OCR Post-correction for Detecting Adversarial Text Images

Niddal H. Imam[a,*], Vassilios G. Vassilakis[a] and Dimitris Kolovos[a]

[a]University of York, Heslington, York, YO10 5DD, United Kingdom

ARTICLE INFO

Keywords:
Deep Learning
Spam image
OCR
Text Recognition
Text Classification
Adversarial Text Attack

ABSTRACT

The amount of images with embedded text shared on Online Social Networks (OSNs), such as Twitter or Facebook has been growing in recent years. It is becoming important to analyse the images uploaded into these platforms, as adversaries may spread images with toxic content or misinformation (i.e. spam). Optical character recognition (OCR) systems have been used to detect images with malicious content, where the embedded text gets extracted and classified using machine learning algorithms. However, most existing OCR-based systems are adversary-agnostic models, in which the extracted text from an image is not checked by humans before the classification. Consequently, these fully automated models become vulnerable to minor modifications of images' pixels or textual content (*e.g., character-level perturbations*), which do not affect human understanding, but could cause the OCR systems to misrecognise the embedded text. In this paper, we propose an OCR post-correction algorithm to improve the robustness of OCR-based systems against images with perturbed embedded texts. Experimental results showed that our proposed algorithm improves the robustness of three state-of-the-art OCR models with at least 10% against adversarial text images, and it outperforms five spellcheckers in correcting adversarial text. Also, we evaluated the perceptibility of our adversarial images, and this study showed that 91% of the participants were able to correctly recognise the adversarial text images. Additionally, we developed an adversary-aware OCR-based system for detecting adversarial text images using the proposed algorithm, and our evaluation results showed considerable improvement in the performance of an OCR-based system.

1. Introduction

Extracting and understanding text in images (e.g., printed and handwritten documents, or natural scene text) is an area of study that has been widely researched in recent years, due to the increasing amounts of images shared on different Online Social Networks (OSNs) platforms. Optical Character Recognition (OCR), a technology for extracting text from images, has been the leading technology to understand text embedded in images. The general system of OCR consists of two components: text detection and text recognition. The success of OCR in extracting text from cleaned documents has led to its adoption as a prepossessing step in many real-world applications, such as Neural Machine Translation (NMT) [26], license plate recognition [53], cancer classification [55], and recently, spam detection [6].

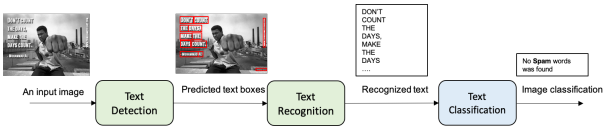

Figure 1: OCR-based system for detecting spam images

The process of classifying images with embedded text using OCR systems involves three steps: text localisation (detection), text recognition, and text classification (see Figure 1) [22]. The text localisation and recognition are used to extract the embedded text, while the last step is for classifying the text. There are two main challenges of using OCR systems for image classification in an adversarial setting, choosing the best text extraction and text classification methods. First, extracting embedded text can be either at sentence-level or word-level. In the first method, an OCR reads the image line-by-line, and the detected text gets checked and corrected by lexicon-based transcriptions [41], whereby the prediction is constraint to a spellchecking dictionary. As most of the text classification models depend on the input context when making their prediction, extracting text from images at sentence-level is suitable for scanned documents, where the embedded text is distributed in lines (see Figure 2). However, word-level text extraction, in which embedded text is extracted as a list of words, is more suitable for images found in OSNs. Figure 3 presents two examples of images with embedded text that is distributed all over the image being posted on Twitter. Extracting the embedded text from such images by OCR systems that extract text at sentence-level is however not applicable. Thus, related studies in detecting spam image use word-level OCR [7]. Nevertheless, classifying list of words is another challenge, as it could lead to higher false positive rates. For example, the simplest way of classifying a list of words is by using a blacklist, but typical spam words (e.g. Viagra or call) might appear in non-spam images. Although advanced text classification methods use state-of-the-art Natural Language Processing (NLP) models to classify extracted text from images, small text perturbations can fool these models [42, 20, 15].

Another challenge is the robustness against adversarial examples. There is a rapidly growing concern about test-time attacks against neural network image classifiers. Some studies investigated the robustness of this type of classifiers against black-box model queries, whereas others studied the robustness to the gradient-based optimization [39]. These

ni571@york.ac.uk (N.H. Imam); vv573@york.ac.uk (V.G. Vassilakis); dimitris.kolovos@york.ac.uk (D. Kolovos)

ORCID(s): 0000-0001-8399-0449 (N.H. Imam); 0000-0003-4902-8226 (V.G. Vassilakis); 0000-0002-1724-6563 (D. Kolovos)

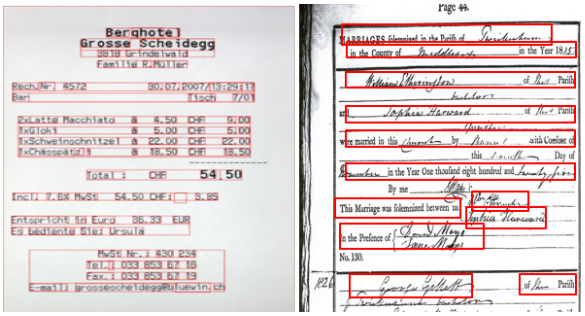

Figure 2: Examples of scanned documents

Figure 3: Examples of Twitter's images

attacks can either be targeted (i.e., designed for a specific model) or non-targeted. There are two types of adversarial examples in image processing models: adding perturbations to the images pixels, or manipulating the embedded text in images. Real-world applications are highly dependent on the correctness of the OCR outputs. Mistakes could lead to serious consequences. For example, a parking fine ticket could be issued to a wrong car; incorrect medical diagnoses might be produced, or messages may be misclassified as spam [7, 5, 42]. Previous works focus on devising adversarial examples against OCR systems by adding perturbation or noise to images using Fast Gradient Sign Method (FGSM [46]), DeepFool [32], or Generative Adversarial Network (GANs) [18]. However, analysing extracted text from the OCR system has not been well studied in the literature. This type of attack can fool OCR-based systems. Lexicon-based OCRs use predefined lexicons and they could be useful for some tasks concerned with only producing outputs that are likely to be words in the target natural language, whereas lexicon-free OCRs use Connectionist Temporal Classification (CTC) [33]. Lexicon-based OCRs may fail if used in security applications, as they cannot recognise manipulated texts, and do not help security analysts to detect new attacks, as they only correct output texts. Although some lexicon-free OCRs can recognise manipulated texts, their output may fool the deployed text classifier. These text classifiers (e.g., NLP applications) are very sensitive to certain words when making predictions, so a small manipulation on the image's textual content could cause the OCR-based system to misclassify the image [52]. Song and Shmatikov [42] stated that there are no automated systems that could check whether the text produced by OCR "makes sense". Also, they mentioned that systems sensitive to out-of-context types of text manipulation, would be prone to false positives and adversarial attacks. Li et al. [29] added that a few defence methodologies have been proposed for adversarial text attacks. Kurita et al. [27] stated that adversarial text attacks are different from users' errors (e.g., misspellings and slang), as users do not initially attempt to avoid being detected.

While existing works investigate the effect of adversarial text attacks against NLP applications, it is more challenging to handle such attacks against OCRs. Related studies discuss two potential defence approaches, spell checking and adversarial training [29, 27]. In adversarial training, noise is added to generate adversarial examples in the training dataset. One of the limitations of this method is the need for the defender to identify details of the incoming attack, such as the strategy and lexicon used by the adversary. Also, the model is trained in an adversarial training fashion, which could be over-fitting to the adversarial examples; thus, leading to worse performance on clean datasets. On the other hand, using spell checking algorithms is the most common defence method against character-level perturbation in NLP tasks [29]. Although spell checking methods could detect and correct errors or adversarial examples, they cannot be applied in all domains because their performance varies depending on the type of misspelling [2].

