11–13]. We discovered that there are certain methodological issues with current SMS Spam detection research and it requires a lot more research. For example, we observe:

- The use of highly outdated and imbalanced datasets with only few hundreds of Spam messages
- Lack of benchmark, which resulted in partial and segmented evaluation of literature
- Lack of comprehensive quantification and comparison of different classes of ML such as DL, positive and unlabeled (PU) learning, traditional two class and one class ML etc. The same is observed for different set of syntactic and semantic features
- Importantly, the literature overlook new challenges that arise as convergence of Internet and Telcom especially the new adversarial attempts of scammers to evade detection

The focus of this paper is to investigate the landscape of fraudulent SMS messages, existing methods for combating SMS scams, state-of-the-art research in SMS Spam defense, points out the required set of challenges and to analyse how current tactics may be abused by scammers and subsequently identify the requirements that are crucial to an acceptable solution. Therefore, we assess the performance and robustness of the SMS Spam models proposed in the literature on the new super dataset (shortly discussed in section 3), based on the requirements of the security domain, which we elaborate upon in section 6, to determine the appropriateness of the proposed solutions. To our knowledge, a comprehensive experimental evaluation of the appropriateness of the previous research on SMS Spam from the security perspective has not been done before. To fill this important gap, this study will assist drive the development of effective/ successful SMS Spam defenses, as well as provide a framework for evaluating SMS Spam methods. The major contributions of our paper are:

**Large scale Spam Dataset.** We make available a new real, public and large SMS Spam corpus from various sources over the previous decade, that is the largest labeled dataset as far as we are aware. We aggregate and characterise publicly available Spam dataset. Given that the benchmark dataset [14] used for SMS Spam research is outdated and incomplete consisting only 5,574 messages with 13% Spam and 87% Ham (or legitimate) messages, it is not suitable for detecting the most recent, stealthy SMS Spam even using advanced machine or DL techniques. We propose data collection methods to crawl 55,686 SMS messages from Spam observatories such as ScamWatch\(^3\) and Action Fraud\(^4\). We labelled the dataset into Spam and non-Spam messages constituting 15,209 (27.31 %) and 40,477 (72.69 %) SMSes, respectively.

**Valuation of machine learning models.** Usually, supervised text classification methods (like binary classification) or DL have been used to identify SMS Spam. However, given the imbalance nature of the SMS Spam datasets, we propose one-class learning (unsupervised learning) and positive unlabeled (PU) learning (semi-supervised learning) algorithms alongside other ML models. To the best of our knowledge, the notion of one-class learning and

---
\(^1\)https://www.fcc.gov/covid-19-text-scams  
\(^2\)https://www.scamwatch.gov.au/scam-statistics  
\(^3\)https://www.scamwatch.gov.au/  
\(^4\)https://www.actionfraud.police.uk/
PU learning for SMS Spam detection was never explored earlier. We evaluate and compare the performance of PU learning and one-class learning models with both traditional two class ML as well as with the state of the art neural network and DL such as the latest transformers based architecture [15]. Furthermore, we analyse and compared the performance of several well-known ML methods in order to establish a good baseline for future comparisons. Additionally, we evaluated the ML models over the new dataset with different set of features/word embeddings starting from Non-semantic Count based Vector space model (Count Vectorization, TF-IDF) to semantic Non Context-Based Vector Space Model (Word2Vec, fastText, GloVe) with static and dynamic modes and with semantic Context Based Vector Space Models (BERT, ELMo etc).

Robustness analysis of Spam models in the face of new challenges. To evade detection, Spammers may leverage obfuscation or perturbation methods [16] to Smish [17] or Punycode [18] Spam SMS texts and embedded URLs. We evaluate the robustness of ML models against 4 levels of adversarial perturbations: character-level, word-level, sentence-level, and multi-level attacks. To the best of our knowledge, such extensive adversarial assessment of the ML models on Spam SMS have not been performed before.

2 RESEARCH CHALLENGES AND RELATED WORK
In the following, we discuss challenges and overview state-of-the-art SMS Spam filtering and detection methods in literature.

2.1 Challenges
Availability of data. The lack of updated, genuine, and publicly large SMS Spam dataset is a major concern. The SMS Spam filtering has a lack of good dataset sources and a dearth of diversity in the data used for evaluation of detection systems. Recency of the data is another challenge. The datasets used by researchers in the literature are extremely outdated and raises questions on its suitability in the present SMS scam landscapes due to non-representative of latest attacks (see table 1). Other concern is the message ambiguity due to short length, limited header information, presence of emojis and abbreviations in their text, therefore, established email Spam filters may have their performance seriously degraded when directly employed to dealing with mobile Spam.

Lack of benchmark. Different methods have been proposed by researchers to study SMS Spam detection and adversarial attacks in texts, but there is no benchmark [19, 20]. Moreover, different datasets have been used by researchers in their work, making it difficult to compare these methods. Meanwhile, it also affects the selection of evaluation metrics. There is no exact statement about which metric measure is better in a situation and why it is more useful than others.

Robustness against new challenges and attacks. Another challenge in the Spam detection is the dynamic behavior of spammers. They always try to find a way to deceive the Spam filters using different adversarial attacks. Moreover, due to the convergence of mobile network with internet, Spammers may likely adapt the techniques used for tricking users on internet.

Lack of strategy and collaboration One of the less highlighted issue in effective tackling of Spam SMS is the lack of collaborations between researchers, end users, network providers, and industry as well as absence of a robust strategy for dealing with threats to the security of mobile users and Spam SMS detection.

2.2 Related Work
To effectively handle the threat posed by SMS Spam, several techniques have been proposed in the academic literature and white papers in the industry. However, none of these studies provides a complete picture of the SMS Spam problem, though there are resources that handle part of the problem or try to reduce the problem to a single dimension. Importantly, there is no previous systematic empirical analysis of SMS scam detection systems.

Abdulhamid et al.[19] presented a review of the currently available methods, challenges, and future research directions on SMS Spam filtering and detection techniques. Wang et al [21] conducted an empirical survey on word embeddings. Buchanan and Grant [1] offered a brief description of Nigerian scam schemes, emphasizing that the growth of the Internet has aided the proliferation of cyber-crime. To classify Spam texts, Almeida et al. [5] used the SVM classifier. They used word frequency as a feature and discovered that SVM performed very well. Another study [22] used SVM classifiers and k-Nearest Neighbor (kNN) to classify Spam text messages. The experimental results confirmed that using a combination of BoW features and structural features to classify Spam messages performed better. Some researchers have also proposed evolutionary methods for SMS Spam detection by assimilating the byte-level features of SMS [23]. Researchers have recently started to use deep neural networks. To filter SMS Spam, Popovac et al. [24] proposed a CNN-based architecture with one layer of convolution and pooling and achieved 98.4% accuracy rate. Jain et al. [25] used Long short-term memory (LSTM) model, for SMS Spam filtering. Their model attained a 99.01% accuracy with the help of 6000 features and 200 LSTM nodes. They also employed various word embedding techniques including ConcepNet, WordNet and Word2Vec. [26] proposed several deep neural network-based model and attained an accuracy of 98.51%.

Despite the preceding algorithms’ success in identifying Spam messages, the existence of adversaries significantly degrades Spam filters’ performance [27]. Graham-Cumming presented an approach against an individual user’s Spam filter in a talk at the 2004 MIT Spam Conference [28]. Random words were added into Spam mailings in this attack. A prior paper [3] attempted to broaden the scope of this attack by substituting regularly used English words with random phrases. After that, many adversarial attacks and countermeasures have been documented in a range of applications. Attacks on text classification jobs are typically carried out by altering characteristics or changing the text sequence’s content [20]. Several strategies have been identified in the field of adversarial attacks on Spam filters [4, 29]: injection of Ham words, obfuscation of Spam words, poisoning, alteration of labels, and synonym replacement. The impact of dictionary-based attacks and well-informed concentrated attacks, on the other hand, can be mitigated by using classifier weights [30].
In many ways, our research is different from that of our predecessors. No previous work, to our knowledge, has attempted to systematize and experimentally evaluate the SMS Spam literature from the perspective of security challenges.

