Metadata-Based Detection of Child Sexual Abuse Material

Mayana Pereira, Rahul Dodhia, and Richard Brown

***Abstract*—In the last decade, the scale of creation and distribution of child sexual abuse medias (CSAM) has exponentially increased. Technologies that aid law enforcement agencies worldwide to identify such crimes rapidly can potentially result in the mitigation of child victimization, and the apprehending of offenders. Machine learning presents the potential to help law enforcement rapidly identify such material, and even block such content from being distributed digitally. However, collecting and storing CSAM files to train machine learning models has many ethical and legal constraints, creating a barrier to the development of accurate computer vision-based models. With such restrictions in place, the development of accurate machine learning classifiers for CSAM identification based on file metadata becomes crucial.**

In this work, we propose a system for CSAM identification on file storage systems based solely on metadata - file paths. Our aim is to provide a tool that is material type agnostic (image, video, PDF), and can potentially scans thousands of file storage systems in a short time. Our approach uses convolutional neural networks, and achieves an accuracy of 97% and recall of 94%. Additionally, we address the potential problem of offenders trying to evade detection by this model by evaluating the robustness of our model against adversarial modifications in the file paths.

***Index Terms*—child sexual abuse medias; child exploitation; CSAM; file path; file storage systems; cloud storage; machine learning; neural networks.**

I. Introduction

INTERNATIONAL law enforcement handle millions of child sexual abuse cases annually. Hotlines internationally have received and reviewed over 37 million child sexual abuse files in 2017, alone [5]. Despite the 2008 *Protect our children act* [20], the numbers of CSAM in digital platforms have dramatically grown in the last decade[30]. Online sharing platforms have facilitated[24] the explosive growth of CSAM creation and distribution [6]. Every platform for content searching and sharing, including social material, likely has CSAM on it [15]. This is an issue that affects society in larger numbers than expected. Recent studies reported alarming statistics, where 25% of girls and 17% of boys in the U.S. will experience some form of sexual abuse before the age of 18 [27].

As the scale of the problem grew, technology became essential in CSAM identification. Technology giants such as Microsoft, Apple, Facebook, and Google have made detection and removal of CSAM one of their top concerns, and work with non-profits such as Project VIC International [1], Thorn [2] and the Internet Watch Foundation[3] to create tools to combat CSAM proliferation.

With the COVID-19 pandemic, experts have observed that the distribution of CSAM in social material and video conference apps has significantly increased [25]. Distributors use coded language to trade links of CSAM hosted in plain sight on content sharing platforms such as Facebook, YouTube, Twitter, and Instagram using cryptic language to evade the current detection tools [6], [25].

A. Preventing Distribution of Undiscovered Material.

The identification of CSAM is an extremely challenging problem. It starts with the fact that it can manifest in different types of material: images, videos, streaming, video conference, online gaming, among others. CSAM still undiscovered and unlabeled on the internet is potentially several magnitudes greater than previously identified CSAM. New material is created daily and digital platforms have a large role to play in the detection and and removal of this material.

Multiple approaches to the CSAM identification problem using computer vision-based methods have appeared in the literature in the past years [28], [16], [32]. Although these approaches appear promising, they lack evaluation in real-world big data unbalanced data sets; data sets with ethnicity diversity, and time efficiency and scalability evaluation. To keep up with all new forms of CSAM content distribution [6], all these factors must be carefully evaluated and taken into consideration.

B. Our Contributions

Given the complexity of the problem of detecting CSAM, it is essential to analyze different aspects of the CSAM distribution. This paper proposes statistical models that compute the likelihood that a file path is associated with a CSAM file. An advantage of this approach is that it does not require possession of the raw CSA image or video for training machine learning models. Specifically, we accomplish the following tasks:

- Train and compare several machine-learning based models that analyze metadata from file storage systems and determine a probability that a given file has child sexual abuse content.

M. Pereira is with Microsoft Corporation, AI for Good Research Lab, Redmond, WA, 98052 and also with the Universidade de Brasília, Campus Darcy Ribeiro, Brasília, Brazil (e-mail: mayana.wanderley@microsoft.com).

R. Dodhia is with Microsoft Corporation, AI for Good Research Lab, Redmond, WA, 98052.

R. Brown is with Project VIC International, Neptune City, NJ, 07753.

[1]https://www.projectvic.org
[2]https://www.thorn.org
[3]https://www.iwf.org.uk

- Analyze the robustness of our models against commonly used techniques for avoiding detection in text-based machine learning classifiers. We study the resilience of our models against black-box attackers (attackers that have no knowledge of the models and their parameters) and against attackers that have knowledge about words that have a high probability to appear in CSAM file paths.
- We train our models on a real-world data set containing over one million file paths. It is the largest file path data set ever used for CSAM detection to date. Our classifiers achieve recall rates over 94% and accuracy over 97%.

Our results demonstrate, for the first time, the practicality of using file paths for CSAM detection in storage systems.

II. Related Work

A. PhotoDNA Hash

Identification of CSAM via statistical algorithms is a fairly recent approach. At the turn of the century, laws targeting online exploitation were introduced and refined (COPA in the US, Crime and Disorder Act UK)[7], but the first widely used methodology was released in 2008. Known widely as PhotoDNA[17], the technology was developed by Dr. Hany Farid and Microsoft.

In PhotoDNA, an image is converted to a long string of characters by a fuzzy hash algorithm, and this hash could then be compared against other hashes to find identical or similar images without people having to view the images and compromising the victim's identity. This system is still one of the most widely used methods for detecting images worldwide, databases of hashed CSAM images are used to identify identical or similar images on many digital platforms, including search engines, social networks, and file storage platforms. The advantage of PhotoDNA Hash is that it has a low false-positive detection rate, but it has some clear drawbacks. It only works on known CSAM that has gone through its hashing algorithm, and identification of new CSAM is still primarily a manual labeling endeavor. Labelers have to be trained and wellness programs have to be developed to counter the psychological impact of viewing these images.

