This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3324275

Metadata-Based Detection of Child Sexual Abuse Material

Mayana Pereira, Rahul Dodhia, Hyrum Anderson, and Richard Brown

Abstract—Child Sexual Abuse Media (CSAM) is any visual record of a sexually explicit activity involving minors. Machine learning-based solutions can help law enforcement identify CSAM and block distribution. Yet, collecting CSAM imagery to train machine learning models has ethical and legal constraints. CSAM detection systems based on file metadata offer several opportunities. Metadata is not a record of a crime and, therefore, clear of legal restrictions. This paper proposes a CSAM detection framework consisting of machine learning models trained on file paths extracted from a real-world data set of over 1 million file paths obtained in criminal investigations. Our framework includes guidelines for model evaluation that account for data changes caused by adversarial data modification and variations in data distribution caused by limited access to training data, as well as an assessment of false positive rates against file paths from common crawl data. We achieve accuracies as high as 0.97 while presenting stable behavior under adversarial attacks previously used in natural language tasks. When evaluating the model on publicly available file paths from common crawl data, we observed a false positive rate of 0.002, showing that the model operating in distinct data distributions maintains low false positive rates.

Index Terms—CSAM, Metadata, File Paths, Digital Crimes, Deep Learning, Machine Learning, Adversarial Examples

1 Introduction

International law enforcement handles millions of child sexual abuse cases annually. In 2022, the National Center for Missing and Exploited Children (NCMEC) tip line [12] received and reviewed over 32 million child sexual abuse files, and over 99.5% of the reports received by the NCMEC's tip line regarded incidents of suspected CSAM. Despite the 2008 *Protect Our Children Act* [33], the quantity of CSAM in digital platforms has dramatically grown over the last decade [12]. Online sharing platforms and social media facilitated [38] the explosive growth of CSAM creation and distribution [5]. Every platform for content searching and sharing, including social material, likely has CSAM on it [24]. Young teens are predominantly the target of predators, where almost 7 in 10 instances of child sexual abuse involving 11-13 year olds [13]. Recent survey [11] reported that over 15% of U.S. children have been exposed to online child sexual abuse. These rates are even higher for high for girls (23%) and transgender or gender-fluid children (20%) [11].

As the scale of the problem grows, technology plays an essential role in CSAM identification. Companies that manage user-generated data, such as Pinterest, Meta, Microsoft, Apple, and Google, have made detecting and removing CSAM a top priority. Although several non-profit organizations such as Project VIC International[1], Thorn[2] and the Internet Watch Foundation[3] focus on building tools to combat CSAM proliferation, the creation and distribution of CSAM is still a growing problem.

The COVID-19 pandemic triggered a significant increase in the distribution of CSAM via social media and video conferencing apps [39], and identification of CSAM is a highly challenging problem. First, it can manifest in different types of material: images, videos, streaming, video conferences, and online gaming, among others. Undiscovered and unlabeled CSAM on the internet is estimated to be magnitudes greater than the currently identified CSAM. Second, discovering new material is still highly dependent on human discovery. Despite the significant progress in machine learning models for CSAM identification with modern deep-learning architectures [28], [42], [47], these models rely on the availability of labeled images, which can lead to technical limitations. As new material is created daily, we understand that utilizing complementary signals can advance the capability of digital platforms in detecting and removing illegal content. The use of metadata, such as file paths and file names, has been proposed by [37]. This is an effective approach since distributors use coded language to communicate and trade links of CSAM hosted in plain sight on content-sharing platforms, websites, newsgroups, bulletin boards, peer-to-peer networks, internet gaming sites, social networking sites, and anonymized networks [4]. In particular, peer-to-peer (p2p) file-sharing networks are an environment where CSAM is actively hosted and shared [14], [27], and searches in p2p networks usually work by matching search terms with file names and file paths.