3 INTRODUCING THE SUPER SMS SPAM CORPUS

The Super SMS Spam corpus is a dataset of labeled SMS messages (reported over the last decade) that we collected for Spam research from the public and free for research sources. In any scientific study, having the most up-to-date, reliable, and representative data is critical. The most widely used SMS datasets including NUS SMS Corpus5 and UCI SMS Spam Collection8 used for Spam research are extremely outdated. Due to the exponential growth of online services and ongoing COVID19 epidemic, the threat landscape has significantly evolved in recent years, necessitating an updated dataset covering all the latest scams.

3.1 Data Collection

A comprehensive survey was conducted in order to identify, collect, and aggregate all of the public SMS datasets. Resultantly, we consolidated a corpus of 153,551 SMS instances from public and free-for-research sources. Details of different SMS datasets aggregated in the consolidated dataset are given in Table 1. Importantly, we manually crawled images of latest scam messages publicly shared on Twitter and those reported to the scam observatories including Scamwatch (Australian Competition and Consumer Commission’s website for public education against scams) and Action Fraud (UK’s national reporting center for fraud and cyber-crime)7 to cover the recent landscape of SMS scams (categories and campaigns [31]). This way, a list of 71 scam messages from Twitter and 141 Spam SMS messages from scam observatories were added to the corpus.

3.2 Data Augmentation

All SMS messages in the consolidated dataset and newly gathered Spam messages from various volunteers and observatories were manually labelled using a set of carefully designed rules (list and details regarding derivation of rules are given in Appendix A.1). We eliminated discrepancies (duplicate and non-English messages) in the consolidated dataset and refined the data into a format suitable for further analysis by performing a series of processes. To this end, we first filter out SMS messages in non-English languages. For this purpose, in particular, we use a two pass filtering mechanism in order to filter the large amount of non-English SMS messages from the consolidated dataset. In the first pass, we use langdetect (an open source Python library for language detection) [32] to determine the language of each SMS and filter out SMS in non-English languages. We then passed the filtered SMS messages returned by langdetect to Google API for language detection and translation) [33] to further filter out the non-English SMS messages. Google API was kept in the second phase of filtering due to the limitation of API calls for free user.

Table 1: Overview of SMS Spam datasets consolidated to generate an augmented dataset.

| Dataset          | # of SMSes | Language | Labeling | Year |
|------------------|------------|----------|----------|------|
| UCI [5, 6]       | 5,574      | English  | Labelled | 2012 |
| NUS [34]         | 67,063     | English, Chinese | Unlabelled | 2015 |
| Github1 [35]     | 77,039     | English, Roman, Hindi | Unlabelled | 2019 |
| Github2 [36]     | 557        | English  | Labelled | 2018 |
| Gupta [7]        | 3,318      | English, Roman, Hindi | Labelled | 2018 |
| **Consolidated** | 153,551    | Multi    | Partial  | -    |
| **Consolidated [Augmented]** | 52,814 | English | Partial | -    |

4 ANALYSIS METHODOLOGY

To identify the state-of-the-art in preventing SMS scams, we gathered existing techniques and counter measures from academic, industry, internet domain, and systematically categorize them. Figure 1 depicts our experiment methodology to perform comparative analysis of various feature models and machine learning techniques.

4.1 Data Split

Splitting the dataset is essential to build a reliable ML model and for an unbiased evaluation of classification performance. Common techniques for training and testing the data involve k-fold cross validation or splitting the original dataset into training (usually 80%-70%) and testing (usually 20%-30%) data. In this study, we divided the data set into three subsets: train (80%), test (20%), and hold-out. The hold-out split was created by randomly selecting 225 Spam SMS messages from all over the dataset. The train set was used to fit the ML model, whereas the test data set was used to estimate the performance of the model on data not used to train the model.

---

5https://github.com/kite1988/nus-sms-corpus
6https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection
7https://www.actionfraud.police.uk/
The hold-out split was created for validation of Spam detection on diverse instances and adversarial evaluation (further elaborated in Section 4.3.6). In order to split the data set into train and test splits, we utilised the train_test_split() method of the scikit-learn [37] library. Moreover, we removed the unnecessary variables such as punctuation, special symbols, and stop words from the dataset with the help of Python NLP library NLTK [38].

4.2 Feature Extraction

The unstructured text sequences in the SMS dataset need to be converted into a structured feature space before applying ML models. Therefore, we converted the list of words in each message into a feature vector. We generated both syntactic and semantic feature vectors of all the messages in the dataset to compare the impact of different feature types on the accuracy of classifiers. A taxonomy of text feature extraction techniques applied in this study is illustrated in Fig. 2.

4.2.1 Syntactic / Non-semantic Count based Vector space model. We applied non-semantic techniques like bag of words (BoW) [39] and n-grams [40] with TF-IDF [41] to convert raw text to numerical features. BoW captures word frequency in the SMS corpus, however, this model is sparse and doesn’t consider word order. As a result, we employed N-grams, which are a document’s sequence of N-words to count the word pairs (bigrams, trigrams etc.). The issue with N-gram is that even here, the resultant matrix is exceedingly sparse. Finally, we transformed the entire Bow and N-grams corpus into TF-IDF (term frequency-inverse document frequency) corpus. TF-IDF measures a word’s importance in a document (SMS) and the corpus. Words that are uncommon in a document will have a high TF-IDF score. These mappings capture lexical features while ignoring semantics. We implemented this task with the help of the Python library scikit-learn.

4.2.2 Semantic/ Word Embedding. Even while we have syntactic word representations, this does not imply that the model accurately represents the semantic meaning of the words. This constraint makes it difficult to interpret words within the model. To tackle this challenge, several academics focused on word embedding [42]. Word embedding is a feature learning approach in which each word or phrase in the vocabulary is mapped to an N-dimensional vector of real values in order to capture word semantics and context. Therefore, we created semantic feature vector for each of the following.

4.2.3 Context-Independent Vector Space Model. To convert unigrams into intelligible input for ML systems, many word embedding approaches have been developed. In this section, we focused on the following popular classic word embeddings that have been and successfully implemented for text classification [43].

- **Word2Vec** uses a neural network to learn word associations from large corpora and generate a vector representation for words or tokens and captures semantic similarity between words [44]. Word2Vec creates a vector space with each unique word in the corpus, often with several hundred dimensions, such that words with similar contexts in the corpus are close to one another in the space. Moreover, the vectors for unknown words are randomly initialized using a generic normal distribution. In our study, we leverage Gensim library [45] to implement both static and dynamic models of Word2Vec. With the static approach, the Word2Vec model uses pre-trained word embedding with 300-dimensional vectors. The vectors are kept static during training. Whereas, in the dynamic approach, the dimensional vectors are modified (fine tune the embeddings) during training phase.

- **GloVe** [46] In comparison to Word2Vec that utilises a window to establish local context, GloVe uses data from the whole text corpus to create an explicit word-context leveraging words’ co-occurrence matrices. Each word in this method is represented by a high-dimensional vector and trained using the surrounding words throughout a large corpus. In order to achieve GloVe embeddings, we utilised the pre-trained 100 dimensional GloVe model in Python, published by the Stanford NLP Group 9.

- **fastText** is an extension of Word2Vec with improved efficiency and effectiveness [47]. It addresses the out-of-vocabulary issues associated with Word2Vec feature model. In this work, we use Python’s library fastText10 for generating embeddings for our analysed machine learning models.