B. Machine Learning for Image Identification

Since PhotoDNA's first development, computer vision models have undergone a revolution resulting in novel machine learning based models for pornography and CSAM detection [19], [16], [32], [22]. The current approaches either combine a computer vision model to extract image descriptors [28], train computer vision models on pornography data [9], perform a combination of age estimation and pornography detection [16] or synthetic data [32]. However, due to legal restrictions in maintaining a database of CSAM images, all current works are based on either unrealistic images [32], or validated by authorities in very small [28], [16], [9] data sets that hardly represents the true data distribution in the internet [6].

C. Machine Learning Based on Complementary Signals and Metadata

While significant efforts have focused on the images themselves, some researchers have looked for complementary signals and metadata that could help in CSAM identification. For example, queries that bring up CSAM in search engines, or conversations that imply grooming or trade in CSAM[26]. Other efforts have used textual signals to identify where CSAM might be located, such as keywords related to website content[29], using NLP analysis[23], [21], [2], conversations[4]. Our work falls into this category.

D. File Path classification

To the best of our knowledge, there is only a single paper that describes an attempt to identify CSAM based solely on file paths [2]. While pioneering, this work was based on a small data set containing only 9,000 CSAM file paths, and their best model achieves recall rates lower than 80% for CSAM file paths. High recall rate is critical for this application. Additionally, the work in [2] does not address the question of classifiers robustness against noise introduced at test time (adversarial attacks).

E. Robustness Against Adversarial Attacks

Whenever a machine learning classifier is used for detecting malicious behavior or content, it is important to consider the possibility that inputs to the model have been modified in an adversarial way. Adversarial inputs to a classifier are inputs that have been modified in an intentional way to make detection of malicious activity harder. An adversary can intentionally flip characters of words that most likely contain signal for helping in the classification task, e.g. using Lo7ita instead of Lolita.

Our proposed techniques are based on classification of file paths. Thus, it is important to study the effects of adversarial inputs in the classifier performance.

Such a question is not new in natural language processing and the effect of random modifications to a classifier's inputs have been previously studied in the context of text classification [1], machine translation [3] and word embedding [11].

In this paper, we study the robustness of our model in two different situations. One where such an adversary has no knowledge of model's parameters [8], or even the score output generated at test time [12]. We call this a black-box adversary.

Additionally, we also present an analysis where the adversary has some knowledge of the models and uses it to target regions of the file path that most likely contain signals for classification.

III. Data set Description

Our approach to identifying CSAM file paths is based on supervised learning. Thus, it is necessary to build a pre-labelled data set (CSA versus non-CSA). Moreover, we also need separating the data set according to storage system precedence. This is done to guarantee independence of samples in train and test time.

Our data set was provided by Project VIC International and consists of 1,010,000 file paths from 55,312 unique storage systems. The data extraction from all storage systems utilizes a forensics technique known as file carving. File carving recovers files by scanning the raw bytes of a storage system and reassembling them. This process comprises examining the header (the first few bytes) and footer (the last few bytes) of a file. File carving is utilized for recovering files and fragments of files when directory entries are corrupt or missing. Forensics experts use this in criminal cases for recovering evidence. Particularly in CSA cases, law enforcement agents can often recover more images from the suspect's storage system using carving techniques.

The data set includes the following columns:

- **File path:** This column contains the entire file path for a given file. File paths are strings that contains location information of a file (folders) in a storage system, as well as the file name. Examples of file paths in our data set:
  C:\Users\SomeUser\Pictures\Seattle.png
  D:\ExternalHD\Files \Pictures\Seattle.png
- **Label:** The data set provided by Project VIC contains 4 different labels to identify the type of content in the file. The labels are defined as:
  [0] Non-pertinent
  [1] Child Abuse Material (CSAM)
  [2] Child Exploitive/Age Difficult
  [3] CGI/Animation Child exploitive

TABLE I
QUANTITY OF FILE PATHS IN EACH LABEL CLASS

Label	Number of file paths
0	717,448
1	33,901
2	250,724
3	7,927

Our system aims to identify all classes of CSA-related material. For the purposes of model training and model evaluation, we created binary labels (CSAM vs non-CSAM). Labels 1,2 and 3 are mapped to CSAM (292,552 file paths); label 0 is mapped to non-CSAM (717,448 file paths).

A. File Path Characteristics

We describe file path characteristics that will help us define technical aspects of our classification models such as the size of the character embedding vectors in our deep neural networks models.

Figure1 shows the distribution of file path lengths in the data set. Only 4,685 file paths have more than 300 characters. For this reason, we truncate the file path by discarding the initial characters and keeping only the last 300 characters for training, validation and testing.

The majority of the data set presents English terms and words. Using the language identification library Fasttext [14], we identified words in over 10 languages, including Russian, German, Swedish, Spanish, Polish, Italian, Japanese, Chinese and Portuguese. The variety of languages and alphabets used in the file paths define the size of the character embedding layer in our models.

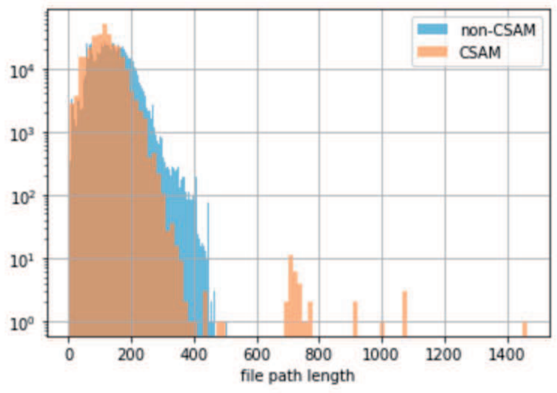


Fig. 1. Number of characters in each file path

B. Cross Validation Data Split

We use a K-Fold Cross Validation methodology in our experiments with K=10. To guarantee independence of file paths in the different partitions of the data, we create the random data folds by splitting the data by storage system information, as illustrated in Fig 2.