In this paper, we investigate the effect of adversarial text images (i.e., images with embedded adversarial text) to OCR-based systems. This type of attack against OCR has not been well studied although adversarial text attacks have been widely discussed in the literature against NLP applications; these attacks may involve character flips [20], scramble text [27], substitute characters [29], or visual perturbations [15]. We preformed a character-level adversarial attack against OCR-based systems, where adversaries perturbed embedded text in an image by replacing a few characters to cause the text recognition part of the OCR (e.g., CTC) to misrecognise the text. We assumed that the adversaries have very limited knowledge about the deployed OCR (black-box setting). Components of automated systems are rarely checked by humans, which makes them vulnerable to adversarial attacks. Adversaries can take advantage of this vulnerability to attack the OCR part of a system [42]. These attacks are common in OSNs, as they neither require any knowledge about the deployed model, nor any linguistic knowledge and human understanding [15]. In our previous study [23], we encountered similar behaviours in health-related spam campaigns on Twitter Arabic hashtags, where adversaries purposefully misspelt spam words that are embedded in images (see Figure 4). Thus, we proposed a text classification step for OCR-based spam detector, in which we used a black/white list method with human assistance to detect new or modified words (adversarial examples) [22]. However, to reduce human intervention, in this current paper a spellchecking-based algorithm was employed as an OCR post-correction step for detecting and tracking malicious text embedded in images was proposed. After the embedded text

in an image gets extracted by the deployed OCR system, the proposed algorithm detects the manipulated (i.e., adversarial) embedded texts and de-noises them before they feed into the text classifier model.

Figure 4: Images with embedded manipulated text

This study aims to answer the following research questions: **RQ1**) Is the recognition of the state-of-the-art OCR systems affected by the adversarial embedded text in images (i.e., adversarial examples)? **RQ2**) Are autocorrectors (i.e., spell-checkers) sufficient for OCR-based systems to mitigate the designed adversarial examples? **RQ3**) How can we improve the robustness of OCR-based systems to mitigate the designed adversarial examples? Our main contributions are as follows:

1. We developed a black-box attack method that can generate images of manipulated embedded text to cause OCR-based systems to mis-recognise the text.
2. Human perception of the generated adversarial text images is an important feature of adversarial examples. We evaluated the perceptibility of our adversarial images by conducting a user study, and the results showed that human understanding is not affected by the manipulations.
3. We proposed an OCR post-correction algorithm for denoising and classifying manipulated embedded text in images. Specifically, the proposed method has been designed to improve the robustness of OCR-based detectors.
4. We developed an adversary-aware OCR-based detector that is robust to adversarial text images, adaptabile to evolving attacks and interpretable to security analysts.

The rest of the paper is organised as follows: Section 2 discusses related work in the field of OCR systems. In Section 3, the problem formulation is presented. The datasets, methodology, and our proposed algorithm are presented in Section 4. We quantitatively measured human perception of the generated attacks and present our findings in Section 5. Section 6 presents the results and analysis of three experiments preformed to answer the research questions. The discussion of the experiments' results and its limitations is presented in Section 7. Finally, Section 8 concludes the paper and discuss future work.

2. Related Work

Related work to our study is divided into two folds: adversarial text attacks and defensive methods in images classification and text classification tasks. Since there is paucity of research that investigates adversarial text attacks against OCR-based models, we discussed adversarial attacks against NLP applications. OCR-based systems often use NLP applications as components of their systems. Thus, it is important to understand attacks and defence methods proposed against these applications.

2.1. Adversarial Text Attacks and Defences in Images Classification Tasks.

OCR-based Security Models. Borisyuk et al., [6] conducted one of the first studies that developed an OCR system for detecting and recognizing text in images uploaded to Facebook. The system, called Rosetta, consists of two models: text detection and text recognition. The text detection model uses Fast Convolutional Recurrent Neural Network (Fast-RCNN) to perform word detection. Then, for each detected box, a fully-convolutional model, referred to as Connectionist Temporal Classification (CTC), is used to recognize text [19]. The recognition model predicts the most likely character at each detected box in the image. Also, Yuan et al., [52] developed a model called Malena, which can detect different types of spam including images carrying text, number, or QR code in Chinese social networks (Baidu, Tieba, and Sina Weibo). The authors used a PixelLink-based OCR [10] for detecting text in images. They generate 200 adversarial examples using the state-of-the-art CW approach [9] to evaluate the robustness of their OCR, and they successfully detected 196 of them. Tramer et al. [45] developed a framework for blocking adversarial ads in Facebook and web pages in general. The proposed framework takes a screenshot of the page, locates images by using off-the-shelf object detector, Yolov3; and then extracts text from images using Tesseract OCR. The authors evaluated the robustness of the framework different against evasion attacks. They evaluated the robustness of Tesseract OCR against the CW attack (ℓ_2 norm).

Remarks. Although these studies employed OCRs for security tasks; the vulnerability of these security models to manipulated textual content of images has not been investigated. *Rosetta* was designed to detect spam images, but the vulnerability of the proposed OCR has not been evaluated. The authors of [52] and [45] evaluated the robustness of their OCRs using gradient-based attacks (e.g. adding noise or blur to images). To the best of our knowledge, our work is the first to investigate the robustness of OCR-based systems against images with manipulated embedded text.

Adversarial Text Attack Methods. Adversarial text attacks against OCR systems could be carried out either by adding noise to the text locations in images, or by injecting adversarial text into images. Although a large number of works have been studied in a bid to examine the former attack, no previous study has been conducted towards investigating the later type of attack. Song and Shmatikov [42] presented the first study of adversarial examples against sequence labelling models in the image domain. They proposed several gradient-based adversarial

attacks against CTC-based OCRs that use Tesseract-ocr, by adding perturbation to the most influential characters or words in the scanned documents. The authors successfully caused Tesseract-ocr to output the desired adversarial texts with 84.8% accuracy. Also, they showed that their attack could detect the prediction accuracy of NLP applications that use OCR systems for pre-possessing. One limitation of their attack is the transferability to different OCR systems.

Defence Methods. There are several studies that used post-OCR correction to correct (denoising) OCR-ed texts. One of the post-OCR correction methods is to use spell checkers for correcting OCR's errors [49, 35]. Taghva and Stofsky [43] proposed spelling correction system (OCR-Spell) for correcting OCR errors in text. The system uses Longest common subsequence calculation to correct errors of segmentation part of the OCR, such as iii → m , or cl → d . In [8], authors proposed post-processing steps to improve OCR's accuracy using the Aspell API and a customized words list. Thompson et al. [44] proposed a customised OCR correction for Historical Medical text. The authors compared the results of 4 spelling correctors: ASpell, Hunspell, Microsoft Word, and MAC OS. Similarly, authors in [16] compare the results of four string-to-string transformation models: (DirecTL, SEQUITUR, ALISETRA, and Contextual Edit Distance) in spelling error correction paradigm. Additionally, a couple of competitions were organised by International Conference in Document Analysis and Recognition (ICDAR) on post-OCR text correction. Participants were asked to perform two tasks: OCR-error detection and correction. Several methods have been proposed in ICDAR-2019, such as context-based Character Correction using BERT [11] and dictionary-based detection.

Remarks. The related studies use a post-correction step to improve the recognition of OCR systems that are designed for analysis of the scanned documents. The tasks of extracting and understanding text in scanned documents are easier than images uploaded into OSNs. Images uploaded into OSNs are much nosier as they could contain overlaid texts on top of images. Also, the embedded text in images cannot be read line-by-line, because the rotation of the embedded text can either be vertical or horizontal. Additionally, these studies showed that applying automatic spellchecking to misspelt words alone is unreliable. Some of the shortcomings of spellcheckers that need to be addressed include, complexity, out-of-vocabulary words (OOV), and many others. For example, OCRSpell is complex and requires extensive feature engineering, as its operation involves five models. Other studies [8, 44] have not considered detecting and tracking manipulated texts. In this paper, we focused on OCR systems designed for analysing images with embedded text. The perturbation of multiple characters in a document does not affect human understanding, as they can infer the meaning of the perturbed words from the context. On the other hand, images with embedded text tend to contain less text, which is a constraint that needs to be considered by adversaries when adding perturbations to text in images.

2.2. Adversarial Text Attacks and Defence in Text Classification Tasks.

Gradient-based Attack Methods. Character-level perturbation attacks against text classification models have been widely studied in the literature. These attacks can be launched as either in a white-box or black-box setting. Heigold et al. [20] proposed several character perturbation methods for attacking NLP models (e.g., bpe-LSTM-BLSTM, char-LSTM-BLSTM and char-CNN Highway-BLSTM). They found that character-based approaches are more sensitive than BPE-based approaches. Ebrahimi et al. [13] proposed the HotFlip method for generating adversarial examples against character-level neural classifiers. The attack targets the one-hot encoding step of the character-level embedding process. They chose the best character to be flipped by computing gradient with respect to one-hot encoding. Also, they used a beam-search optimisation to find a set of manipulation (flip, insert, delete) that could fool the deployed classifier. Moreover, the authors extended HotFlip in Ebrahimi et al. [12] by adding targeted attacks against character-level NMT. The authors concluded that adversarial training could improve the robustness of the deployed text classifier against such attacks. Similarly, Miyato et al. [31] investigated the robustness of recurrent neural networks (RNNs) by perturbing the continuous word embedding, rather than the discrete word inputs, such as one-hot vectors. Their results showed that adversarial and virtual adversarial training improved the classification performance and the quality of word embeddings. Also, Gong et al. [18] proposed adversarial text attack against Convolutional Neural Network (CNN), with only a few words changed using FGSM and DeepFool. On the other hand, Belinkov and Bise [5] proposed several black-box attacks against NMT, such as swap, middle and fully random, and keyboard type. They treated typos and misspellings as adversarial attacks. Their experiments showed that models trained on mix noise performs worse than models trained on a specific type of noise.