4.2.4 Context-Dependent Vector Space Model. The context-independent vector space model provides context-free or static embeddings, which are incapable of capturing polysemy. Second, we only have one representation for a word, despite the fact that words have diverse features such as semantics, grammatical behavior, and register/connotations. To address the aforementioned issues, the “Contextualized Word Representations” technique was introduced, which aims to capture word semantics in various settings in order to solve the problem of polysemous words and their context-dependence [21]. Here, we focused on the following contextualised word embeddings.

- **Bidirectional Encoder Representations from Transformers (BERT)** [48] BERT is a powerful transformer-based architecture that has demonstrated state-of-the-art performance on different NLP tasks [15]. Unlike embedding representations of individual words (such as Word2Vec, GloVe, or fastText), BERT is backed by Transformer and it’s core principle - attention, which understands

---

9[http://nlp.stanford.edu/data/glove.6B.zip](http://nlp.stanford.edu/data/glove.6B.zip)

10[https://github.com/facebookresearch/fastText](https://github.com/facebookresearch/fastText)
the contextual relationship between different words. In this study, we generated the BERT embeddings by implementing a pre-trained BERT model using the SimpleTransformers [49] package in Python (further details in Appendix A.3).

Embedding from Language Model (ELMo) [50] ELMo is a bidirectional Language Model (biLM) that extracts multi-layered word embeddings. ELMo word vectors address Polysemy, where a word has several meanings. These word vectors are pre-trained, deep bidirectional language model (biLM) functions. They can be easily integrated into current models and progress complex NLP tasks like question answering, textual entailment, and sentiment analysis. In our implementation, we accessed ELMo via TensorFlow Hub (a library / repository of trained ML models that enables transfer learning)\(^\text{31}\) using Python.

4.3 Classification Techniques

Choosing the right classifier/ architecture for model building/ training is one of the most critical steps in the Spam detection pipeline. However, researchers have difficulties in defining acceptable structures, architectures, and methodologies for text classification. In this section, a brief overview of different classifiers and architecture implemented in this study are discussed. Figure 3 depicts various machine learning based approaches analysed in this work.

Given that there is no solid theory on how to map algorithms onto problem types; instead, a practitioner should research the literature work for the specific problem and conduct controlled trials to determine which algorithm and algorithm configuration performs best for a certain classification job. At this stage, we designed a first set of experiments to evaluate how well the most popular techniques of text classification would perform on the original messages. For this purpose, we developed several models using different ML approaches. The approaches considered in this study are stated below.

4.3.1 Two-class Classification. We started with the conventional methods of text classification such as two-class classification or binary classification [52]. Some algorithms are primarily built for binary classification and do not natively handle more than two classes; for example, Support Vector Machines (SVM) [53]. SVM have shown to be one of the most powerful and efficient state-of-the-art classification algorithms for handling the Spam problem [54]. They’re supervised learning models that examine data and find patterns that may be used to categorize and investigate the relationship between variables of interest. SVM outperforms many classifiers on Spam detection by employing content-based filtering to find particular features in the text [55]. In the two-class classification approach designed in this study, we implemented a binary SVM using Scikit-learn.

4.3.2 One Class Classification. Next, we covered the one-class classification (OCC) [56] approach which includes methods for detecting outliers and anomalies. The one-class algorithms are recognition-based, with the goal of recognizing data from one class and rejecting data from all others.

In this approach, we employed One Class Support Vector Machine (OCSVM) [57], which is a natural extension of SVMs. The OCSVM is an unsupervised technique for learning a decision function for novelty detection, which involves categorising fresh data as similar or dissimilar to the training set. Generally, an OCSVM or one class learning is suitable for imbalanced dataset (similar to our dataset) because only the majority class is needed to train a given ML model. Moreover, such an approach is useful when it is hard to create outliers. In many practical scenarios, it is hard and expensive to create anomalies, certainly the case for fraud detection which happens rarely. Also, if you have a new type of fraud that has never been encountered before, it would be easy for OCSVM to detect it, whereas a traditional classification model may struggle to do so [58].

4.3.3 Positive and Unlabeled (PU) Learning. After the supervised (two class SVM) and unsupervised (OCSVM) approaches, we next considered positive and unlabeled data (PU) learning [59], which is a semi-supervised approach that learns from the positive cases in the data and uses them to predict the unknown cases. There are two primary methods for implementing PU learning. These include PU bagging and a two-step process. The PU bagging is basically an ensemble of ensembles; training many ensemble classifiers in parallel. Each ensemble balances the classes according to the size of the positive class. On the other hand, the two-step method is a more complicated way to learn about PU. It uses ML techniques to change the labels of data while training and takes longer time to train. Therefore, we decided to opt the PU bagging approach.

\(^\text{31}\)https://www.tensorflow.org/hub
Adversarial Attacks

Adversarial Attack
Knowledge based
White box
Black box
Character-level Charswap
Word-level
EDA
Homograph
Spacing
Sentence-level Paraphrasing
Multi-level
Paraphrasing
Hybrid
Targeted based
Targeted
Non-targeted

4.3.4 Neural Networks (NNs). Most of the state-of-the-art text classification is based on neural networks [52]. As discussed earlier in Sec 4.2.3, Facebook recently released a lightweight shallow neural network library (fastText) for unsupervised (embedding) representations of words and supervised text classification [61]. fastText employs a hierarchical classifier that decreases the training and testing time complexity of text classifiers from linear to logarithmic in terms of the number of classes. It also exploits the fact that classes are imbalanced (some classes appearing more often than other, similar to our Spam dataset) by using the Huffman algorithm [62] leading to further computational efficiency.

4.3.5 Deep Learning. Lately, various DL architectures have emerged and outperformed shallow ML in NLP, image classification, etc [63]. Unlike shallow ML, DL can create models directly on high-dimensional raw data using automated feature learning. Therefore, we finally considered several state-of-the-art DL models that have been successfully tested for text or Spam classification by researchers. Depending on the learning task, DL offers various architectural variants such as recurrent neural networks (RNNs) [64], convolutional neural networks (CNNs) [65], transformer [66], etc. Architectural variations are distinguished primarily by the layers, neuronal units, and connections employed. Below, we describe each DL model used for Spam detection in this study.

Recurrent Neural Network (RNN): RNN [64] is one of the most utilized and successful DL architectures for text classification. RNNs are typically used to classify text using LSTM or GRU [52]. In this work, we implemented the three widely used RNN based models such as LSTM [67], BiLSTM [68] and BiGRU [69] (brief introduction of each in Appendix A.3) for Spam detection using Keras [70] and TensorFlow2 [71] in Python.

Convolutional Neural Network (CNN) Despite several advantages, RNN-based designs might be biased when later words are more significant. CNN models were developed to address this bias by utilising a max-pooling layer to identify discriminative words in text input [65]. Moreover, CNNs can help us with parallelization, local dependencies and distance between positions. Consequently, we implemented a 1D CNN for the Spam detection since it has demonstrated to perform effectively on text categorization with comparatively small parameter tuning [72]. Moreover, we employed a generic temporal convolutional network (TCN) [73], which is a dilated-causal variant of the CNN and a viable alternative to recurrent architectures since it is capable of dealing with extended input sequences without experiencing vanishing gradient problems. Additionally, we constructed a model based on ensemble learning by merging a 1D CNN with a single Bidirectional GRU (BiGRU).

One of the reason for choosing BiGRU is that it performs effectively with temporal data because it incorporates both prior and subsequent information in the sequence. In this study, we implemented all of the convolutional architectures using Keras and TensorFlow2 in Python.