The lack of frameworks for evaluating machine learning models for CSAM detection prevents a better understanding

- *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 and H. Anderson are with Microsoft Corporation, Redmond, WA, 98052.*
- *R. Brown is with Project VIC International.*

1. https://www.projectvic.org
2. https://www.thorn.org
3. https://www.iwf.org.uk
4. https://www.thorn.org/child-pornography-and-abuse-statistics/

of model performance under multiple scenarios that can happen during deployment. Before deployment, organizations should test the CSAM detection model under different conditions. An evaluation scenario needs a real-world data set with similar data distributions to what the model will be exposed to after deployment. A critical analysis is testing the model on completely benign out-of-sample data sets. The burden caused by a high false-positive rate can halt the deployment of such systems. Furthermore, it is crucial to understand how adversarially modified data impacts model performance.

1.1 Our Contributions

Building machine learning systems for detecting CSAM media is a complex task. Due to the associated legal constraints, systems that rely on metadata for detecting and blocking the distribution of CSAM can expedite the hard work of NGOs and content moderators.

This work proposes a framework for training and evaluating machine learning models for CSAM detection based solely on file metadata. Our framework provides guidelines for assessing CSAM detection models against adversarial attacks and models' ability to perform on different data distributions.

The proposed models compute the likelihood that a file depicts CSAM based on its file path. Our experiments show that the resulting model achieves the desired performance in challenging real-world deployment scenarios without relying on the availability of CSAM images and videos for training.

We list the contributions as follows:

- We propose a framework for evaluating machine learning models for CSAM identification to prepare for deployment. Our framework, illustrated in Figure 1, presents a testing pipeline that covers real-world circumstances - expected when deploying a machine learning model for CSAM detection: (i) test on CSAM and non-CSAM samples; (ii) test on adversarially modified CSAM samples to evade detection; (iii) test on benign samples from open data sources.
- We train and compare several machine-learning models that analyze file paths and file names from file storage systems and determine a probability that a given file has child sexual abuse content. Our experiments include traditional machine learning algorithms, deep neural network architectures, and Transformer-based models. We train our models on a real-world data set containing over one million file paths from apprehended hard drives during investigations. It is the most extensive file path data set composed solely of file paths from apprehended hard drives. Our best classifier achieves recall rates over 0.94 and accuracy over 0.97 on holdout sets; it maintains a high recall rate in adversarially modified inputs; when tested against benign samples from other data distributions, it achieves a false-positive rate of ≈ 0.01.

To our knowledge, our work is the first to propose a framework for evaluating CSAM detection systems that include adversarial examples in the evaluation stage. Our results show that machine learning based on file paths can detect CSAM in storage systems and achieve the desired performance in all the proposed evaluation scenarios.

Finally, we remark that our solution is part of the technology suite used by the non-profit Project VIC[5] and currently in use.

2 Related Work

Identification of CSAM via statistical algorithms is a reasonably recent approach. In the early 2000s, the US and the UK introduced laws targeting online exploitation of minors (COPA in the US, Crime and Disorder Act UK) [6]. However, only in 2008 was the first widely used CSAM identification technology released.

PhotoDNA Hash PhotoDNA Hash (PDNA) is a widely used technique for the automated identification of CSAM. The PDNA uses a fuzzy hash algorithm to convert a CSAM image to a long string of characters. The converted hashes are compared against others to find identical or similar images. PDNA technology enabled a faster discovery of CSAM while protecting the victim's identity. This system is still one of the most widely used methods for detecting CSAM images worldwide. Search engines, social networks, and image-sharing services utilize databases of hashed CSAM images to eradicate harmful content from their platforms. PDNA is a signature-based technology; it recalls only known CSAM. Therefore, identifying new CSAM in a PDNA-based system requires manual labeling.

Machine Learning for Image Identification Since PDNA's first development, computer vision models have undergone a revolution resulting in novel machine learning-based models for pornography and CSAM detection [28], [32], [36], [47]. The current approaches either combine a computer vision model to extract image descriptors [42], train computer vision models on pornography data [16], perform a combination of age estimation and pornography detection [28] or synthetic data [47]. However, due to legal restrictions in maintaining a database of CSAM images, all current works are based on either unrealistic images [47] or validated by authorities in small data sets [16], [28], [42] that hardly represent the actual data distribution in the internet [5].