Other Attack Methods. Unlike previous works that used projected gradient, the authors in [17] proposed a novel framework, DeepWordBug, which generated character-level perturbation on text data. The authors swap, substitute, delete, and insert characters, to generate OOV, in a bid to force RNN to classify text as unknown. Their method of attack successfully reduced the prediction accuracy of word-LSTM and char-LSTM models. Also, they concluded that adversarial training could improve the robustness of these models against their attack better that using auto-correctors. Li et al. [29] proposed adversarial attacks against Deep Learning Text Understanding (DLTU) in black-box and white-box settings. They proposed five bug generation methods: insert, delete, swap, substitute-character, and substitute-word. The authors evaluated their attacks against spelling checker and adversarial training. The results showed that spelling checker could be used against adversarial text attacks. Whereas, the effectiveness of adversarial training reduced in defending against unknown adversarial attacks.

Schuster et al., [39] proposed a non gradient-based attack at training time, in which they changed words locations in the embeddings to misclassification. The authors investigated two defence methods: using anomaly detection in word frequencies and filtering out high-perplexity sentences. They found that filtering out sentence with ungrammatical sequence of words was the better defence method; whereas, using conjection-based and perplexity-based type of poisoning attacks could evade such detection method.

Defence Methods. Rojas Galeano [38] proposed a method for detecting the homoglyph anomaly, which is a type of text manipulation intended to circumvent verbatim-based filters (e.g. sma!l or s.m.a.i.l). The authors used a penalty function crafted specifically to homoglyph substitutions-type of manipulation, aimed at detecting and tracing the locations of the potential obfuscations in text. The function takes users' generated text and an obscenity (vulgarity) as inputs and computes the edited distance between the two. The proposed edit penalty function compares between characters of the two inputs, and if admissible substitution symbols or bogus segmentation characters are found, these characters' positions are assigned 0, otherwise the positions will be 1. Although the proposed homoglyph/segmentation–safe distance (HS-dist) outperforms Levenshtein distance (L-dist) in discovering homoglyph similarities between obfuscations and obscenities, HS-dist runs four-times slower than L-dist.

BERT-based Defence Methods. Recent studies have evaluated the robustness of BERT, which is one of the best state-of-the-art methods for NLP benchmarks and sentiment analysis tasks. Authors in [36] proposed a semi-character based RNN (ScRNN) to tackle character-level adversarial attacks. The proposed defence method consists of two stages: a word recognition model and a classifier. The defence method was able to restore BERT's accuracy from 45% to 75% against character-level adversarial attacks. Another BERT-defence method was proposed in [25] and it outperforms spell checker and ScRNN. The authors have not considered the adaptability and interpertability which are important factors to ensure longevity. Additionally, Eger et. al.,[14] propose catalogue and benchmark of low-level adversarial attacks against NLP's models. Their results show that the Robustly optimized BERT approach (RoBERT) is not robust to several benchmark attacks.

Remarks. These attacks used norm restrictions to ensure the validity of the perturbations. However, applying the same method for crafting adversarial text examples require some adaptations. A survey conducted by Zhang et al. [54] provides the difference between attacking Deep Neural Networks (DNNs) using adversarial images and text. There are two main challenges when generating adversarial text: 1) the gradient-based adversarial attack cannot be directly applied to the discrete data; 2) the level of manipulation is constrained by human perception [50]. The closest attack method to the attack investigated in this paper is the DeepWordBug [17], but our attack targets OCRs and we focused on replacing characters with visually similar symbols or numbers. In terms of defensive methods, The discussed studies have shown that adversarial training is very effective against these gradient-based attacks. However, such defence requires knowing details about the incoming attacks, and the trained model may become overfitted to the adversarial examples [29, 42]. Additionally, the possible forms of a manipulated word makes the training dataset more sparse [20]. The authors in [39] discussed defending adversarial text attacks by measuring the perplexity of sequences (i.e., how linguistically likely a sequence is), and they found that adversaries could evade such methods by deliberately reducing the perplexity. The algorithm proposed in [38] requires an extra step to find the matching words before calculating the edited distance. Also, it requires manually building a blacklist of potential words that could be manipulated by adversaries. Since calculating the edited distance, adding extra step and retraining ScRNN [36] are expensive and time consuming, we choose to use a spell checker in this paper. Although the model proposed in [25] outperforms the spell checker and ScRNN, it consists of 3 steps, which include a BERT and a language model. This adds extra computation time and complexity when used as an OCR post-correction. Since the authors have not considered how these models can evolve over time, they cannot detect new modified words. Table 1 summarises the related works to our current paper in terms of the model used, type of attack and defence method.

3. Attacks Against OCR-based Detectors

3.1. Problem Formulation

Consider an input image with embedded text sequence $X = [X_1, X_2, ..., X_N]$, where X is a sentence that consists of a number of tokens (i.e, words) X_N, and N is the number of words in a sentence. Each word X_N consists of characters $X_N = [x_1, x_2, ..., x_n]$, where n is the number of characters in the word. The deployed OCR model f scans the input X image's contents and outputs the class of the image $Y = [Y_1, Y_2, ..., Y_M]$, which is a predicted sequence of words, and M is the number of predicted tokens. The neural network of CTC that is used by the OCR outputs a sequence of probability vectors for each word $f(X_N) = Y_M$ where $y_m \in [0, 1]^{|\Gamma|}$ is the probability distribution over all characters (alphabets) Γ at position m. Since the length of the input sequence n and predicted sequences m are not generally equal $m \geq n$, it is hard to measure $p(Y|X)$ from $f(X)$. Thus, a *valid alignment* a of y is used to measure $p(Y|X)$. If sequence $a = [a_1, a_2, ..., a_i]$ and $a_i \in \Gamma \cup \{blank\}$ can be turned into y by removing blanks, which symbolized by $-$ and sequential duplicate characters, a is considered a valid alignment of y. For example, [b, b, -, u, y, y] is a valid alignment for [b, u, y] [42, 19]. To this end, there are two problems OCR-based systems can face at two stages under an adversarial attack.

At the text recognition stage, a pre-traind CTC-based OCR model f is used to map $X \rightarrow Y$. An adversary aims to launch an untargeted attack that causes the CTC neural network to misrecognize a few characters of the input sequence $x \epsilon X$ and predict invalid alignment $\acute{a}$ that is

Table 1
Related works that studied adversarial text attack either against OCR-based or NLP-based applications

Title	Model	Attack	Defence
Rosetta- Large Scale System for Text Detection and Recognition in Images [7]	OCR-based system for detecting spam in Facebook	None	None
Stealthy Porn-Understanding Real-World Adversarial Images for Illicit Online Promotion [52]	OCR-based framework for spam images in Weibo	C&W	None
AdVersarial-Perceptual Ad Blocking meets Adversarial Machine Learning [45]	OCR-based framework for detecting adversarial ads in facebook and the web page	C&W	None
Fooling OCR Systems with Adversarial Text Images [42]	OCR-based for license plate recognition system and scan document	optimization-based	None
Strategies for Reducing and Correcting OCR Errors [38]	OCR-post correcter model for OCR error correction	None	None
Improving OCR Accuracy for Classical Critical Editions [8]	OCR-post correcter model for OCR error correction	None	None
A Tool for Facilitating OCR Postediting in Historical Documents [43]	OCR-post correcter model for historical documents correction	None	None
On Obstructing Obscenity Obfuscation [38]	Verbatim-based filters	character level perturbations	Spelling correction-based
Towards Robust Toxic Content Classification [27]	NLP-based model for toxic comment detection	character level perturbations	Denoising Autoencoder
Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers [17]	NLP-based classifier	character level perturbations	None
TEXTBUGGER-Generating Adversarial Text Against Real-world Applications [29]	Deep Learning-based Text Understanding (DLTU)	character and word level perturbations	None
Our paper	OCR-based system for detecting images in Twitter	character level perturbations	Spelling correction-based

different from the ground truth $a \epsilon Y$. So that $F(\acute{x}) = a(a \neq y)$. For example, an adversary can manipulate one or two characters of an input image's textual content, which could cause the CTC to misrecognise the text since the conditional probability of CTC depends on the probability vectors and the ground truth label [33]. Figure 5 shows an example of character-level perturbation.