Transformers CNNs are successful with shorter phrases, but gathering word associations in longer sentences is tedious and impractical. In 2017, Google introduced Transformers to deal with the long-range dependency challenge [15]. Transformer avoids recursion via the use of attention mechanisms and positional embeddings to comprehend phrases in their entirety. In 2018, BERT was open-sourced by Google 4.2.4. With just one extra output layer, the pre-trained BERT model may be fine-tuned to build state-of-the-art classifiers for a variety of tasks, including text classification, without requiring significant task-specific architectural adjustments [48]. Recently, modified BERT models have been proposed by researchers to enhance BERT’s prediction metrics or computational performance, and these models have been demonstrated to attain state-of-the-art results to counter email phishing [74] and applied to filter multilingual Spam messages [75] with promising results. In this study, in addition to the original BERT, we employed some of the modified models proposed by researchers that outperform BERT in many tasks. These models include DistilBERT [76], RoBERTa [77], and Cross-lingual Language Model (XLM) [78] (brief introduction of each is given in Appendix A.3). We implemented all of the BERT-based models using the SimpleTransformers package in Python.

4.3.6 Adversarial Attacks. Effective adversarial attacks depend on the attacker’s knowledge of the targeted ML model. Attacks can be classified as black-box or white-box based on target model information and targeted or non-targeted based on whether the erroneous output is desired [11]. In a white-box scenario, an attacker can access target models and training data and generate perfect adversarial instances. Black-box attacks are used in situations when the attacker has limited knowledge of the target model [79]. An attacker must repeatedly test the model with different inputs to find a flaw. Adversarial attacks in the text domain are further classified as char-level, word-level, sentence-level, and multi-level based on the perturbation units used to generate adversarial samples to produce hostile samples that trick detectors. Multi-level attacks combine many perturbation attacks to produce an unnoticeable and high-successful attack.

Figure 4: A taxonomy of eight different adversarial attacks conducted in this study. The analysed attacks are grouped into perturbations onto character, words, sentence, and miscellaneous, are shown in the figure.

In order to assess the robustness of ML models, we reviewed the literature on evasion tactics in text categorisation and Spam detection. Currently, Spam detection models are examined using conventional blackbox adversarial mechanisms [8]. However, due
to the convergence of mobile networks and the internet, scammers may likely adapt the techniques used for tricking internet users to deceive mobile users as they go online. For instance, Smishing is an SMS Spam campaign in which scammers send an enticing text message to deceive receivers into opening a link and transferring the victim private information or downloading malware [80]. Consequently, it is necessary to proactively seek such approaches.

Punycode attack is a homograph technique used in URL phishing [81] [82]. The attacker creates a homograph URL by choosing similar letters from other languages, especially Cyrillic or non-Latin. To the human eye, these Cyrillic glyphs or other similar non-Latin characters can easily be confused with their Latin counterparts. Computers, however, interpret it differently, since different hex codes are assigned to them. This technique may not only be used in Smishing but other Spam SMS campaigns by encoding the top Spam keywords in order to bypass the Spam filters. The homograph technique like Punycode haven’t been explored for SMS Spam problem earlier. Figure 4 presents taxonomy of adversarial attacks performed in this study. We created the adversarial texts with the following principle in mind: they must retain the same semantic meaning as the originals. Moreover, we developed a thesaurus of the 200 most frequent keywords from the Spam messages in the dataset with the help of the Python library WordCloud [83].

In the following, we elaborate on the adversarial attacks employed in this study:

- **Paraphrasing** A popular attack for fooling Spam filters by replacing vocabularies with synonyms or similar phrases, rendering the message unrecognizable to the detection model. We carried out this attack in 2 folds. In the first fold, we used a commercial tool Quillbot12 to restructure the sentences. In the second fold, we replaced all of the Spam keywords from thesaurus with its synonyms in each SMS message.

- **Easy Data Augmentation (EDA)** This tactic bypasses Spam filters by randomly removing, swapping, replacing, or adding a word’s synonym in a sentence. We carried out this attack with the help of TextAttack [16].

- **Homograph** This attack involves replacing thesaurus’s keywords found in SMS with its punycode, by generating homoglyphs for each character in the corresponding Spam word. We generated the homoglyphs with the help of an online homoglyph attack generator 13 tool.

- **Spacing** In this technique, spaces are added between the characters of Spam keywords to evade detection. We designed this attack manually by adding spaces between the characters of the thesaurus’s keywords found in SMS.

- **Charswap** This type of attack randomly substitute, delete, insert, and swap adjacent characters of Spam keywords in the message. We utilise the TextAttack framework to generate adversarial examples of this type.

- **Hybrid** This type of attack involves combination of different attacks. We designed this attack using a combination of spacing technique and replacement of visually same characters in Spam words.

### 4.4 Evaluation Metrics

For the evaluation of ML models, multiple aspects have to be taken into account, such as performance, computational resources, interpretability and robustness. Performance-based metrics quantify how effectively a model achieves the learning task’s purpose. In this work, we evaluated the models using the metrics of Precision (PR), Recall (RE), Accuracy (ACC) and F1-score (F1) [84]. The latter is considered the most appropriate metric for this particular task, considering that the dataset is highly unbalanced. Formally, these metrics are defined by the following equations:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

\[
F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{2 \times TP + FP + FN}
\]

Here, we consider Spam SMSes as positives while non-Spam (i.e., Ham) ones as negatives. We represent the total number of Spams and Ham SMSes by \(P\) and \(N\), respectively. Whereas, in the above equations, the TP represents the number of true positives (i.e., correctly classified as Spam) and FP shows the number of false positives (i.e., wrongly classified as Spam). Similarly, we denote true negatives and false negatives by, TN and FN, respectively. TN shows the number of SMSes correctly classified as Ham while FN represents the number of SMSes that are wrongly classified Ham.

### 5 COMPARATIVE EVALUATION

In this section, we evaluated the efficacy/competence of the models in two steps. The first set of experiments approximated the performance of the models in order to evaluate their efficacy in classifying Spam and legitimate messages, while the second set of experiments assessed the robustness of the models.

#### 5.1 Results Using The Traditional ML Approach

First, the ML models were evaluated without adversarial examples. Initially, we encoded the messages in the Train and Test splits with different representation models (ref para 4.2) to obtain feature vectors which were then fed to the classification algorithms. Each classifier was trained with the encoded vectors of the Train split along with their respective labels; once trained, their performance was evaluated on the Test split.

We started the experimentation with traditional two-class SVM (TCSVM). We trained several models of TCSVM with different syntactic and semantic features including BoW, n-grams, Word2Vec and GloVe, extracted from the train split of the super dataset. We then used the trained models to classify the instances of the test set. Next, we fed the same set of features to the OCSVM and PU classifiers. However, for OCSVM and PU, we use only Spam messages of the train split since these two classifiers only require samples from the positive class. The trained models are then evaluated with the messages of the test split for training. The performance evaluation
results of the aforementioned ML models are summarized in Table 3. The performance metrics are reported and grouped by the encoding/feature model and classification algorithms.

Roughly speaking, the results varied widely depending on the classification algorithm and feature model. With all the feature models including TFIDF, Word2Vec and GloVe, the OCSVM performance remains significantly lower than other models, both in terms of f1-score and accuracy. On the other side, good performance was demonstrated by most of the TCSVM, PU and fastText models, however, TCSVM with Word2Vec produced the highest among all the shallow ML models with 83% and 95.5% f1-score and accuracy respectively. It is pertinent to mention that to all of the classifiers demonstrated good performance with BoW. With the exception of OCSVM, the models perform better with classic word embedding techniques like Word2Vec and GloVe, producing higher f1-score in comparison to other models. The PU and TCSVM with Word2Vec and GloVe embeddings outperforms all other models including OCSVM and fastText. Lastly, the models with n-grams in this experiment cannot do much despite trying different “n” grams and having so many hyper parameter tuning.

5.2 Results Using The DL Approach

Next, we evaluated the state-of-the-art DL models (discussed in section 4.3.5), proposed by academia/researchers in top-tier conference proceedings, workshops, symposiums, journals, and pre-prints published on arXiv. Like in shallow ML evaluation, the DL models were first trained on the training split of the super dataset with different embeddings/representation models and then evaluated on the test split. Some of the DL classifiers are also trained with random word embedding (WE-Random). For WE-Random, we simply define a randomly initialised trainable embedding layer. Thus, it is not pre-trained but learned from scratch with the rest of the downstream model parameters and corrected during training. Table 4 shows performance evaluation of DL models.