The storage system identification for each file path is done as follows. We check if the information before the first backlash corresponds to external storage system or a laptop/desktop file system. If we are unable to extract the identifying factors before the first backlash, we extract the information of the first two backlashes and use it as the storage system identifier.

IV. File Path-Based CSAM Classifiers

We investigated three approaches for CSAM file path classification.

- The first approach is based on extracting bag-of-words from the file paths following by a traditional supervised machine learning classifier (logistic regression, boosted decision trees, and naive Bayes);
- The second approach is based on computing n-grams from each file path and presenting these n-gram vectors to a traditional supervised machine learning classifier

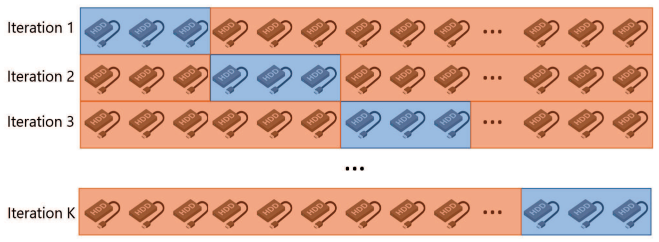


Fig. 2. Data split for K-fold cross validation. The data is partitioned by storage system ids. File paths from a same storage system are all assigned to a same data fold. The data folds in the orange shaded areas represent the training data in each iteration, and the data folds in the blue shaded areas represent the test data.

(logistic regression, boosted decision trees, and naive Bayes);

- The third approach is based on character-based deep learning models. This approach uses a character quantization of each file path and presents it to a deep learning model architecture.

We now describe how the input data (file path) is represented in each one of these approaches and the deep learning architectures that were used.

A. Data Input Representation

a) Bag-of-Words and TF-IDF: For each file path, we consider a "word" to be a sequence of alphanumeric characters that are separated by a dash, slash, colon, underscore or period. The bag-of-words model is constructed by selecting the 5,000 most frequent "words" from the training subset. For the standard bag-of-words, we use the counts of each word as the features. In the TF-IDF (term-frequency inverse-document-frequency) [13] version, the term-frequency (TF) is the number of times a term occurs in a given document. The inverse-document frequency-component (IDF) is computed as:

$$\mathrm{IDF}(t) = \log \frac{1+n}{1+df(t)} + 1$$

where n is the total number of documents in the document set, and $df(t)$ is the number of documents in the document set that contain term. For each term it is computed the product of the TF and IDF components. The resulting TF-IDF vectors are then normalized by the Euclidean norm. The data set of vectorized file paths is used as input to three different learning algortihms: logistic regression, naive Bayes and boosted decision trees.

b) Bag-of-Ngrams and TF-IDF: We extract from each file path string its n-grams, for $n \in \{1, 2, 3\}$. The set of n-grams of a string s, is the set of all substrings of s of length n. The bag-of-ngrams models are constructed by selecting the 50,000 most frequent n-grams (up to 3-grams) from the training subset for each data set. The feature values are computed the same way as in the bag-of-words model. The data set of vectorized file path n-grams is used as input to three different learning algortihms: logistic regression, naive Bayes and boosted decision trees.

c) Character-Based Quantization: As input to the deep neural networks models, we use sequences of encoded characters. We use the same approach proposed in [33]. An alphabet of size m is defined as the input language, which is quantized using 1-of-m encoding. Each file path is then transformed into a sequence of such m sized vectors with fixed length 300, and any characters exceeding length 300 is ignored. If the file path name is shorter than 300, we pad with zeroes on the left. In our data sets, over 99% of the file paths have length less than 300 characters, as shown in Figure 1.

Characters that are not in the alphabet are quantized as all-zero vectors. The alphabet used in all of our models consists of m = 802 characters, which includes English letters, Japanese characters, Chinese characters, Korean characters and special alphanumeric characters. The alphabet is the set of all unique characters in the training data.

All deep neural network architectures start with an embedding layer that learns to represent each character by numerical vector. The embedding maps semantically similar characters to similar vectors, where the notion of similarity is automatically learned based on the classification task at hand.

B. Deep Learning Architectures

Deep Neural Networks in combination with character embeddings have succeeded in several short text and string classification tasks. From natural language text [33] to domain classification [31], deep neural networks architectures have outperformed traditional methods in several applications.

In our work, we use two architectures of deep neural networks: convolutional networks and long-short term memory networks.

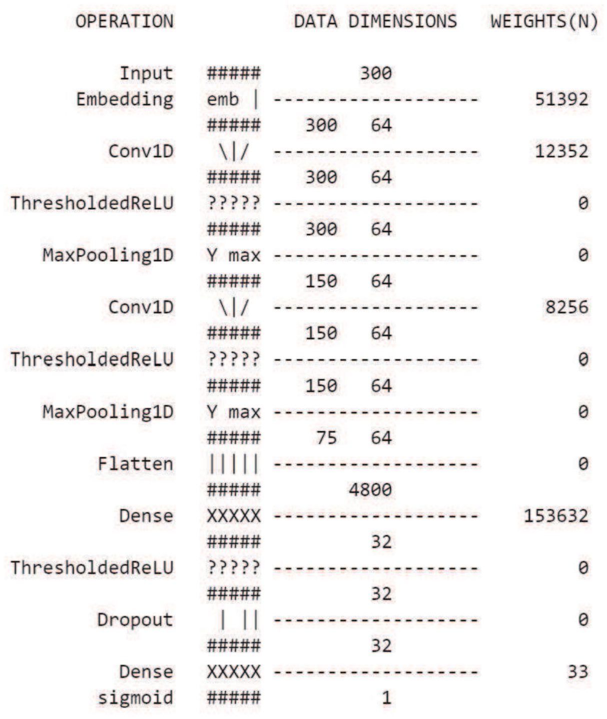
a) Convolutional Neural Networks: Convolutional Neural Networks (CNN) are known for state-of-the-art advances in image processing, and apply to inputs of grid-like topology. One-dimensional CNNs are a natural fit when the input is text, treated as a raw signal at the character level [33]. CNNs automatically learn filters to detect patterns that are important for prediction. The presence (or lack) of these patterns is then used by the quintessential neural network (multilayer perceptron, or MLP) to make predictions. These filters, (also called kernels) are learned during backpropagation. An intuitive example in image processing is a filter which detects vertical edges, while in text processing the filters detect sub-strings, or n-grams. The underlying operation of CNNs is elementwise multiplication, performed between each filter and sub-sections of the input. The resulting values indicate the degree to which these sub-sections match the filters. In this manner, the filters are convoluted over the input to form an activation map, which represents the locations of discovered features.