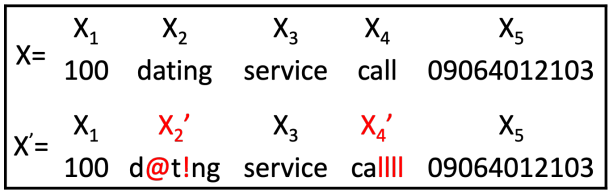

Figure 5: An example of adversarial text

At the text classification step, NLP models (e.g., BERT) are used to classify the extracted text. For example, in Figure 5 the manipulated embedded text of the input image $X = [X_1, X_2, X_3, X_4, X_5]$ is $X' = [X_1, X_2', X_3, X_4', X_5]$. The manipulated words X_2' and X_4' could cause the deployed classifier to missclassify the input image. Even if we train the CTC to recognise manipulated characters, the extracted manipulated text could fool the deployed text classifier because NLP applications are sensitive to character-level perturbations. Utilizing adversarial training to improve the robustness of the deployed NLP model against this attack is challenging since the possible perturbed word formed make the training datasest more sparse. For example, flipping at most one character of a word of length n could make up to n^C different word forms, where C is the number of characters in the vocabulary [20]. Such an attack is common in OSNs as it does not require knowledge about the functionality of the deployed OCR-based system. The adversary only needs to know about the blacklist (e.g., sensitive words) used by the deployed model, which can easily be done in OSNs through exploratory attack, and then manipulate these words [23]. Hence, a pre-processing step that could check the extracted text from images before it being fed into the deployed text classifier is needed.

3.2. Threat Model

The adversaries' goal is to violate the integrity of the deployed OCR-based detector by obfuscating detection through manipulating images embedded text. The attack specificity can either be targeted or indiscriminate. Here we consider targeted attacks, in which the adversaries manipulate specific words (e.g., spam or toxic words). The adversaries' knowledge about the deployed detector is assumed to be very limited (black-box attack setting). In OSNs adversaries could learn about the deployed detector by sending carefully crafted massages and use the detector's feedback to learn some of its characteristics [23]. However, they do not know the details of the deployed model. In terms of the adversaries capability, it is assumed that the adversary is only capable of influencing the deployed detector before the testing stage (*exploratory attack*). Although some studies showed that adversaries are capable of manipulating the training data (*causative attack*), we consider the more realistic attack, in which they can only influence the detector before the testing stage. This assumption can be generalised to different OSNs that utilize OCR-based detectors since these platforms share

a lot of similarities. Generally, the steps of OCR systems are easy for adversaries to reconstruct [42].

4. Methodology and Design

This section describes the methodology used in this work. First, the datasets used for building and evaluating the proposed OCR-based system. Then, we present the algorithm used for generating the adversarial examples. Finally, the proposed defence method is described.

4.1. Datasets

We study adversarial examples of images with embedded text on three benchmark datasets created for text classification tasks. The testing datasets were partially manipulated and used for evaluation. These datasets were chosen because they are widely used in related studies that focused on detecting malicious activities (e.g., spam and toxic or offensive comments detection).

SMS Spam Dataset[1] We used the dataset built by Almeida et al. [1], for building our images dataset. The adopted dataset is a collection of SMS massages collected from the Grumbletext Web site, which is a UK forum created for cell phone users to make public claims about SMS spam messages. The dataset consists of 5,574 short messages; 4,827 ham and 747 mobile spam messages.

Jigsaw[2] dataset consists of 215,000 annotated comments from Wikipedia, and it was used in one of Kaggle's challenges (Toxic comments classification Challenge). The dataset contains six attributes (toxic, severe toxic, obscenity, threat, insult, and identity hate). However, we only use one of them which is a general toxicity label.

The offensEval 2019[3] dataset consists of 13,240 tweets that have been annotated through crowd-sourcing. The dataset is labelled into two classes: offensive and non offensive tweets [27]. It was built for identifying and categorising offensive language on Social Media.

These datasets were used to create our images since there are no publicly available datasets that suit our needs. All the generated datasets were published at Mendeley Data [21]. The following section discusses the steps preformed for building our dataset.

4.2. Adversarial Text Images

Defining gradients in symbolic text is hard [17], so we generated adversarial examples (i.e., images with embedded adversarial text) directly on the images content through the following steps. First, we used Algorithm 1 for automatically generating adversarial text. The algorithm scans the input's characters, and if a character in the input matches one of the characters in the mapping list, it flips the character. The list includes the following: !, @, $, 1, 0, and more symbols can be added. If an input word includes any of the characters in the character map list, that character will be flipped to its visually similar symbol or number. We chose the most frequent spam words in SMS Spam dataset, toxic words in Jigsaw dataset, and offensive words in OffensEval 2019 dataset using Term Frequency-Inverse Document Frequency (TF-IDF). Then, we used a synthetic data generator [4] for embedding the perturbed text into images. Figure 6 provides examples of the adversarial examples used in this study

Algorithm 1: Adversarial Example Generator

Input: word $X = (x_1, ..., x_n)$
Output: adversarial word X'

```
/* most frequent pairs of letters mapping dictionary      */
SymbolMap_Dict = [a: @, s: $, i: !, l: 1, o: 0]
for each char x_n in word X do
    if char in SymbolMap. Keys then
        word.replace(char, SymbolMap.get(char)
end for
Return X'
```

Figure 6: Images with embedded adversarial text

4.3. Proposed Defence Method

Directly applying an automatic spellchecker to misspelled words is often unreliable. The suggested candidate words of the deployed spellchecker could be wrong or, the word list used in the spellchecker might not include some words. These limitations could correct the input word with an error [8], which could increase the false negative rate in security applications. Also, spelling correction algorithms cannot differentiate between typos or misspelling and text perturbations. Spell checkers often correct mistakes and compute the edit distance, which could tell whether the mistake is addition or deletion. In our case, we want to know whether the misspelling is an adversarial example (e.g., replacement or swapping), or not and to keep tracking these adversarial examples so we can update our system. An important point that needs to be considered when designing a detection system in an adversarial setting is the ability to handle *adversarial drift* (i.e, a drift that may occur as a result of new adversarial examples) [40]. Thus, the spell checker must either find the closest correction or print out the error, as it might be a new kind of manipulation.

Proposed Algorithm. In order to overcome the above shortcomings, we propose an algorithm [5] for detecting adversarial text embedded in images. The proposed algorithm was inspired by a basic spelling correction algorithm that was proposed by Peter Norvig [34]. Our algorithm denoises

[1] https://www.kaggle.com/uciml/sms-spam-collection-dataset/data
[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
[3] https://competitions.codalab.org/competitions/20011
[4] https://github.com/Belval/TextRecognitionDataGenerator
[5] https://github.com/niddal-imam/Post-OCR-Correction

errors and outputs the class of the error detected in the text (e.g., repeated characters, swapped characters, substituted characters, or OOV). Detecting the type of error helps in distinguishing between adversarial attacks, typos, or misspellings along with other adversarial activities that need to be considered, such as the importance of the word being perturbed. The process of the proposed algorithm involves three steps: lookup, scanning, and correction. First, it checks if the extracted word includes errors by looking up for a matching word in the dictionary. Second, if a matching word does not exist, the algorithm will check if it contains any symbols, repeated or swapped characters to be denoised. Also, this scanning step tracks the type of error and checks if the corrected word exists in the dictionary. For example, if a symbol or repeated character is detected, the algorithm will substitute the symbol or delete the repeated character and denoise the text by using the adopted spell checker. The adopted spell checker finds the correction c, out of all possible candidate corrections, that maximizes the probability that c is the intended correction, given the original word w [34]:

$$argmax_c \in_{candidates} P(c)\ P(w|c)\ /\ P(w) \quad (1)$$

Finally, if the extracted word (i.e., input) cannot be corrected, it will be classified as OOV. As mentioned above, adaptability of the detection system is important in an adversarial setting. Hence, extracted text that could not be corrected will be collected and used for updating the system as these OOV might be new adversarial examples. The output of the algorithm will be the input correction and the type of detected error.