In comparison to traditional ML models, the variability in the accuracy of the DL models is less noticeable. All of the classifiers obtained an accuracy greater than 90%, with the Ensemble model of CNN-BiGRU with Word2Vec(static) achieving the highest accuracy (98.4%), slightly higher than LSTM and BiLSTM (98.3% & 98.2% respectively). On contrary, the variability in the f1-score of the models is quite high. The RoBERTa models produced the lowest f1-score whereas LSTM, BiLSTM, CNN, and TCN achieved a high score of 97% each.

From the high accuracy of shallow ML and deep ML models in Table 3 and 4, it is evident that all of the models produce impressive results for the Ham (legitimate messages) class, whereas for the Spam class the results are very poor in most of the models, as shown by the f1-score. Out of a total of sixteen shallow ML models, only two models (TCSVM with Word2Vec and GloVe) achieved greater than or equal to 80% f1-score (83% being the highest achieved by TCSVM with Word2Vec). Similarly, only four deep ML models (LSTM, BiLSTM, CNN with Random and GloVe embeddings) out of fifteen models crossed the 90% mark.

Table 3: Performance evaluation of shallow ML classifiers with the Super dataset (cf. Table 2). Here PR, RE, FS, ACC, and CM represent precision, recall, F1-score, accuracy, and confusion matrix, respectively.

| Feature Model | Classifier | PR | RE | F1 | ACC | CM (TP FN FP TN) |
|---------------|------------|----|----|----|-----|------------------|
| BoW (TF-IDF)  | TCSVM      | 90%| 59%| 71%| 93.3% | (9974 48 352 1113) |
|               | OCSVM      | 53%| 40%| 45%| 86.3% | (9974 48 352 1113) |
|               | PU         | 87%| 71%| 78%| 94.4% | (9974 48 352 1113) |
| Bigram (TF-IDF) | TCSVM    | 94%| 24%| 38%| 89.0% | (10044 120 152 148) |
|               | OCSVM      | 7% | 35%| 12%| 27.1% | (10044 120 152 148) |
|               | PU         | 90%| 30%| 45%| 89.6% | (10044 120 152 148) |
| Trigram (TF-IDF) | TCSVM    | 94%| 7% | 12%| 86.6% | (10044 120 152 148) |
|               | OCSVM      | 12%| 80%| 21%| 12.4% | (10044 120 152 148) |
|               | PU         | 99%| 9% | 16%| 87.0% | (10044 120 152 148) |
| Word2Vec      | TCSVM      | 89%| 78%| 83%| 95.5% | (9974 48 352 1113) |
|               | OCSVM      | 17%| 94%| 29%| 34.4% | (9974 48 352 1113) |
|               | PU         | 74%| 84%| 79%| 93.5% | (9974 48 352 1113) |
| GloVe         | TCSVM      | 86%| 77%| 81%| 94.9% | (9974 48 352 1113) |
|               | OCSVM      | -  | -  | -  | 27.8% | (9974 48 352 1113) |
|               | WE-Random  | 47%| 74%| 65%| 87.5% | (9974 48 352 1113) |
| fastText      | fastText   | 92%| 54%| 68%| 92.8% | (9974 48 352 1113) |

Table 4: Performance evaluation of Deep ML classifiers with Super Dataset (cf. Table 2).

| Classifier | Embedding | F1 | ACC | CM (TP FN FP TN) |
|------------|-----------|----|-----|------------------|
| BERT       | bert-base-uncased | 79%| 94.9% | (10056 69 10068 10044) |
| ELMo       | ELMG      | 80%| 95%  | (10056 69 10068 10044) |
| RobERTa    | roberta-base | 53%| 90.0% | (10056 69 10068 10044) |
| XLM-RoBERTa| xlm-roberta-base | 52%| 90.5% | (10056 69 10068 10044) |
| DistilBERT | distilbert-base-uncased | 78%| 94.8% | (10056 69 10068 10044) |
| LSTM       | WE-Random  | 97%| 98.3% | (10056 69 10068 10044) |
| BiLSTM     | WE-Random  | 97%| 98.2% | (10056 69 10068 10044) |
| CNN        | WE-Random  | 97%| 98.3% | (10056 69 10068 10044) |
|            | GloVe (static) | 97%| 98.2% | (10056 69 10068 10044) |
| TCN        | WE-Random  | 80%| 97.6% | (10056 69 10068 10044) |
|            | Word2Vec (static) | 76%| 97.3% | (10056 69 10068 10044) |
|            | Word2Vec (dynamic) | 69%| 94.1% | (10056 69 10068 10044) |
| Ensemble   | WE-Random  | 80%| 95.8% | (10056 69 10068 10044) |
|            | Word2Vec (static) | 86%| 95.4% | (10056 69 10068 10044) |
|            | Word2Vec (dynamic) | 80%| 95.7% | (10056 69 10068 10044) |
Table 5: Instances of Adversarial Attacks (Spam messages) colored red.

| Attack   | SMS Text                                                                 |
|----------|--------------------------------------------------------------------------|
| Actual SMS | You may get a $750 Economic Support Payment. For more details, Click https://xxx.info/covid/ |
| Paraphrasing | A $750 Economic Support Compensation may be available to you. For further details https://xxxx.info/covid/ |
| Charswap | You may get a $750 Economic Support Payment. For further details https://XXXXXXX.info/COVID/?r=xxxxxx |
| EDA      | You may get a $750 Economic Support Payment                                |
| Homograph | You may get a $750 Economic Support Payment                                |
| Spacing  | You may get a $750 Economic Support p a y m e n t. For more details, C l i c k https://xxx.info/covid/ |
| Hybrid   | You may get a $750 Economic Support p a y m e n t. For more details, C l i c k https://xxx.info/covid/ |

6 ADVERSARIAL EVALUATION: ROBUSTNESS ASSESSMENT OF ML MODELS

The second set of experiments focused on quantitatively evaluating the influence of adversarial examples on our trained models. In order to assess the robustness of our trained models, we employed six different sets of black-box attacks discussed previously (ref para 4.3.6). To do this, we used the Hold-out split, together with the thesaurus, to carry out the attacks. Each attack attempted to substitute all Spam words in the message that were found/matched with the thesaurus (see examples in Table 5). Once the attack was completed, the modified messages were encoded with different representation models to obtain the corresponding feature vectors, and then fed to the previously trained models in order to evaluate their performance with the same metrics. To begin, we first passed the actual instances of the Holdout split to the trained models in order to estimate their performance on positive samples and get a baseline result. Following that, the adversarial examples were passed to the model to assess its robustness. The results of the adversarial evaluation of shallow ML and deep ML models are presented in Table 6 and Table 7, respectively.

For shallow ML models, the spacing tactic remains the most effective technique, evading eight models the most. The spacing tactic was most effective in evading all classifiers trained using BoW/TF-IDF (TCSVM, OCSVM, PU), GloVe (TCSVM, OCSVM, PU), and fastText as well as the TCSVM-Word2Vec model, which demonstrated the highest performance in the previous section. Character swap, on the other hand, was the second most successful attack against the six models presented in Table 6, completely evading three (Trigram/TCSVM, Trigram/PU, Word2Vec/PU), while the Bi- gram/TCSVM and Bigram/PU models survived to achieve a little score. Moreover, the Homograph/Punicode attacks also proved to be quite efficient against all the models except those trained with BoW.

In comparison to shallow ML models, the deep ML models showed some robustness to different adversarial techniques. However, their performance degraded significantly in the presence of the spacing technique. The spacing evasion technique proved to be the most successful attack against thirteen models out of the total fifteen models presented in Table 7. The hybrid attack proved to be the second most successful attack, beating the spacing tactic on CNN-BiGRU Ensemble models trained with WE-Random and Word2Vec (dynamic) embeddings. A bird’s-eye view of the adversarial evaluation is illustrated in Appendix A.2.