b) Long Short-Term Memory Networks: We train an depp neural network architecture that uses a recurrent neural network layer, namely long-short term memory (LSTM) networks. The variant of LSTM used is the common "vanilla" architecture as used in [31].

LSTM is a type of recurrent neural network that can capture combinations of letters. This characteristic can be critical to discriminating CSA file paths from non-CSA file paths. This flexible architecture generalizes manual feature extraction via n-grams, for example, but instead learns dependencies of one or multiple characters, whether in succession or with arbitrary separation.

The LSTM layer can be thought of as an implicit feature extraction, as opposed to explicit feature extraction (e.g., n-grams) used in other approaches. Rather than represent file paths names explicitly as a bag of n-grams, for example, the LSTM learns patterns of characters (or in our case, embedded vectors) that maximize the performance of the second classification layer.

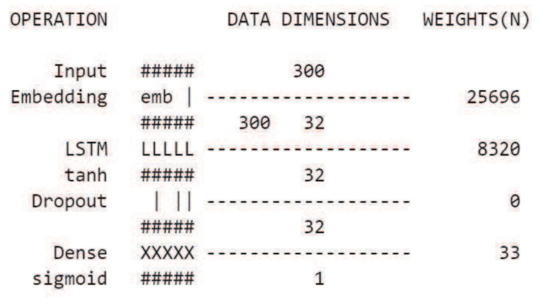
c) Implementation Details: Figures 3 and 4 show detailed information of both architectures(charCNN and charLSTM), including data dimensions and the number of weights in each layer.



```
      OPERATION           DATA DIMENSIONS   WEIGHTS(N)

          Input   #####         300
      Embedding   emb | -------------------     51392
                  #####    300   64
         Conv1D    \|/  -------------------     12352
                  #####    300   64
ThresholdedReLU   ????? -------------------         0
                  #####    300   64
   MaxPooling1D   Y max -------------------         0
                  #####    150   64
         Conv1D    \|/  -------------------      8256
                  #####    150   64
ThresholdedReLU   ????? -------------------         0
                  #####    150   64
   MaxPooling1D   Y max -------------------         0
                  #####     75   64
        Flatten   ||||| -------------------         0
                  #####        4800
          Dense   XXXXX -------------------    153632
                  #####         32
ThresholdedReLU   ????? -------------------         0
                  #####         32
        Dropout    | || -------------------         0
                  #####         32
          Dense   XXXXX -------------------        33
        sigmoid   #####          1
```

Fig. 3. Diagram of the deep neural network architecture with CNN layers used for training one of our charCNN model. All data dimensions and number of weights in each layer of our charCNN model are indicated in the above diagram.



```
 OPERATION           DATA DIMENSIONS   WEIGHTS(N)

     Input   #####         300
 Embedding   emb | -------------------     25696
             #####    300   32
      LSTM   LLLLL -------------------      8320
      tanh   #####         32
   Dropout    | || -------------------         0
             #####         32
     Dense   XXXXX -------------------        33
   sigmoid   #####          1
```

Fig. 4. Diagram of the deep neural network architecture with LSTM layer used for training one of our charLSTM model. All data dimensions and number of weights in each layer of our charLSTM model are indicated in the above diagram.

V. Evaluation

We present our results for all our classifiers in Table II. All performance metrics were measure using a 10-fold cross-validation methodology. For each of our classifiers, we report the mean and the standard deviation of the area under the ROC curve (AUC), accuracy, precision, and recall for predicting CSAM files. We focus on two primary metrics for model comparison: Recall and AUC. Additionally, we assess all machine learning models' generalization by looking into the standard deviations over the cross-validation folds.

A. Traditional Machine Learning

There are significant advantages of traditional machine learning models in comparison with deep neural networks. Understanding how well these models perform can help scientists and investigators leverage one of such models' most remarkable characteristics: feature interpretability. The most relevant predictive tokens, or n-grams, can give clues about relevant vocabulary words in the data set and be leveraged in other CSAM detection systems. On table II, we observe that the model trained with bag-of-words and bag-of-ngrams operates in similar AUC and accuracy ranges. When analyzing recall rates of traditional models, we note that both naive Bayes models have the highest rates, with the lowest standard deviation. The naive Bayes with bag-of-ngrams features presents the best recall of all traditional models, of about 91%. Among the other models trained using bag-of-ngrams, naive Bayes presents a much smaller recall standard deviation (σ = 0.085) when compared to logistic regression (σ = 0.20) and boosted decision trees(σ = 0.20).

In figure 5, we compare side by side the ROC curves of all the models. The ROC curve illustrates the operation of a binary classifier as its discrimination thresholds vary. Although the evaluation of CSAM classification models heavily relies on recall rates, when deployment a model in an environment that potentially analyses hundreds on thousands of file systems, and consequently millions of file paths, precision rates must be analyzed. The burden of having several thousands of false positives can result in an inefficient process and potentially delay investigations and discovery of true positives. By analyzing the traditional models' ROC curve, we observe that boosted decision trees overall performs better than the two other techniques.