5. Human Perception

We quantitatively measured human perception of the generated images with embedded perturbed text. This is an important step when devising adversarial examples against ML models, as it ensures that the perturbations do not affect human understanding [29, 53, 48]. We conducted a survey, which consists of ten images (adversarial examples) and two multiple choice questions for each image. We perturbed one character for the first five images and two characters for the rest. To avoid any bias, participants are asked whether they can understand the text in an image and only if their answer is yes, then they can choose the correct word. The total number of questions is 35. The survey's link was distributed using Facebook, Twitter, and Amazon Mturk.

Survey Results. The total number of participants was 220, and participants of age 25-34 were the majority, with 46.33%. 55% of the participants were male and 45% female. 85.77% hold a Bachelor or higher degree. We first showed the participants the image and asked them if they can understand the text. Then, if the answer is YES, we asked them to choose the correct word. Participants were able to recognise the text with at least 91% accuracy, which means users' understanding is not affected by our adversarial examples. Also, 56.07% of them have encountered similar images, and 37.11% reported that such images could be found on social media. Moreover, 51.40% of the participants stated that they have used scrambled text when they are writing text.

The results of the survey showed that humans can recognise the embedded perturbed text in images. Also, these kind of images are found in social media more often than in e-mails or SMS. Additionally, most of the participants have seen such images and used scrambled text.

6. Experimental Results and Analysis

We ran the experiments on Linux Ubuntu 18.04 LTS operating system with Intel(R) Core(TM) i7-8750H CPU 2.20GHz x 12 of 983.4 GB memory. Through experiments, our goals were to answer the three research questions that were defined in Section 1.

To answer the research questions, three different experiments were performed. First, we compared the results of five spellcheckers in correcting adversarial text. Second, we launched our black-box attack against four state-of-the-art OCRs. Third, we investigated using our proposed algorithm in OCR-based systems designed for detecting images with malicious contents (i.e., spam, toxic, and offensive).

6.1. Effect of Adversarial Text on Auto-correction

Since we chose to use a spellchecker in our defence method, a natural question arise: can spell checkers detect adversarial text? For this experiment, we used six spellchecking tools to correct/ denoise the same manipulated text (i.e., substitute one or two characters). The tools are listed below:

1. **Peter Norvig** [34]: a spellchecking algorithm that generates candidates within 2 edited distance from the original word, and the highest frequent word is chosen as the correct word.
2. **SymSpell** [6]: a spellchecking tool that uses the Symmetric Delete spelling correction algorithm to reduce the complexity of calculating edit distances and looking up dictionary.
3. **Hunspell** [7]: one of the most popular spellcheckers used by LibreOffice, OpenOffice.org, Mozilla Firefox 3 Thunderbird, and Google Chrome.
4. **Pyspellchecker** [8]: a spellchecker based on Peter Norving's algorithm that uses a Levenshtein Distance algorithm and a list of frequency words to find corrections.
5. **Textblob** [9]: a python library that provides different NLP tasks including spell correction, which uses Peter Norving's algorithm.

Following the methodology used in [2] to test the spellcheckers, we generated four types of adversarial text including our attack using a list of top 20 frequent words in the SMS dataset. We used the DeepWordBug method proposed by Gao et al. [17] to generate three adversarial texts:

[6] https://github.com/wolfgarbe/SymSpell
[7] https://github.com/hunspell/hunspell
[8] https://pypi.org/project/pyspellchecker/
[9] https://github.com/sloria/TextBlob

Table 2
Some of the adversarial text examples used in experiments

Original	Flipping	Swapping	Deletion	Insertion
Busy	Bu$y	Bsuy	Bsy	B*usy
Caller	C@1ler	Claler	Cller	C*aller
Reply	Rep1y	Rpely	Rply	R*eply
winner	w!nner	wniner	wnner	w*inner

(1) insertion: inserted one random character to the words (e.g., c*all), (2) deletion: we removed the second character (one was removed per word), and (3) swapping: we swapped the second and third characters in the word (one swap per word). Also, we used our developed method to generate (4) flipping: we substituted one or two characters with visually similar symbols or numbers. Table 2 presents some of the adversarial text examples used in the experiments. The metric used for evaluating the performance of the spellcheckers is correction accuracy. Figures 7,8,9,10 present the results of the six spellcheckers to correct the four types of adversarial text. The experiments showed that the six spellcheckers preformed the worst when we deleted a character from the words, whereas they preformed the best against the swapping type of adversarial text. The proposed algorithm outperforms the five spellchecking tools in correcting two types of adversarial text attacks: flipping and swapping. Also, it achieved the second best results in correcting deletion and insertion type of adversarial text attacks. Figure 7 shows that Pyspellchecker achieves the best result as it corrects 80% of the adversarial text examples, whereas Hunspell is the worst tool among the five spellchecking tools. Also, our proposed algorithm outperformed the five spellcheckers in correcting the targeted type of adversarial text attack with 20%. Hence, in this experiment, we answered (RQ1) as the results showed that using an auto-correction tool against adversarial attacks is not sufficient. These findings support the results of related studies [27] and [17].

6.2. Effect of Adversarial Text Images on OCRs

In the second part of experiments, we demonstrated the effect of the character-level text attack against the text recognition part of OCR systems. Three benchmark OCRs were used to evaluate the effectiveness of the attack. In the text recognition part of the OCR, two techniques are commonly used for the prediction stage: CTC and Attention-based sequence prediction (Attn). These two techniques have been used in many OCR systems. Several related works used CTC-based OCRs for spam image detection, such as Rosetta [7] and Ads blocking [45]. To evaluate OCRs that use CTC, we chose Rosetta, as it was designed for detecting spam images uploaded to Facebook. Also, *Tesseract* was chosen because it is open-source and publicly available, and is widely used in many OCR-based applications [45]. For Att-based OCR experiment, we used Thin-plate-spline (TPS) based Spatial transformer network (STN)[3]. The TPS achieves the 1st place in ICDAR2013 focused scene text and ICDAR2019 ArT, and 3rd place in ICDAR2017 COCO-Text and ICDAR2019 ReCTS (task1). Two different versions of TPS were used: TPS-NS (non case-sensitive) and TPS-S (case-sensitive). Table 3 shows some examples of images with perturbed embedded text that are misrecognised by TPS.

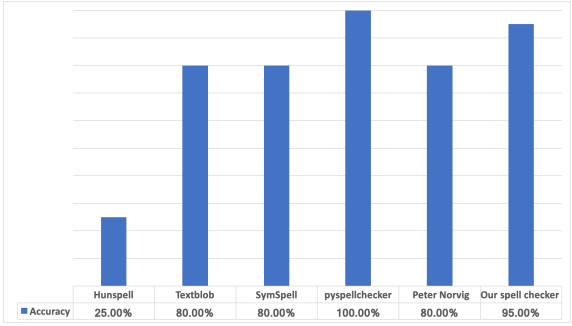

Figure 7: Insertion

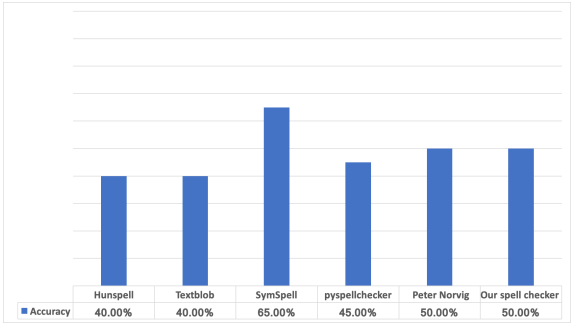

Figure 8: Deletion

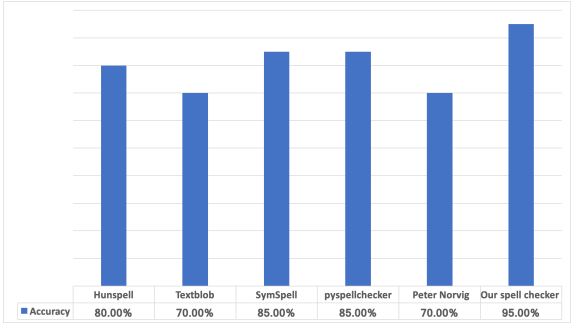

Figure 9: Swapping

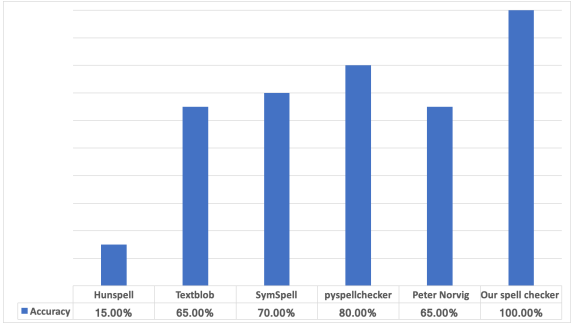

Figure 10: Flipping

Table 3
Examples of images with perturbed embedded text misrecognized by Att-based OCR