To summarise, the spacing tactic proved to be the most devastating attack for both shallow and deep ML models, whereas EDA remained the least effective attack since all of the shallow ML and DL models show high robustness to this attack. The experiment shows the comparatively high robustness of word embedding to different adversarial attacks, especially paraphrasing and EDA, in making a better classification. Moreover, in comparison to shallow ML, the deep ML models demonstrated fair robustness to several adversarial tactics. In general, the results support the premise of the usefulness of contextual embedding and deep ML to resist different types of attacks. Moreover, it can be concluded that context plays a vital role in the classification and outcome of Spam detection as well as the robustness of the models. The classification accuracy of the models during adversarial evaluation with contextual embedding and deep ML remains comparatively high in comparison to the models trained with syntactic feature models and shallow ML. Furthermore, compared to deep ML, the classification performance of the ML models experienced a significant decrease in accuracy across most adversarial attacks. This indicates that ML models are more sensitive to adversarial attacks. Conversely, the classification performance of the deep ML models, especially the transformer-based architecture demonstrates good resistance to different sorts of adversarial attacks while classifying Spam messages.

7 DISCUSSION & FUTURE RESEARCH DIRECTION

In this section, we summarise the key observations that we found while comprehensively evaluating the ML models concerning performance and robustness.

Spam detection. In the first set of experiments, we trained thirty-one ML models to evaluate and identify suitable models for SMS Spam detection. While the majority of shallow ML models provide excellent accuracy, just one model achieved a decent f1-score, indicating that the shallow ML models have a poor Spam detection rate. In contrast, a few deep ML models achieve acceptable f1-scores when compared to shallow ML models. This demonstrates the superiority of deep ML over shallow ML in SMS Spam classification.

Spam detection with few positive samples (small training set problem). The PU learning method learns from the positive cases in the data and applies what it has learned to filter Spam messages. The PU technique outperforms OCSVM in all feature models and gets a high f1-score and accuracy in both performance and adversarial assessment, which is similar to TCSVM and fastText. This makes it a good alternative when only positive samples and small datasets are available.
As such, it is surprising that imbalanced classification in Spam SMS assessment for robustness using different adversarial tactics. Overall, designed for this purpose, and the previously trained models were adversarial samples. Therefore, a second set of experiments was get shallow ML and deep ML-based SMS Spam models by generating research was to explore how adversarial attacks can be used to tar-

### Table 6: Robustness/Adversarial Evaluation of Shallow ML Classifiers on The Holdout Split

| Feature Model | Classifier | Actual ACC | Paraphrasing ACC | EDA ACC | Homograph - ACC | Spacing - ACC | Charswap - ACC | Hybrid - ACC |
|---------------|------------|------------|------------------|--------|----------------|--------------|--------------|-------------|
| Bow (TF-IDF)  | TCSVM      | 80.9%      | 51.1%            | 79.1%  | 88%            | -            | 28.4%        | 35.6%       |
|               | OCSVM      | 65.8%      | 22.6%            | 62.2%  | 77%            | 63.6%        | 78%          | 87%         |
|               | PU         | 83.6%      | 62.2%            | 81.8%  | 90%            | 82.2%        | 90%          | 32%         |
| Bigram (TF-IDF) | TCSVM      | 48%        | 10.7%            | 44.4%  | 62%            | -            | 10.2%        | 5.8%        |
|               | OCSVM      | 40.8%      | 26.7%            | 42%    | 57%            | 35.1%        | 52%          | 65.8%       |
|               | PU         | 60.4%      | 20.9%            | 53.2%  | 69%            | 56.4%        | 72%          | 6.7%        |
| Trigram (TF-IDF) | TCSVM      | 28.9%      | 1.3%             | 20%    | 33%            | 25.3%        | 40%          | 1.8%        |
|               | OCSVM      | 66.2%      | 80.4%            | 62.7%  | 77%            | 64%          | 78%          | 96.4%       |
|               | PU         | 33.8%      | 3.6%             | 23.6%  | 38%            | 29.8%        | 46%          | 2.2%        |
| Word2Vec      | TCSVM      | 83.1%      | 76%              | 83.1%  | 91%            | 33.3%        | 50%          | 11.6%       |
|               | OCSVM      | 87.5%      | 88.9%            | 94%    | 95%            | 74.4%        | 85%          | 83.3%       |
|               | PU         | 90.2%      | 89.8%            | 95%    | 87.6%          | 59.2%        | 74%          | 28%         |
| GloVe         | TCSVM      | 79.6%      | 69.8%            | 82%    | 88%            | 72%          | 84%          | 2.2%        |
|               | OCSVM      | 91.6%      | 94.2%            | 97%    | 92.8%          | -            | 34.7%        | 51%         |
|               | PU         | 85.8%      | 88%              | 94%    | 84.4%          | 54.7%        | 71%          | 8%          |
| fastText      | fastText   | 87.1%      | 58.2%            | 74%    | 89%            | 61.3%        | 76%          | 37.8%       |
| fastText      | fastText   | 87.1%      | 58.2%            | 74%    | 89%            | 61.3%        | 76%          | 37.8%       |

Table 7: Robustness/Adversarial Evaluation of Deep ML Classifiers on The Holdout Split.

| Classifier | Embedding                     | Actual ACC | Paraphrasing ACC | EDA ACC | Homograph ACC | Spacing ACC | Charswap ACC | Hybrid ACC |
|------------|--------------------------------|------------|------------------|--------|---------------|--------------|--------------|-------------|
| BERT       | bert-base-uncased              | 80.8%      | 77.8%            | 88%    | 94%           | 68.4%        | 81%          | 24.9%       |
| ELMo       | ELMo                           | 89.3%      | 77.7%            | 88%    | 94%           | 83.9%        | 91%          | 41.6%       |
| RoBERTa    | roberta-base                   | 92.9%      | 83.6%            | 91%    | 92.4%         | -            | -            | 51.6%       |
| XLM-RoBERTa| xlm-roberta-base               | 90.2%      | 78.2%            | 88%    | 89.8%         | 91.1%        | 95%          | 54.2%       |
| DistilBERT | distilbert-base-uncased        | 95.1%      | 83.6%            | 91%    | 92.9%         | 78.7%        | 88%          | 43.1%       |
| LSTM       | WE-Random                      | 75.1%      | 53.3%            | 70%    | 72.4%         | 42.2%        | 59%          | 9.8%        |
| BiLSTM     | WE-Random                      | 70.2%      | 48.4%            | 65%    | 68.0%         | 28.7%        | 45%          | 8.4%        |
| CNN        | WE-Random (static)             | 74.2%      | 50.7%            | 67%    | 70.2%         | 34.1%        | 51%          | 7.6%        |
| TCN        | TCN                            | 78.7%      | 54.7%            | 71%    | 77.3%         | 47.2%        | 59%          | 7.1%        |
| Ensemble   | WE-Random (static)             | 80.9%      | 68.4%            | 81%    | 80.4%         | 60.9%        | 76%          | 24.0%       |
|            | WE-Random (dynamic)            | 83.1%      | 73.8%            | 85%    | 82.2%         | 72.0%        | 84%          | 15.1%       |
|            | WE-Random (CNN-BiGRU)          | 91.1%      | 80.4%            | 89%    | 90.7%         | 87.6%        | 93%          | 42.2%       |

Tackling the imbalance dataset. Currently, all of the public Spam SMS datasets, including the super dataset, are highly imbalanced, therefore it is not suitable to apply traditional ML techniques. As such, it is surprising that imbalanced classification in Spam SMS detection does not get more attention. The OCSVM (one-class learning) in this research cannot perform well, however, it is strongly recommended to explore other approaches like one-class neural networks, etc.