B. Deep Neural Networks

We were able to achieve the best performance across all categories with deep neural network architecture. We trained two different architectures: our first architecture utilizes CNNs and the second LSTMs. The LSTM model achieves results very similar to the bag-of-words naive Bayes classifier. However, our CNN model consistently outperformed all the other models, both in mean performance metrics across all folds and in the smallest standard deviation. The resulting ROC curve also indicates a model that operates with a higher recall for multiple ranges of false-positive rates. The recall of over 94% and precision over 93% makes this model an excellent candidate as an investigative tool in environments with large volumes of storage systems.

VI. Model Robustness to Adversarial Inputs

In classical machine learning applications, we assume that the underlying data distribution is stationary at test time. However, there are multiple scenarios where an intelligent, adaptive adversary can actively manipulate samples.

In the scenario of CSAM file path identification, it is easy to imagine a scenario where perpetrators include typos and modifications to the file paths to evade vocabulary blacklists and even machine learning-based CSAM detection mechanisms. We address this problem by simulating attack scenarios where an adversary actively changes the file paths to elude the classifiers.

TABLE II
PERFORMANCE METRICS FOR VARIOUS MODELS

Features	Algorithm	AUC		Accuracy		Precision		Recall	
		Mean	σ	Mean	σ	Mean	σ	Mean	σ
Bag of Words	Logistic Regression	0.967	0.035	0.922	0.062	0.904	0.090	0.787	0.202
Bag of Words	Naive Bayes	0.972	0.011	0.927	0.032	0.875	0.070	0.859	0.114
Bag of Words	Boosted Trees	0.982	0.013	0.934	0.062	0.903	0.096	0.827	0.203
Bag of N-grams	Logistic Regression	0.980	0.021	0.931	0.060	0.919	0.088	0.793	0.202
Bag of N-grams	Naive Bayes	0.958	0.023	0.929	0.032	0.839	0.083	0.913	0.085
Bag of N-grams	Boosted Trees	0.983	0.015	0.931	0.060	0.906	0.094	0.822	0.203
Character quantization	CNN	**0.990**	0.011	**0.968**	0.019	**0.938**	0.034	**0.943**	0.060
Character quantization	LSTM	0.977	0.029	0.930	0.072	0.862	0.144	0.846	0.213

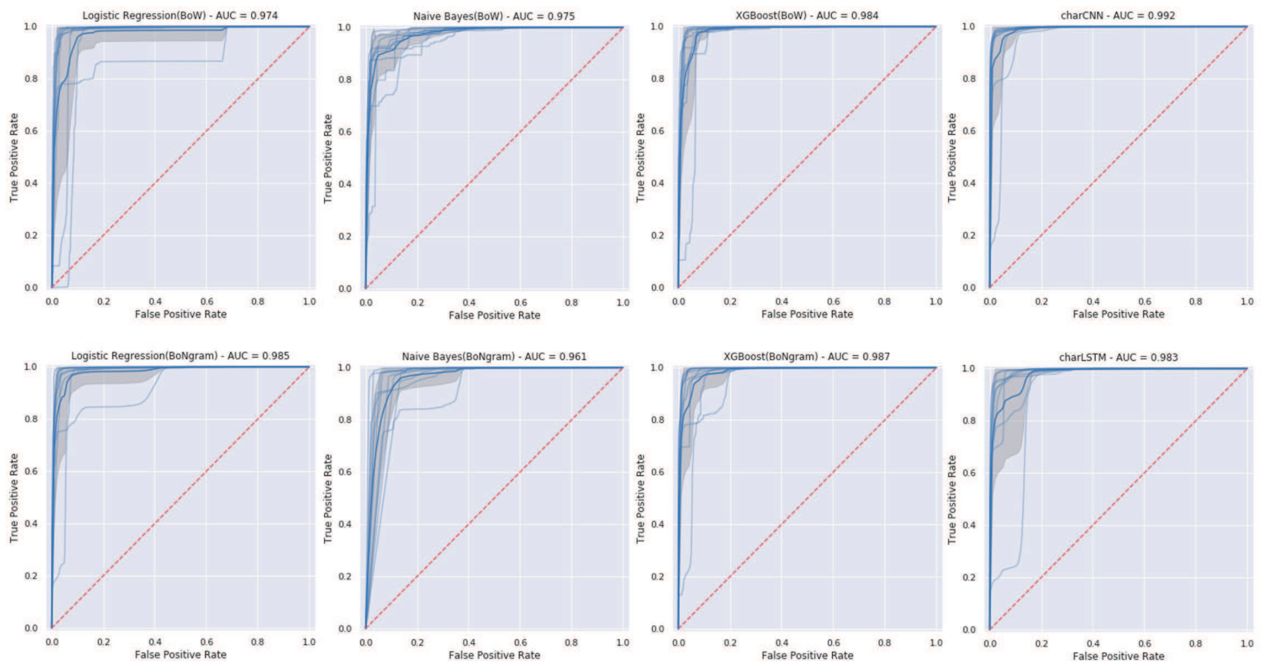


Fig. 5. ROC curves of all models trained in our experiments. Each subplot shows the roc curves of all models trained in the 10-fold cross validation procedure for a same machine learning technique.

We present two different flavors of the adversarial attack. In the first scenario, we consider an adversary that does not have any knowledge about the CSAM file detection mechanism, or access to its scoring results. The adversary makes random changes to characters in the file paths. The number of modifications allowed in the file path is the adversary's budget. The determination of an adversary budget is a challenging task since we want to enable the adversary to make the maximum amount of changes, without compromising the human comprehension of the meaning of the string of characters. Based on previous results, we believe that this amount lies between 10% and 15% of the length of the file path [12]. To stress-test our models, we also analyze the performance of our models under a 20% change.