Adversarially modified data samples Adversarial inputs are small perturbations intentionally crafted into data to evade detection by a model. For text applications, this can include injecting random noise that does not dramatically alter human understanding of the text. Substitutions such as replacing "before" with "b4", homoglyph substitutions, and other substitutions, such as using "Lo7ita" instead of "Lolita" [45]. The effects of adversarial modifications in text classification have been explored for different NLP techniques, including classification [1], machine translation [3], and word embeddings [18]. Depending on the information available to the adversary, it distorts portions of the text most likely to contain a signal necessary to the classification task.

5. "A non-profit whose technologies are used by thousands of law enforcement officers worldwide" - https://www.projectvic.org/vic-point

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3324275

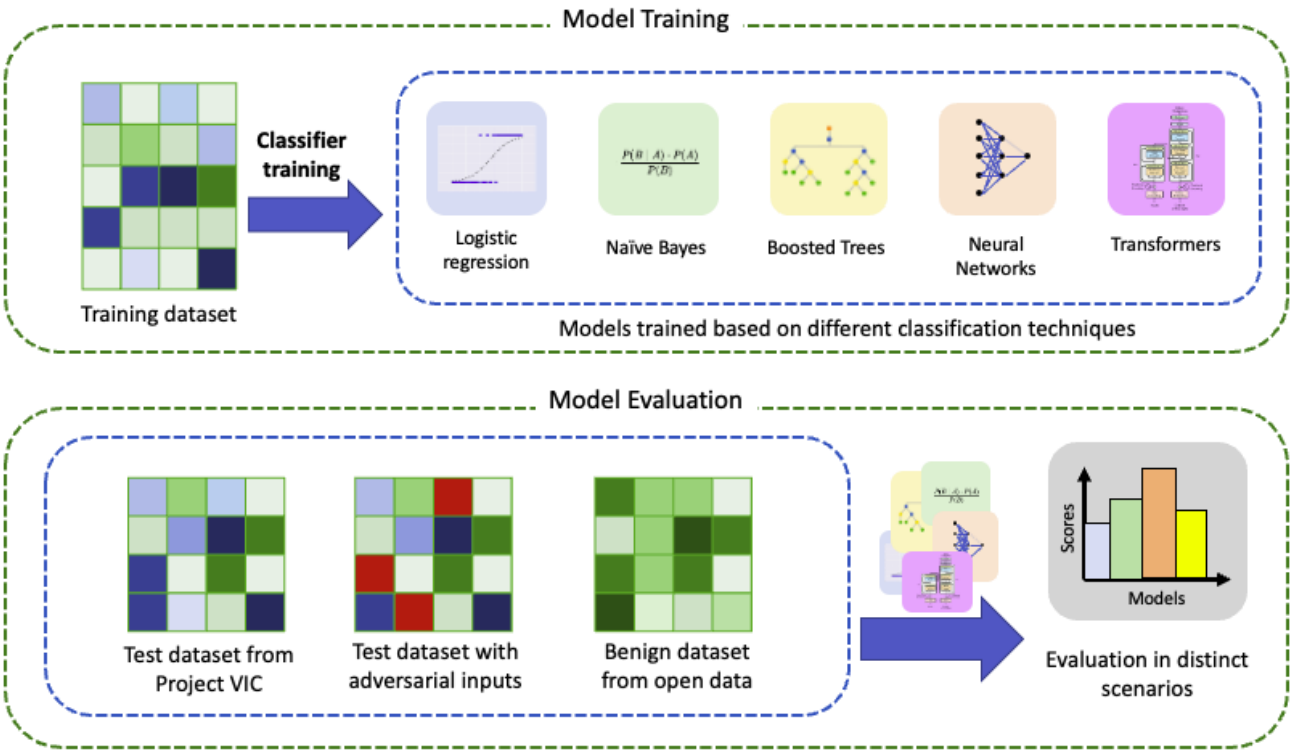

Fig. 1: Pipeline for model training and evaluation of machine learning models for CSAM detection. (i) During model training, we train models utilizing several machine learning techniques, such as logistic regression, Naive Bayes, boosted trees, deep neural networks, and Transformers. (ii) We construct different testing data sets to model performance in different circumstances of practical relevance during the model evaluation. We propose a testing framework that tests the model under three scenarios: File paths from out-of-sample hard drives, file paths intentionally modified by an adversary to evade detection, and file paths from benign sources (open data).

CSAM File Metadata Classification While significant efforts have focused on the images, some researchers have looked for complementary signals to help identify CSAM. Such measures include queries that return CSAM in search engines, file metadata, and conversations that imply grooming or exchange of CSAM [40]. Other efforts have used textual signals to identify where CSAM might be located, such as keywords related to website content [43], using NLP analysis [2], [10], [34], [35], [37], conversations [4]. Our work falls into this category. Previous works have found that perpetrators use a specific CSAM vocabulary to name files [37]. For this reason, using file paths, which is the combination of the file location and file name, is a promising approach for CSAM identification. Other related works aim to identify CSAM based solely on file path [2], [2]. However, these works do not address important questions such as classifiers' robustness against adversarial examples and performance in out-of-sample benign data sets.