Lack of robustness. One of the most important aspects of this research was to explore how adversarial attacks can be used to target shallow ML and deep ML-based SMS Spam models by generating adversarial samples. Therefore, a second set of experiments was designed for this purpose, and the previously trained models were assessed for robustness using different adversarial tactics. Overall, both the shallow ML and deep ML models had a significant decrease in classification performance when adversarial samples were present. The adversarial evaluation proved that all ML models are currently vulnerable to a wide range of adversarial attacks. Therefore, a robust model for SMS Spam detection with high-performance accuracy is required.

Lack of cross evaluation. Common techniques for training and testing the data involve k-fold cross-validation or splitting the original dataset into training (usually 80%-70%) and testing (usually 20%-30%) data. But in such cases, the source of the data remains the same and the test set may have attributes that are common to the original dataset into training (usually 80%-70%) and testing (usually 20%-30%) data. But in such cases, the source of the data remains the same and the test set may have attributes that are common to the training set—thus ensuring better performance of the classifier. This, however, may not be the case in a real-world scenario. Therefore, the proposed methodology should be cross-evaluated (trained on one dataset and tested on a different dataset). The results received
in Table 6 and Table 7 from unaltered instances of holdout split on the same models are quite low in comparison to the results in Table 3 and Table 4, which clearly prove this point.

**Monitoring the threat landscape.** For effective Spam filtering, it is mandatory to stay on top of adversaries by proactively looking for new threats that may arise due to new technologies like the internet of things or current trends and events (like COVID-19). It is recommended to search and crawl the latest threats from social media (Twitter etc) and scam observatories.

**Lack of generalization abilities and adversarial training of models.** The newly developed Punycode attack for SMS Spam proved to be effective in evading different models by encapsulating the Spam keywords. However, one of the most important use cases of Punycode is to make phishing URLs look like legitimate website URLs, which needs to be evaluated in detail for SmiShing. Moreover, research work needs to be conducted to tackle the fact that SMS Spam filtering is a concept drift problem. Researchers have been working towards this goal of designing a general method. However, none of the existing works meet this need. Therefore, future work should focus more on designing a general defense for a single NLP task and then extending it to other NLP tasks. Adversarial training of ML models is one important aspect to consider for this purpose. Recent adversarial examples can be leveraged to improve the performance and robustness of ML models. Also, we believe that a blend of encoding models would be a plausible solution to this problem.

**Performance, robustness, cost, and time trade-off.** DL models are powerful and have the capacity to solve sophisticated problems by exploiting the computational power of modern GPUs. The performance assessment of shallow and deep ML models in Table 3 and Table 4 respectively clearly proved the dominance of deep ML over shallow ML (high f1-score of several DL models). Moreover, the results received from the robustness assessment of the models indicate that transformer-based DL models can significantly improve model robustness. DL models are slow to train and require a lot of computational power. The shallow ML model, fastText, was much faster than DL for training and evaluation and scored well in Spam classification (f1-score of 82%) and fairly in robustness assessment in comparison to other ML models (performance a little lower than DL models). Therefore, fastText is a viable alternative to DL models in situations where low computation and training time are desired.

**Word embedding as default feature extractor.** The results received in both the performance evaluation and robustness assessment demonstrated the importance of context-based word embedding as a default feature extractor. This is one potential step to counter adversarial attacks along with other techniques such as adversarial training. Any model constructed on top of semantic-based word embedding performs well in terms of Spam classification and robustness.

**Appropriate Evaluation metrics.** From Table 3&4, we concluded that the accuracy metric does not capture the true performance of the classifier in the case of unbalanced data, and currently, all of the public Spam SMS datasets are highly imbalanced. A better way to evaluate would be to include the F1 score along with the accuracy metric.

**Requirement of Collaborative approach.** From the literature, we realised that very few studies were performed on the server side (network providers) due to the poor collaboration of researchers with the industry. Therefore, hybrid techniques that blend content-based and non-content-based aspects with a collaborative approach (solutions that reside both on the network provider’s side and the user’s side) may also need to be investigated. The non-content-based techniques make use of particular message attributes like the message’s size or temporal and network features. This might not only improve the performance but also overcome some of the challenges faced in Spam SMS detection, such as limited textual content, data sparsity, and unstandardized acronyms.

**Crowdsourcing Spam SMS collection.** We recommend a crowdsourcing approach to get the most Spam data, reduce the problem of data imbalance, increase data diversity, and cover the most recent Spam campaigns that are targeting mobile users. Note that contemporary NLP models have a huge number of parameters, so training them without a large dataset is not ideal.

**Lack of better unsupervised ML methods.** We observed an important gap in the detection techniques employed for Spam SMS. The majority of the implementations use supervised learning methods. The major necessity for training and evaluating the performance of such supervised systems is the considerable amount of labelled data, which brings us back to a major security challenge – data availability. Therefore, we implemented the OCSVM for Spam detection in this work, however, more efficient unsupervised approaches need to be evaluated for this problem.

**8 CONCLUSION**

In this study, we presented an accumulated view of the performance and robustness of a large number of AI models in detecting spam SMS. We evaluated and compared the performance of AI models on a large new SMS dataset to determine their efficacy in the evolving threat landscape. We found these models to be highly effective in identifying the Ham SMS. However, only a few deep learning models perform well in classifying Spam with a precision score above 90% whereas others fail to achieve this landmark. Next, we designed several black-box attack strategies using different levels of adversarial perturbations, keeping the present threat landscape in mind, which significantly lowered the effectiveness of all trained models in our large-scale experiments. In addition, we highlighted several challenges and discussed future research prospects. We anticipate that this work will serve as a catalyst for qualitative research leading to the development of a robust Spam SMS detection model.

**REFERENCES**

[1] J. Buchanan and A. J. Grant, “Investigating and prosecuting nigerian fraud,” US Att’y’s Bull., vol. 49, p. 39, 2001.
[2] L. Zhang, J. Zhu, and T. Yao, “An evaluation of statistical spam filtering techniques,” ACM Transactions on Asian Language Information Processing (TALIP), vol. 3, no. 4, pp. 243–269, 2004.
[3] G. L. Wittel and S. F. Wu, “On attacking statistical spam filters.” in CEAS. Citeseer, 2004.
[4] S. Webb, S. Chatti, and C. Pu, “An experimental evaluation of spam filter performance and robustness against attack,” in 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing. IEEE, 2005, pp. 8–pp.
[5] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of sms spam filtering: new collection and results," in Proceedings of the 11th ACM
symposium on Document engineering, 2011, pp. 259–262.

[6] T. Almeida, J. M. G. Hidalgo, and T. P. Silva, "Towards sms spam filtering: Results under a new dataset," International Journal of Information Security Science, vol. 2, no. 1, pp. 1–12, 2010, https://www.ijiss.org/sms-spam-prediction, last accessed 20 Oct 2021.

[7] M. Gupta, A. Bakiwl, S. Agarwal, and P. Mehndiratta, "A comparative study of spam sms detection using machine learning classifiers," in 2018 Eleventh International Conference on Contemporary Computing (IC3). IEEE, 2018, pp. 1–7.

[8] S. Rojas-Galeano, "Using bert encoding to tackle the mad-lib attack in sms spam detection," arXiv preprint arXiv:2107.06400, 2021.

[9] C. Oswald, S. E. Simon, and A. Bhattacharya, "Spotspam: Intention analysis driven sms spam detection using bert embeddings," ACM Transactions on the Web (TWEB), 2022.

[10] S. Y. Yerima and A. Bashar, "Semi-supervised novelty detection with one class svm for sms spam detection," in 2022 29th International Conference on Systems, Signals and Image Processing (WISSP). IEEE, 2022, pp. 1–4.

[11] W. Wang, R. Wang, L. Wang, Z. Wang, and A. Ye, "Towards a robust deep neural network in texts: A survey," arXiv preprint arXiv:1902.07285, 2019.