In the targeted keywords scenario, we consider that the adversary has access to a vocabulary that is highly correlated with the CSAM samples in the training set. The adversary makes targeted character substitutions only on the terms that are good predictors for each class. The adversary's goal is to cause a decrease in the confidence output of the scoring algorithm and, consequently, result in a wrong label attribution to a specif file path. In this scenario, the adversary can modify up to four keywords, one character per keyword.

A. Adversarial Examples

Given a model $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$, which maps the input space $\mathcal{X}$ to the set of labels $\mathcal{Y}$, a valid adversarial example x_{adv} is generated by altering the original data example $x \in \mathcal{X}$ and should conform to the following requirements: $\mathcal{F}(x_{adv}) \neq \mathcal{F}(x)$ and $S(x_{adv,x}) \leq \epsilon$, where $S : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}^+$ is a similarity function and $\epsilon \in \mathcal{R}^+$ is a constant modeling the budget available to the adversary, the total number of modifications allowed.

B. Threat Model

In our threat model, the attacker is not aware of the model architecture, parameters and does not have access to the confidence scores output of the model [10]. In our first setting, the only knowledge the adversary has about the model is the input space $\mathcal{X}$ and the output space $\mathcal{Y}$.

In the targeted keywords setting, we assume that the adversary also has some knowledge of the training data. The training data originates from storage systems confiscated from perpetrators. In that case, we can assume other malicious entities have access to similar or identical data and understand the vocabulary of keywords used to identify the files. We describe details of the adversary's knowledge in section VI-D.

C. Adversarial Examples with Random Modifications

The adversarial examples are generated by randomly selecting a position in the file path string, and substituting the character in the selected position by a random alphanumeric character. This technique has been previously used to attack language models [3]. We evaluate our models for an adversarial budget of 10%, 15%, and 20% of file path length. For example, an adversary with a budget of 10% would modify 3 characters the file path *data/10_fold_data_split/dataset*, which is 31 characters long. A randomly modified file path, with ϵ=10%, is *data/10_f9ld_dkta_split/datoset*.

D. Adversarial Examples with Random Modifications in Targeted Keywords

In our experiments, we assume an adversary has access to the training data set, and uses the training data to identify character sequences that are highly correlated to CSAM file paths.

In this attack, we generate a list of tokens that are highly correlated with CSAM file paths, and we assume that the adversary has access to this list. We create the list of highly correlated tokens using Odds ratio, a widely used technique in information retrieval, and used for feature selection and interpretation of text classification models [18].

To generate this list, the first step of the attack is to identify which tokens are more likely to appear in CSAM file paths. We extract all tokens from the data set of file paths as described in section IV-A. For all keywords, we calculate the odds of the keyword being part of a CSAM file path and the odds of the keyword being part of a non-CSAM file path. The Odds ratio of a keyword k is defined as OR_k, and is computed as:

$$OR_k = \frac{\text{odds of k appear in CSAM file}}{\text{odds of k appear in non-CSAM}}$$

The list of CSA keywords, T_{csa}, comprises all keywords with Odds ratio greater than two. We make this list available to the adversary. We rank the keywords by Odds ratio and make the ranking available to the adversarial as well.

To create an adversarial example from an existing file path f, the adversary generates $K_{csa} = T_{csa} \bigcap T_f$, where T_f is the set of tokens of f. The file path modification occurs as follows: The adversary selects the word $k \in K_{csa}$ with the highest ranking. The adversary randomly modifies one character in k by randomly selecting a position in the keyword string, and substituting the character in the selected position by a random alphanumeric character. We denote by $\hat{k}$ the keyword with one randomly modified character. The adversary replaces k with $\hat{k}$ in the file path f. The adversary repeats this procedure for the next highest ranked keyword in the set K_{csa}. The number of allowed keyword replacements is determined by the budget ϵ.

E. Evaluation under adversarial examples

We evaluate the impact of adversarial modifications in test samples in the model's performance. We are especially interested in understanding which machine learning techniques are more robust when the data is adversarially modified at test time. All attacks target only CSAM file paths, and therefore we only evaluate the variation in recall rates. Additionally, we analyze the mean deviation in confidence scores for all models.

a) Random Modifications: Under this scenario, an adversary randomly modifies a percentage of the file path by randomly selecting characters and replacing them with random characters. A reasonable adversary budget in this scenario is between 10 % and 15%. Previous works have also considered this percentage range for perturbing text strings [12]. Considering that most file paths have a length between 40 and 200 characters, this results in changing between 6 and 30 characters in each file path. To stress-test our models, we also analyze the performance of our models under a 20% change.

Table III shows details on the confidence score variations for different adversarial budgets.

Figure 6 demonstrates the variation in recall rates as the percentage of random flipped characters increases. For an adversarial budget of 15%, we observe a decrease in recall rates of 0.02 for bag-of-words and naive Bayes and 0.07 in the CNN model. Interestingly, bag-of-ngrams naive Bayes suffers an increase in recall rates after flipping a percentage of the characters in the file path. This phenomenon results from the fact that modifications in the file paths can also increase the model's confidence score output. The boosted decision trees models undergo the most significant decrease in recall rates. A possible model overfitting can explain this to this specific data set distribution. In the case of 20% of flipped characters, the deep neural networks models' recall rates decrease ≈ 0.1, and bag-of-words naive Bayes decreases ≈ 0.05, and bag-of-

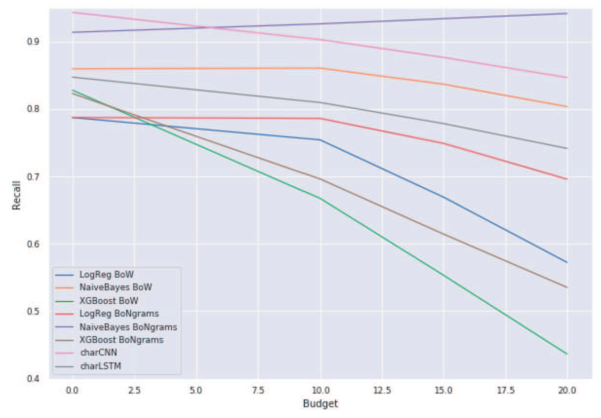


Fig. 6. Recall decrease as adversary budget increases in black box setting for different models.