Adversarial Examples	Prediction	Confidence
1!ve	ilve	0.65
c@$h	cosh	0.86
C@1ler	coiler	0.85
1@test	10test	0.68

Table 4
Results of the adversarial images with perturbed text against the three OCRs. The best results are highlighted in **bold**

Models	Perturbed Character	CRW	TED
TPS-NS	0	85%	6%
	1	**72%**	12%
	2	**54%**	28%
TPS-S	0	100%	6%
	1	51%	19%
	2	26%	46%
Tesseract	0	100%	100%
	1	21%	33%
	2	15%	60%
Rosetta	0	87%	5%
	1	72%	12%
	2	46%	20%

In this section, we evaluate the text recognition accuracy of the four state-of-the-art OCR systems: *TPS-NS*, *TPS-S*, *Tesseract*, and *Rosetta*. First, we evaluate the recognition accuracy when using images with clean embedded text. Then, we evaluate the OCR systems when one or two characters are perturbed. Measurement metrics used for our experiments were Correctly Recognized Word (CRW) and Total Edited Distance (TED) [24]. CRW is the total number of correctly recognized words by an OCR system, whereas TED is a weighted sum of the Levenshtein distances between the correction of the OCR and the corresponding token in the Ground Truth [37]. The lower the total edited distance, the better. The code adopted for the evaluation was built by [24] for the Incidental Scene Text 2015 competition. The results of our attacks against the four OCRs are presented in Tables 4 and 5. Table 4 presents the results of the four OCRs without using the proposed OCR post-correction algorithm. The results showed that the recognition accuracy of the OCRs dropped significantly when manipulating two characters. These results proved that adversaries can launch a black-box adversarial text attack against OCRs without knowledge about their functionality and parameters (answer to RQ2). The highest recognition accuracy when manipulating two characters was achieved by TPS-NS, while the worst was achieved by Tesseract.

In Table 5, we evaluated the four OCRs using our proposed OCR post-correction algorithm. The results demonstrated that our algorithm improves the recognition accuracy of the OCRs by at least 10%. The best text recognition results archived by TPS-S with 82% and 72% CRW. The results of this OCR system improves by at least 30% when manipulating one character of an input image, and 45% when manipulating two characters (answer to RQ3).

Table 5
Results of the three OCRs after using the proposed post-correction. The best results are highlighted in **bold**

Models	Perturbed Character	CRW	TED
TPS-NS	1	82%	10%
	2	69%	19%
TPS-S	1	**82%**	9%
	2	**82%**	10%
Tesseract	1	72%	17%
	2	69%	25%
Rosetta	1	82%	10%
	2	69%	31%

6.3. An Adversary-aware OCR-based Detector

In the last part of the experiments, we investigate using a Multiple Classifier System (MCS) to design an adversary-aware OCR-based detector that is robust, adaptable and interpretable. As the generated attack can affect the text classification part of OCR-based detector, we evaluate the developed adversary-aware OCR-based detector using an MCS for the text classification task. Additionally, we design this part of the detector to ensure adaptability and interpretability. For aggregating the output of the MCS, there are different methods that could be used, such as linear, non-linear, statistical, and ML combination methods [4]. Kurita et al. [27] used a linear aggregation method that makes its final prediction based on the arithmetic mean of two models' predicted probabilities. Their results showed that the ensembled model outperforms a single classifier when tested on a dataset that includes adversarial text. In the ML aggregation method, a learner algorithm (e.g., DT, K-NN) that learns from the base classifiers' accuracy is applied as a higher level classifier. [51] discussed the potentials of using Multi Kernel Learning (MKL) for combining the output of multiple classifiers, visual and textual. The authors discussed the advantages and disadvantages of aggregating different deep learning based models using kernel learning. Voting rules, which is a non-linear aggregation method, is widely used, and there are several combination rules, such as majority voting, weighted voting, minimum probability, maximum probability, multiplication of probabilities, and average of probabilities [30]. Using such methods adds more complexity to the detector, which makes it non-interpretable and makes it difficult to be adaptable. Hence, we used a Fuzzy Rule Based (FRB) classifier for aggregating the outputs of the three classifiers.

The developed adversary-aware OCR-based detector was designed to extract text from images and to classify the images based on the embedded text. First, we designed the components of the OCR-based detector and then we used it for toxic comments and offensive tweets detection. Figure 11 shows the structure of the developed OCR-based detector. It consists of three steps: OCR system, OCR post-correction,

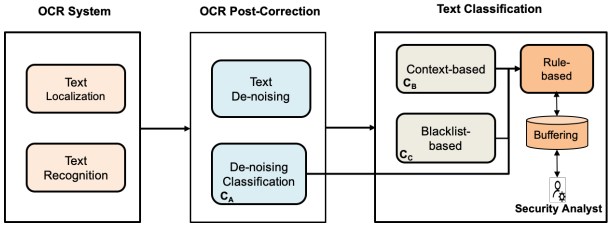

Figure 11: The Structure of the developed OCR-based detectors

and text classification. In the OCR system, we use off-the-shelf tools, PixelLink [10], for localising/ detecting text in images. It is a scene text detector model that has been used in [52]. For the text recognition, we chose the TPS-S (case sensitive) model since it achieves the best results in the above experiments. In the second step of our detector, the proposed algorithm was used for denoising and classifying errors or adversarial text. The last step is text classification, in which we used two text classifiers: context-based and blacklist-based classifiers. We used MCS for three reasons: (1) the output of the OCR system is a list of words, which can affect the accuracy of a context-based classifier, (2) a related study [27] shows that the ensemble model outperforms a model that uses a single classifier, and (3) to ensure that an adversary-aware OCR-based detector can evolve over time in the face of a potential new text manipulation attack.

The context-based classifier relies on the input words order when making its final decision and the blacklist-based classifier, which is a unigram model based on single words, and it is very sensitive to most frequent words. Consequently, we designed our detector to make its final decision based on the output of three classifiers: The results of our proposed algorithm, the context-based classifier and the blacklist-based classifier, which are fed into the FRB part of the text classification step for the final decision. The classifiers were trained in parallel using the same training dataset. The FRB will make its final decision based on the majority voting rules. For example, if one of the text classifiers and the proposed algorithm classification agree on an input X class Y, or if both disagree with the output of the proposed algorithm, then the output of the two classifiers will be used for the final decision. Samples that the classifiers disagree on will be collected and used by a security analyst to update the classifiers. The denoising classification part of the OCR-post correction outputs four type of errors (swapping, flipping, repetition, or OOV). In this study, we consider text that contains a combination of letters and numbers or letters and symbols (i.e., flipping), or a text that contains a letter repeated more than twice (i.e., repetition) as malicious classes. Thus, the output of the denoising classification part of the OCR-post correction will be 1 if flipping or repetition is detected and 0 otherwise.

For generating adversarial text images, we used Algorithm 1. We selected 60 samples from each testing dataset (40 non-toxic and 20 toxic), and manipulated the most frequent words. We only manipulated the toxic and offensive samples of the testing datasets since our threat model assumes that adversaries would only manipulate malicious samples.

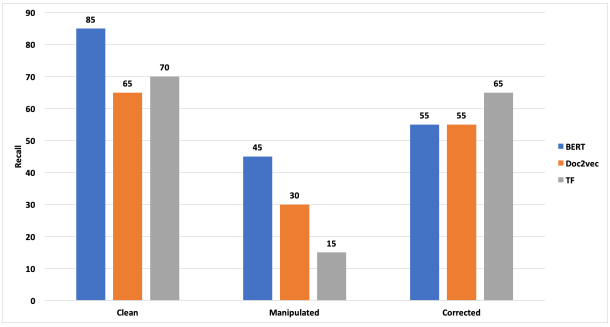

Figure 12: The results of the developed OCR-based detector using three different text classifiers that were evaluated on clean, manipulated, and corrected test datasets

After describing the components of our detector, we now discuss the functionality. First, the OCR system part localises the text in an input image and extracts the detected text. Second, our proposed OCR post-correction corrects (denoises) and classifies the extracted text. The output of the text denoising part of the OCR post-correction is fed into the text classifiers. On the other hand, the output of the denoising classification part of the OCR-post correction, which is a type of error (swapping, flipping, repetition, or OOV), is used along with the output of the text classifiers for the final decision. In the last step, we used the two text classifiers (context-based and blacklist-based), and the FRB for the final decision. In case of a disagreement between classifiers, majority voting will be considered and the samples will be collected for updating the classifiers by the security analyst. In Figure 12, we compared the results of our OCR-based detector using three sota text classifiers. We used BERT, which has been used in related studies and has been shown to have the capability to handle OOV [27]. Also, several studies showed that Doc2vec [28], which captures the meaning within embedding, could detect spam tweets with high accuracy [47]. The third classifier was a simple logistic regression with unigram (LR). All classifiers were trained on clean datasets (i.e., datasets that do not include adversarial text) and evaluated on clean, manipulated and denoised test datasets. Following the evaluation method in [27], the results showed that BERT achieves the best results among the three classifiers. In addition, the results showed that when using our algorithm for denoising the output of the OCR, the recall (i.e., ability of the classifier to detect malicious samples) of the developed detector improved by at least 10%.