[12] A. Hug, M. Pervin et al., "Adversarial attacks and defense on texts: A survey," arXiv preprint arXiv:2005.14108, 2020.

[13] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text under a new dataset," in Proceedings of the 5th international ACM SIGIR conference on Research and development in information retrieval, 2020, pp. 1029–1030.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[15] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text for adversarial attacks, data augmentation, and adversarial training in nlp," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 119–126.

[16] F. Salahdine and N. Kaaboob, "Social engineering attacks: A survey," Future Internet, vol. 11, no. 4, p. 89, 2019.

[17] A. Costello, "Punycode: A bootstrapping encoding of unicode for internationalized domain names in applications (idna)," RFC 4379, March, Tech. Rep., 2006.

[18] M. A. Sha’fi, M. S. Abd Latif, H. Ichiro, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, "A review on mobile sms filtering techniques," IEEE Access, vol. 5, pp. 15650–15666, 2017.

[19] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," arXiv preprint arXiv:1812.05271, 2018.

[20] C. Wang, P. Nulty, and D. Lillis, "A comparative study on word embeddings in deep learning for text classification," in Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, 2020, pp. 37–46.

[21] A. K. Uysal, S. Gunal, S. Ergin, and E. S. Gunal, "The impact of feature extraction and selection on sms spam filtering," Elektronika ir Elektrotechnika, vol. 19, no. 5, pp. 67–72, 2013.

[22] O. Abayomi-Alli, S. Misra, A. Abayomi-Alli, and M. Oduasimi, "A review of soft balancing approach and regularized deep neural networks," in International Journal of Information Technology, 2019.

[23] A. Barushka and P. Hajek, "Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks," Applied Intelligence, vol. 48, no. 10, pp. 3538–3556, 2018.

[24] N. H. Imam and V. G. Vassilakis, "A survey of attacks against twitter spam detectors in an adversarial environment," Robotics, vol. 8, no. 3, p. 50, 2019.

[25] J. Graham-Cumming, "How to beat a bayesian spam filter," http://www.spamconference.org, 2004.

[26] P. Laskov and R. Lippmann, "Machine learning in adversarial environments," pp. 115–119, 2010.

[27] J. Peng and P. P. Chan, "Revised naive bayes classifier for combating the focus attack in spam filtering," in 2013 International Conference on Machine Learning and Cybernetics, vol. 2. IEEE, 2013, pp. 610–614.

[28] "ScanWatch - Types of scans," https://www.scanwatch.gov.au/types-of-scans, last accessed 20 Dec 2021.

[29] "Langdetect," https://pypi.org/project/langdetect/, last accessed 20 Oct 2021.

[30] "Google Language Detection API," https://cloud.google.com/translate/docs/basic/detect-language#detect_language_python, last accessed 20 Dec 2021.

[31] T. Chen and M. Y. Kan, "Creating a live, public short message service corpus: the mms corpus," Language Resources and Evaluation, vol. 47, no. 2, pp. 299–335, 2013.
Asking or requesting users for payment
Asking users to contact back on the same number or
Asking users to contact on email within text message
Asking users to forward or circulate the message
Asking users for personal or sensitive information
Messages with users to visit or tap on the links within

We derived a set of 8 rules to govern our manual labelling of SMS
was poor to that of DL models (see Figure 5).
The work was done while holding a joint PhD scholarship spon-
A.1 Spam Labelling Rules

| Rule | Description |
|------|-------------|
| Rule1 | Promotional or unwanted messages (advertising, prose-
| Rule2 | Messages with users to visit or tap on the links within message |
| Rule3 | Asking users to contact on email within text message |
| Rule4 | Asking users to contact back on the same number or another contact number within text message. |
| Rule5 | Asking users for personal or sensitive information |
| Rule6 | Asking or requesting users for payment |
| Rule7 | Asking users to forward or circulate the message |
| Rule8 | Asking users to download or install a file |

A.3 Explanations
Word2Vec uses a neural network to learn word associations from large corpora and generate a vector representation for words or tokens and captures semantic similarity between words. Word2Vec creates a vector space with each unique word in the corpus, often with several hundred dimensions, such that words with similar contexts in the corpus are close to one another in the space. Moreover, the vectors for unknown words are randomly initialized using a generic normal distribution.
BERT is a language model developed by Google. Generally, language models read the input sequence in one direction: either left to right or right to left. This kind of one-directional training works well when the aim is to predict or generate the next word. But in order to have a deeper sense of language context, BERT uses bidirectional training. BERT applies the bidirectional training of Transformer to language modeling and learns the text representations. BERT generates a language model depending on its context, capturing lexical, semantic and grammatical features. BERT is designed to pre-train deep bidirectional representations from unlabeled text. Consequently, the pre-trained BERT model is used for extracting word embedding/representations. LSTM Due to the vanishing gradient issue, simple RNNs are incapable of representing longer contextual connections. They have mostly been supplanted by so-called long short-term neural networks (LSTMs), which are similar to RNNs but can capture the lengthier context included in texts [67].
Bidirectional LSTM (BiLSTM) LSTMs can only process unidirectional sequences. Consequently, state-of-the-art techniques based on LSTMs evolved into so-called bidirectional LSTMs that can read the context left to right as well as right to left [68].
Bidirectional Gated Recurrent Unit (BiGRU) Bidirectional GRU’s [69] are a type of bidirectional recurrent neural networks with only the input and forget gates. It enables the utilisation of knowledge from prior and subsequent time steps to create predictions about the present state.
DistilBERT The DistilBERT [70] is a lighter, faster, and more affordable version of BERT. It learns a distilled (approximate) version of BERT that retains 95% of the performance but uses half the parameters. Specifically, it lacks token-type embeddings, a pooler, and preserves just half of Google’s BERT’s layers. DistilBERT makes use

ACKNOWLEDGEMENT
The work was done while holding a joint PhD scholarship spon-
A APPENDIX
A.1 Spam Labelling Rules
We derived a set of 8 rules to govern our manual labelling of SMS as Spam or Ham. The guidelines were developed after a thorough analysis of labelled SMS corpora, debate among the authors and review of different scam types on scam observatories (scam watch, Action Fraud). A single expert analysed the unlabeled SMSes in the consolidated dataset and labelled them using the defined labelling criteria. In the case of two or more reviewers, there may be differences in judging the intent of SMSes. The Spam labelling rules we utilised are summarised in Table 8.

A.2 Adversarial Evaluation of ML Models
As discussed in Section 6, Adversarial Evaluation of ML Models reveals that all of them are vulnerable to some type of attack. In most adversarial techniques, the classification accuracy of ML models was poor to that of DL models (see Figure 5).
of a method called distillation to resemble Google’s BERT, i.e. replacing the huge neural network with a smaller one. Following training a big neural network, the network’s whole output distributions can be approximated using a smaller network.

**Robustly optimized BERT approach (RoBERTa)** RoBERTa [77] is a retraining of BERT that incorporates an enhanced training process, 1,000% more data, and significantly more computational capacity. To improve the training approach, RoBERTa omits the Next Sentence Prediction (NSP) task from BERT’s pre-training and introduces dynamic masking, in which the masked token fluctuates between training epochs. Additionally, it was demonstrated that bigger batch sizes were more effective during the training process.

**Cross-lingual Language Model (XLM)** XLM is an enhanced version of BERT suggested by Facebook AI [78] to deliver state-of-the-art results in a variety of natural language processing tasks, most notably cross-lingual classification, supervised and unsupervised machine translation. XLM employs a dual-language training mechanism in conjunction with BERT to acquire knowledge about the relationships between words in multiple languages. When a pre-trained model is used to initialize the translation model, the model outperforms other models in a cross-lingual classification problem and considerably improves machine translation.

![Figure 5: A bird’s-eye view of the adversarial evaluation of ML models. The results of different attacks against each model are stacked together.](image)