TABLE III
MODEL ROBUSTNESS TO ADVERSARIAL RANDOM MODIFICATIONS WITH DIFFERENT LEVELS OF ADVERSARIAL BUDGET: 10%, 15% AND 20%

Features	**Algorithm**	**Budget: 10%**		**Budget: 15%**		**Budget: 20%**	
		Mean	σ	Mean	σ	Mean	σ
Bag of Words	Logistic Regression	0.05	0.19	0.09	0.24	0.13	0.28
Bag of Words	Naive Bayes	≈0	0.25	0.03	0.29	0.07	0.32
Bag of Words	Boosted Trees	0.16	0.35	0.26	0.40	0.35	0.40
Bag of N-grams	Logistic Regression	0.05	0.11	0.08	0.13	0.12	0.15
Bag of N-grams	Naive Bayes	-0.01	0.18	-0.01	0.2	-0.02	0.21
Bag of N-grams	Boosted Trees	0.12	0.30	0.19	0.35	0.26	0.38
Character quantization	CNN	0.03	0.13	0.05	0.18	0.08	0.22
Character quantization	LSTM	0.05	0.22	0.09	0.27	0.13	0.31

TABLE IV
MODEL ROBUSTNESS TO ADVERSARIAL TARGETED KEYWORDS MODIFICATIONS WITH DIFFERENT LEVELS OF ADVERSARIAL BUDGET: 1 ,2,3 AND 4 WORDS IN THE FILE PATH

Features	**Algorithm**	**Budget: 1**		**Budget: 2**		**Budget: 3**		**Budget: 4**	
		Mean	σ e	Mean	σ	Mean	σ	Mean	σ
Bag of Words	Logistic Regression	0.01	0.03	0.02	0.05	0.03	0.05	0.03	0.06
Bag of Words	Naive Bayes	≈0	0.05	≈0	0.06	≈0	0.07	≈0	0.07
Bag of Words	Boosted Trees	0.01	0.06	0.02	0.07	0.02	0.08	0.03	0.09
Bag of N-grams	Logistic Regression	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.02
Bag of N-grams	Naive Bayes	≈0	0.05	≈0	0.06	≈0	0.06	≈0	0.07
Bag of N-grams	Boosted Trees	≈0	0.05	0.01	0.06	0.01	0.07	0.01	0.07
Character quantization	CNN	≈0	0.02	≈0	0.03	≈0	0.03	≈0	0.04
Character quantization	LSTM	≈0	0.04	≈0	0.05	≈0	0.05	≈0	0.06

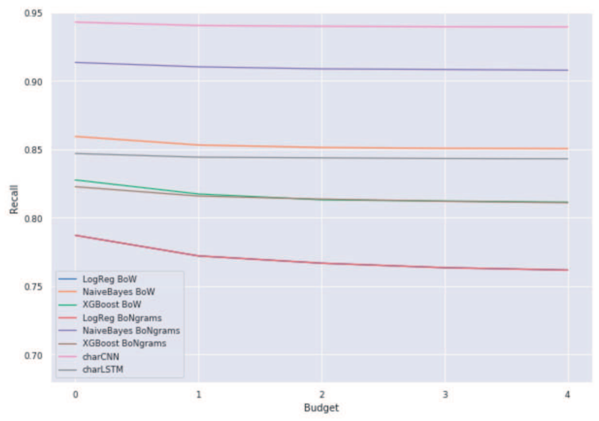


Fig. 7. Recall decrease as adversary budget increases in white box setting for different models.

ngrams naive Bayes, once again presents an increase in its recall rate.

b) Targeted Keywords: Having access to a list of highly correlated keywords with CSAM file paths, T_{csa}, will permit the adversary to make targeted changes in the CSAM file paths. This experiment allows the adversary to change one character per keywords, up to 4 keywords per file path. The recall variation as a function of the adversarial budget is illustrated in figure 7.

As indicated in figure 7, the most significant drop in accuracy happens for budget = 1. This is justified by the fact that the adversary also has access to the Odds ratio for each keyword. For budget = 1, the adversary will modify the keyword with the largest Odds ratio; for budget = 2, the adversary will modify the keywords with the two largest Odds ratio, and so on.

Overall, the targeted changes in the file paths result in small changes in the recall rates. Logistic regression and boosted decision trees have more considerable recall variations than naive Bayes and deep neural networks models.

Confidence score variation details are described in table IV. It is easy to observe that the mean change in confidence score in less than 0.1 for all models and all budgets. However, we see examples in our experiments where a single change resulted in a drastic drop in the confidence score.

We conclude that, without access to the model output, it is hard to craft an adversarial example close to the original sample, even when an adversary has access to a list of keywords highly correlated with the positive class.

VII. CONCLUSION

In this paper, we present a system for CSAM identification based solely on file paths. CSAM identification systems based on file paths have the advantage of not working directly with CSA photos or videos, thus providing a a classifier that is medium agnostic, of easy maintenance and reduced legal restrictions for acquiring and using a data set. To the best of our knowledge, our classifier is the first of the kind to achieve precision and recall rates over 90%, making an ideal candidate for deployment in investigations and detection of CSAM content in file storage systems. Our results have been evaluated with a data set consisting of over one million entries obtained from real investigations all over the world. Our experiments show that our proposed models generalize well to identifying CSAM content in file storage systems and are robust to some adversarial attacks introduced at test time.