Recent Works The recent publication of a survey on detecting and preventing online child sexual abuse material [31] compares over 35 studies on the topic. The findings of this survey highlight the problem's complexity and the need to combine computer vision and natural language processing techniques to combat this heinous crime. In [26], the authors propose a pipeline for extracting signals from data, which they claim is highly valuable in highlighting essential aspects of the overall distribution of data. This pipeline can provide valuable insights into databases that cannot be disclosed. Moreover, in [46], the authors used real-world data from the United Kingdom to study the behavior and preferences of 53 anonymous CSAM suspects who were active on the dark web and noticed by the police. This research provides a unique perspective into the minds of these dangerous individuals and can be used to develop more effective strategies to prevent them from exploiting innocent children online.

3 Methods

This section describes our experiments' data set, methods, and algorithms.

3.1 Training data set

Our supervised learning approach to identify CSAM file paths utilizes a binary labeled data set. To separate the data set into independent training and test sets, we split the data by storage system information (e.g., driver designations) to not leak information from the training to test set, which is also known as model leakage [23]. Our data set consists of file paths collected by Project VIC International[6]. The data consists of 1,010,000 file paths from 55,312 distinct storage systems. File paths are strings that contain the location

6. https://www.projectvic.org

information of a file (folders) in a storage system and the file name. In Table 1, we present details on the different types of content that constitute the data set and the number of samples for each type.

TABLE 1: Project VIC data set description. Data set used for model training and model testing. It contains non-pertinent (negative class) file paths and different types of file paths of child exploitative and child sexual abuse material (positive class).

Content Type	Samples	Label
Non-pertinent	717,448	0
Child exploitative and child sexual abuse	292,552	1

The Project VIC data set used in our experiments contains 717,448 non-pertinent file paths and 292,552 file paths containing child exploitative and sexual abuse material. We note that all the provenance of 1,010,000 file paths in our data set is from hardware apprehended for investigations. The training and testing data sets accurately represent the data in a deployment scenario.

Due to ethical implications, the authors cannot share the Project VIC file path data set or make it publicly available. The data set contains real file paths and file names, which, in the wrong hands, can be used to search for CSAM files in search engines, discussion forums, and peer-to-peer networks.

3.1.1 File Path Characteristics

The distribution of file path length helps us define the size of the character embedding vectors in our deep neural networks models and the size of the word vectors used as input to the transformers-based model we fine-tune to the task of CSAM file path identification.