In addition, we investigated whether the accuracy of the text classifiers used in the developed adversary-aware OCR-based detector is affected by the output of the OCR, which is a set of words instead of a sentence. As we discussed earlier in Section 1, one of the adversary techniques used to fool OCR-based detectors is to spread the embedded text all over the image (see Figure 2). Thus, to mitigate against such

Table 6
Examples of the original test dataset samples and the OCR's output samples

Original	OCR output
Great. So should i send you my account number	account should number Great i send you my So
Sir Goodmorning, Once free call me	morning Good Once me call Sir free
Where are you call me	Where you me are call
Just now saw your message.it k da:)	message Just your saw now tl age da

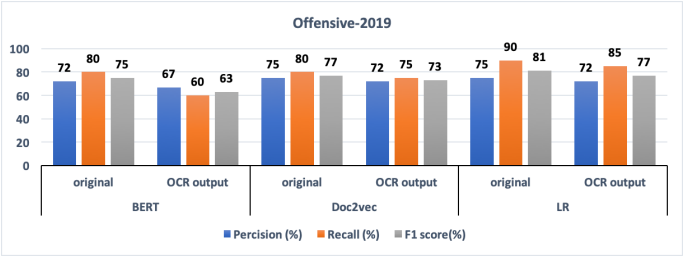

Figure 14: Effect of change in the words order on the text classifiers using the OffensEval-2019 dataset

adversarial activities, extracting a list of embedded words from images could be used. However, since we use a text classifier as part of the OCR-based detector, we compared the results of the three text classifiers using the original version of jigsaw and OffensEval-2019 test datasets and the OCR's output version of the test datasets. Table 6 presents some examples of samples from the SMS test dataset. It shows the original sentences and the output of the OCR. Figures 13 and 14 compared the results of the developed OCR-based detector using the three text classifiers. The comparison was performed using the two test datasets, jigsaw and OffensEval-2019. After training the three text classifiers, we evaluated the classifiers on the original test datasets and the output of the OCR system. The results in Figures 13 and 14 show that the performance of the context-based classifiers (i.e., BERT and Doc2vec) was affected by the change in the words order, whereas it was more stable when using the LR with unigrams. As the LR classifier is sensitive to the most frequent words, we used MCS (BERT and LR) for the text classification task. In the last experiment, we compared the results of the developed adversary-aware OCR-based detector using a single classifier and MCS.

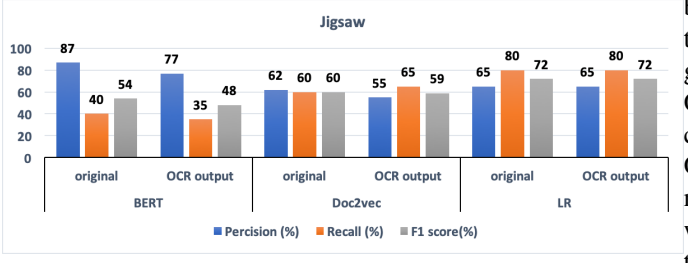

Figure 13: Effect of change in the words order on the text classifiers using the Jigsaw dataset

In Table 7, we make the following comparison: the results of using a single text classifier and MCS with or without our OCR post-correction. We found that the developed OCR-based detector achieves better results when using MCS than when using a single text classifier, in terms of Precision, Recall, and F1-score. In addition, the results of each individual classifier improved when using OCR post-correction. In detail, the results in Table 7 show that the performance (i.e., recall and F1-score) of the three OCR-based detector using two datasets decreased under attack by at least 10%, while they improved by at least 10% when using our algorithm. For example, the recalls of Doc2vec under attack improve when using our algorithm from 30% to 55% and from 35% to 75% on Jigsaw and Offensive-2019 datasets respectively. We considered the recall and F1-score as our threat model assumes that adversaries would only manipulate the malicious samples. These results reflect that fact that the robustness of the OCR-based detectors improve when our algorithm is used. Robustness here refers to the action of decision makers (i.e., text classifiers) through correcting/ denoising the output of the OCR system, and classifying the type of error.

In addition, one of the goals of using MCS is to reduce the false positive rate of the detector. We can see in Table 7 that the precision of the blacklist-based classifier (the LR) is lower than precision of the context-based classifiers (BERT), because it is sensitive to the most frequent words. Thus, to overcome this issue, we used the FRB that considers majority voting of the two classifiers and the proposed algorithm classification of the corrected error. The results show that the precision of the MCS is better than that of both classifiers. Additionally, we conducted our investigation using the OCR-based detector with MCS to deal with evolving attacks and to be interpretable for debugging. The results in Table 7 show that using the MCS of BERT and LR with the proposed algorithm achieved better overall performance. The proposed OCR post-correction is designed to denoise errors and to classify the type of errors. When the error is classified as OOV by the proposed algorithm, but becomes classified as non-malicious by any of the text classifiers, this type of error will be used as an indicator of a new possible attack. Hence, these examples will be collected and checked by the security analyst to confirm if the detector needs to be updated.

7. Discussion

The experiments performed in this study showed how the proposed algorithm can improve the robustness of OCR-based systems to images with embedded manipulated text. Denoising the output of the OCR systems could be helpful in the classification task. Most importantly, classifying the type of errors helps detect adversarial examples. Although we focused on a particular adversarial attack, where an adversary replaces characters with similar looking symbols or numbers, our algorithm could be modified to detect different

Table 7
Detailed results of the developed adversary-aware OCR-based detector on two datasets and three test datasets: clean, manipulated, and denoised datasets. The best results are highlighted in **bold**. The MCS (BERT and LR) outperforms using a single text classifier

Model	Testing Data	Jigsaw			Offensive-2019		
		Percision	Recall	F1	Percision	Recall	F1
BERT	clean	77	35	48	67	60	63
Doc2vec		55	65	59	72	75	73
LR		65	80	72	72	85	77
BERT	manipulated	77	25	38	67	15	24
Doc2vec		55	30	39	72	35	47
LR		65	15	24	72	20	31
BERT	Denoised	77	55	64	67	80	83
Doc2vec		55	55	55	72	75	73
LR		65	75	51	72	85	78
MCS		87	85	86	82	85	83

adversarial attacks. For example, it could be modified to detect an adversarial text attack, in which adversaries replace some characters in important words with asterisks (e.g., st**id). This has been shown in the first experiment (see Section 6). Also, the experiments show that the proposed algorithm can be used for improving the robustness of NLP applications. The results in Subsection 6.1 show that our algorithm outperforms five spellcheckers.

Generally speaking, when designing an automated ML-based model in an adversarial environment, very important points that need to be taken into account at the design stage are the adaptability to evolving attacks and interpretability to security analysts. In other words, we need our model to evolve over time in the face of new attacks. The proposed OCR post-correction has been designed to detect OOV, which could be used as an indicator of a new attack. For instance, an increased number of OOV in images with non-malicious content requires debugging the model to investigate if there is a new attack. Additionally, in this paper, we show that the proposed algorithm can be used with any OCR-based detector designed for adversarial environments. For example, we investigate improving the robustness of the Facebook spam image detector *Roseeta* using our OCR post-correction algorithm.

There are some limitations that future research needs to consider. In order for our OCR post-correction to be effective, the OCR system needs to accurately recognise the embedded text in images. The recognition accuracy of OCRs is affected by noise and adversarial perturbations. This is especially true for images uploaded onto OSNs, which are images with text overlaid on top of them; extracting text from such images with high accuracy is an area that requires more research. Also, the robustness of the OCR-based systems against adversarial image attacks (e.g., FGSM or Adversarial Watermarks) needs to be considered. Additionally, one of the drawbacks of OCR systems when used for detecting spam images is that the embedded text gets extracted as a list of words, which affects the text classification. In this study, we used an MCS with the FRB classifier to tackle this issue. However, more research to find a method for re-ordering the extracted words needs to be conducted.