We believe that this method, along with PhotoDNA hash, computer vision tools, and other forensics tools, should form part of a global toolset that enables organizations to fight the distribution of CSAM. Its purpose is to be a fast identifier that allows investigation agencies and technology companies to find images and videos in file storage systems quickly. This speed in detection can potentially reduce repeated victimization of abused children.

Online child sexual abuse imagery falls into a category of images that should not be distributed or be present in file storage systems. Although this is a complex problem to solve given the distributed nature of the internet, automated tools and machine learning-based systems can help technology companies and investigation agencies rapidly identify and take the appropriate actions.

REFERENCES

[1] Sumeet Agarwal, Shantanu Godbole, Diwakar Punjani, and Shourya Roy. How much noise is too much: A study in automatic text classification. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 3–12. IEEE, 2007.

[2] Mhd Wesam Al Nabki, Eduardo Fidalgo, Enrique Alegre, and Rocío Alaíz-Rodríguez. File name classification approach to identify child sexual abuse. In *ICPRAM*, pages 228–234, 2020.

[3] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

[4] Dasha Bogdanova, Paolo Rosso, and Thamar Solorio. Exploring high-level features for detecting cyberpedophilia. *Computer speech & language*, 28(1):108–120, 2014.

[5] R. Brown, E. Oldenburg, and J. Cole. Project vic: Helping to identify and rescue children from sexual exploitation. *Police Chief Online*, 2018.

[6] Elie Bursztein, Einat Clarke, Michelle DeLaune, David M. Elifff, Nick Hsu, Lindsey Olson, John Shehan, Madhukar Thakur, Kurt Thomas, and Travis Bright. Rethinking the detection of child sexual abuse imagery on the internet. In *The World Wide Web Conference*, WWW '19, page 2601–2607, New York, NY, USA, 2019. Association for Computing Machinery.

[7] J Davidson and P. Gottschalk. *Internet Child Abuse: Current Research and Policy*. Routledge-Cavendish, 2010.

[8] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

[9] Abhishek Gangwar, E Fidalgo, E Alegre, and V González-Castro. Pornography and child sexual abuse detection in image and video: A comparative evaluation. In *8th International Conference on Imaging for Crime Detection and Prevention (ICDP 2017)*, pages 37–42. IET, 2017.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Georg Heigold, Günter Neumann, and Josef van Genabith. How robust are character-based word embeddings in tagging and mt against wrod scramlbing or randdm nouse? *arXiv preprint arXiv:1704.04441*, 2017.

[12] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Textfool: Fool your model with natural adversarial text. http://groups.csail.mit.edu/medg/ftp/psz-papers/2019%20Di%20Jin.pdf, 2019.

[13] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[14] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[15] Michael H Keller and Gabriel J X Dance. The internet is overrun with images of child sexual abuse. what went wrong? *New York Times*, Sep 2019.

[16] Joao Macedo, Filipe Costa, and Jefersson A dos Santos. A benchmark methodology for child pornography detection. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 455–462. IEEE, 2018.

[17] Microsoft. Photodna cloud service, 2016. https://www.microsoft.com/en-us/PhotoDNA, Last accessed on 2020-05-08.

[18] Dunja Mladenić. Feature subset selection in text-learning. In *European conference on machine learning*, pages 95–100. Springer, 1998.

[19] Fudong Nian, Teng Li, Yan Wang, Mingliang Xu, and Jun Wu. Pornographic image detection utilizing deep convolutional neural networks. *Neurocomputing*, 210:283–293, 2016.

[20] Department of Justice. S.1738 - protect our children act of 2008. https://www.congress.gov/bill/110th-congress/senate-bill/1738, 2008.

[21] Claudia Peersman. *Detecting deceptive behaviour in the wild: text mining for online child protection in the presence of noisy and adversarial social media communications*. PhD thesis, Lancaster University, 2018.

[22] Claudia Peersman, Christian Schulze, Awais Rashid, Margaret Brennan, and Carl Fischer. icop: Automatically identifying new child abuse media in p2p networks. In *2014 IEEE Security and Privacy Workshops*, pages 124–131. IEEE, 2014.

[23] Claudia Peersman, Christian Schulze, Awais Rashid, Margaret Brennan, and Carl Fischer. icop: Live forensics to reveal previously unknown criminal media on p2p networks. *Digital Investigation*, 18:50–64, 2016.

[24] Ethel Quayle and Nikolaos Koukopoulos. Deterrence of Online Child Sexual Abuse and Exploitation. *Policing: A Journal of Policy and Practice*, 13(3):345–362, 04 2018.

[25] Olivia Solon. Child sexual abuse images and online exploitation surge during pandemic. *NBC News*, 2020.

[26] Thorn. Meet the new anti-grooming tool from microsoft, thorn, and our partners. https://www.thorn.org/blog/what-is-project-artemis-thorn-microsoft-grooming, Last accessed on 2020-05-08.

[27] Unicef. *Child safety online: Global challenges and strategies*. UNICEF Innocenti Research Centre, 2011.

[28] Paulo Vitorino, Sandra Avila, and Anderson Rocha. A two-tier image representation approach to detecting child pornography. In *XII Workshop de Visão Computational*, pages 129–134, 2016.

[29] B. Westlake, M. Bouchard, and R. Frank. Comparing methods for detecting child exploitation content online. In *2012 European Intelligence and Security Informatics Conference*, pages 156–163, 2012.

[30] Janis Wolak, Marc Liberatore, and Brian Neil Levine. Measuring a year of child pornography trafficking by us computers on a peer-to-peer network. *Child Abuse & Neglect*, 38(2):347–356, 2014.

[31] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Predicting domain generation algorithms with long short-term memory networks. *arXiv preprint arXiv:1611.00791*, 2016.

[32] E. Yiallourou, R. Demetriou, and A. Lanitis. On the detection of images containing child-pornographic material. In *2017 24th International Conference on Telecommunications (ICT)*, pages 1–5, 2017.

[33] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.