8. Conclusion

Using an OCR in cybersecurity systems is an important and challenging problem. In this paper, we investigated the robustness of OCR-based systems against images with embedded adversarial text (i.e., adversarial text images). Similar to other application domains (e.g., NLP), we found that state-of-the-art OCRs are vulnerable to such slightly manipulated embedded text that does not affect human understanding. We described our proposed OCR post-correction, which denoises and classifies various types of errors to improve the robustness of OCR-based systems against character-level adversarial attacks. Our algorithm improves the text recognition of state-of-the-art OCRs by over 10%. Additionally, as a case study, we showed how our algorithm could improve the overall performance of OCR-based systems designed to detect images with embedded malicious content (e.g., spam, toxic or offensive comments). The developed adversary-aware OCR-based detector consists of an OCR and an MCS text classification model along with the proposed algorithm. As demonstrated by our experiments, our algorithm improves the accuracy of the detector by over 20%. From a security point of view, utilising an OCR post-correction not only provides robustness against adversarial examples, but it also provides adaptability and interpretability, which helps detect unknown attacks.

The investigation of OCR-based detectors in this paper reveals that there are some limitations that future research needs to consider. Specifically, the extracted text from images needs to be checked before it is fed into a text classification part of an OCR-based detector. Although the recognition accuracy of OCRs has been improving over the last few years in document classification tasks, this accuracy drops with noisy images. Thus, improving the text recognition accuracy and robustness of OCRs in an adversarial setting is an area that requires more research. Additionally, another area that future research needs to focus on is the word order of the extracted text from an image. One of the drawbacks of OCRs when used for detecting spam images is that the embedded text gets extracted as a list of words, which affects the text classification. Hence, a method for re-ordering the extracted words needs to be researched further.

References

[1] 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*, 2(1):1–18, 2013.

[2] B. Alshemali and J. Kalita. Toward mitigating adversarial texts. *International Journal of Computer Applications*, 178(50):1–7, 2019.

[3] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4715–4723, 2019.

[4] S. Barak, A. Arjmand, and S. Ortobelli. Fusion of multiple diverse predictors in stock market. *Information Fusion*, 36:90–102, 2017.

[5] Y. Belinkov and Y. Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*, 2018.

[6] F. Borisyuk, A. Gordo, and V. Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79. ACM, July 2018.

[7] F. Borisyuk, A. Gordo, and V. Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79, 2018.

[8] F. Boschetti, M. Romanello, A. Babeu, D. Bamman, and G. Crane. Improving OCR accuracy for classical critical editions. In *International Conference on Theory and Practice of Digital Libraries*, pages 156–167. Springer, 2009.

[9] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[10] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, 2019.

[12] J. Ebrahimi, D. Lowd, and D. Dou. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, 2018.

[13] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2)*, pages 31–36, 2018.

[14] S. Eger and Y. Benz. From hero to zéroe: A benchmark of low-level adversarial attacks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 786–803, 2020.

[15] S. Eger, G. G. Şahin, A. Rücklé, J.-U. Lee, C. Schulz, M. Mesgar, K. Swarnkar, E. Simpson, and I. Gurevych. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[16] S. Eger, T. vor der Brück, and A. Mehler. A comparison of four character-level string-to-string translation models for (OCR) spelling error correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77, 2016.

[17] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[18] Z. Gong, W. Wang, B. Li, D. Song, and W.-S. Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[20] G. Heigold, S. Varanasi, G. Neumann, and J. van Genabith. How robust are character-based word embeddings in tagging and mt against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 68–80, 2018.

[21] N. Imam. Synthetic datasets of adversarial images. https://data.mendeley.com/datasets/2g3c836mh3/1, 2021.

[22] N. Imam and V. Vassilakis. Detecting spam images with embedded Arabic text in Twitter. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 6, pages 1–6. IEEE, 2019.

[23] N. H. Imam and V. G. Vassilakis. A survey of attacks against Twitter spam detectors in an adversarial environment. *Robotics*, 8(3):50, 2019.

[24] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. ICDAR 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.

[25] Y. Keller, J. Mackensen, and S. Eger. BERT-Defense: A probabilistic model based on BERT to combat cognitively inspired orthographic adversarial attacks. *arXiv preprint arXiv:2106.01452*, 2021.

[26] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017.

[27] K. Kurita, A. Belova, and A. Anastasopoulos. Towards robust toxic content classification. *arXiv preprint arXiv:1912.06872*, 2019.

[28] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[29] J. Li, S. Ji, T. Du, B. Li, and T. Wang. TextBugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*, 2019.

[30] O. Matan. On voting ensembles of classifiers. In *Proceedings of AAAI-96 workshop on integrating multiple learned models*, pages 84–88. Citeseer, 1996.

[31] T. Miyato, A. M. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

[32] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[33] N. Mor and L. Wolf. Confidence prediction for lexicon-free OCR. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 218–225. IEEE, 2018.

[34] P. Norvig. How to write a spelling corrector. *Online at: http://norvig.com/spell-correct. html*, 2007.

[35] A. Poncelas, M. Aboomar, J. Buts, J. Hadley, and A. Way. A tool for facilitating OCR postediting in historical documents. In *Proceedings of LT4HALA 2020-1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 47–51, 2020.

[36] D. Pruthi, B. Dhingra, and Z. C. Lipton. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, 2019.

[37] C. Rigaud, A. Doucet, M. Coustaty, and J.-P. Moreux. ICDAR 2019 competition on post-OCR text correction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1588–1593. IEEE, 2019.

[38] S. Rojas-Galeano. On obstructing obscenity obfuscation. *ACM Transactions on the Web (TWEB)*, 11(2):1–24, 2017.

[39] R. Schuster, T. Schuster, Y. Meri, and V. Shmatikov. Humpty dumpty: Controlling word meanings via corpus poisoning. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1295–1313. IEEE, 2020.

[40] T. S. Sethi and M. Kantardzic. Handling adversarial concept drift in streaming data. *Expert systems with applications*, 97:18–40, 2018.

[41] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[42] C. Song and V. Shmatikov. Fooling ocr systems with adversarial text images. *arXiv preprint arXiv:1802.05385*, 2018.

[43] K. Taghva and E. Stofsky. OCRSpell: an interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition*, 3(3):125–137, 2001.

[44] P. Thompson, J. McNaught, and S. Ananiadou. Customised OCR correction for historical medical text. In *2015 Digital Heritage*, volume 1, pages 35–42. IEEE, 2015.

[45] F. Tramèr, P. Dupré, G. Rusak, G. Pellegrino, and D. Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2005–2021, 2019.

[46] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[47] C. VanDam, F. Masrour, P.-N. Tan, and T. Wilson. You have been caute! early detection of compromised accounts on social media. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 25–32. IEEE, 2019.

[48] P. Vijayaraghavan and D. Roy. Generating black-box adversarial examples for text classifiers using a deep reinforced model. In U. Brefeld, E. Fromont, A. Hotho, A. Knobbe, M. Maathuis, and C. Robardet, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 711–726, Cham, 2020. Springer International Publishing.

[49] M. Volk, L. Furrer, and R. Sennrich. Strategies for reducing and correcting OCR errors. In *Language technology for cultural heritage*, pages 3–22. Springer, 2011.

[50] B. Wang, H. Pei, H. Liu, and B. Li. Advcodec: Towards a unified framework for adversarial text generation. *arXiv preprint arXiv:1912.10375*, 2019.

[51] T. Wang, L. Zhang, and W. Hu. Bridging deep and multiple kernel learning: A review. *Information Fusion*, 2020.

[52] K. Yuan, D. Tang, X. Liao, X. Wang, X. Feng, Y. Chen, M. Sun, H. Lu, and K. Zhang. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *2019 IEEE Symposium on Security and Privacy*. IEEE, 2019.

[53] M. Zha, G. Meng, C. Lin, Z. Zhou, and K. Chen. RoLMA: A practical adversarial attack against deep learning-based LPR systems. In *International Conference on Information Security and Cryptology*, pages 101–117. Springer, 2019.

[54] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

[55] G. Zuccon, A. N. Nguyen, A. Bergheim, S. Wickman, and N. Grayson. The impact of OCR accuracy on automated cancer classification of pathology reports. In *Health Informatics: Building a Healthcare Future Through Trusted Information*, pages 250–256. IOS Press, 2012.