When analyzing the distribution of file path lengths in the data set, we observe that 95% of file paths have 300 characters or less. Limiting the size of the character embedding layer helps increase the time and memory efficiency of the model. We set our character embedding layer size to 300 characters. We truncate file paths with more than 300 characters by discarding the initial characters and keeping only the last 300 characters. We pad with zeros on the left all file paths with less than 300 characters.

The transformers-based model also takes a fixed-sized vector, in this case, a vector of words, as input. We consider a word to be a sequence of alphanumeric characters separated by a dash, slash, colon, underscore, or period. By counting the number of words in each file path, we see that over 99% of file paths have at most 64 words. More precisely, in our data set, there is only one file path with more than 60 words. For this reason, we set the input vector size for the transformers-based model to 64 words.

In addition to the file path size, the size of the set of possible characters is a data attribute that influences model architecture. Although over 96% of the file paths are represented by alphanumerical characters and words in English, the remaining 4% of the data contains multiple languages and characters from Chinese, Korean, and Japanese alphabets.

3.1.2 Cross Validation Data Split

We use a K-Fold Cross Validation methodology in our experiments with K=10. The file path contains information about the storage system in our data set. If a storage system contains a high volume of CSAM files, the model could learn that files from specific storage systems are highly likely to be CSAM. This is known as model leakage [23]. Leakage in machine learning modeling consists of introducing information about the target of a machine learning problem at training time. To avoid model leakage, we split the data by storage system information. We divide each cross-validation fold into 80/10/10 for training, validation, and testing.

The information before the first backlash of a file path specifies the external storage system or a laptop/desktop. We use this information to partition the data set for cross-validation.

3.2 Text Vectorization

We present the concepts utilized for text vectorization: term-frequency inverse-document-frequency (TF-IDF), character-based quantization, and word vectors that will serve as input to the transformers-based model.

3.2.1 TF-IDF

This technique attributes weights to words (or sequences of characters) in a text [22]. First, it computes the term frequency (TF), which is the number of times a term occurs in a given document. We calculate the inverse-document frequency component (IDF) as follows:

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

n is the total number of documents in the document set, and $df(t)$ is the number of documents in the document set containing the term. For each term, we compute the product of the TF and IDF components. The Euclidean norm then normalizes the resulting TF-IDF vectors.

When vectorizing a text with TF-IDF, the terms in a text can be words or sequences of characters. We investigate both approaches in our work. When using *words* as terms in TF-IDF, we refer to the text vectorization as bag-of-words (BoW). When vectorizing the file paths as BoW, we consider a *word* to be a sequence of alphanumeric characters separated by a dash, slash, colon, underscore, or period. We construct the bag-of-words model by selecting the 5,000 most frequent *words* from the training subset. We utilize this text representation in combination with TF-IDF. The data set of vectorized file paths is used as input to three different learning algorithms: logistic regression, naive Bayes, and boosted decision trees.

When using the sequence of characters as terms in TF-IDF, we refer to the text vectorization as bag-of-n-grams or n-grams. When vectorizing the text as n-grams, 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 in s of length n. We construct the bag-of-n-grams models by selecting the 50,000 most frequent n-grams (up to 3-grams)

from the training data set. We utilize this text representation in combination with TF-IDF. The data set of vectorized file paths is used as input to three different learning algorithms: logistic regression, naive Bayes, and boosted decision trees.

3.2.2 Character-based quantization

This type of text representation defines an alphabet of size m as the input language, quantized using 1-of-m encoding. We transform each textual input of length l into a sequence of such m sized vectors with fixed length l. We truncate texts with more than l characters, discarding the exceeding initial characters. If the text is shorter than l, we pad the string with zeroes on the left. We represent characters that are not in the alphabet as all-zero vectors. The alphabet used in our models consists of m = 802 characters, including 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.

3.2.3 Word vectors for pre-trained models

We use transformer-based models in our experiments. We utilize bidirectional encoder representation from transformers, BERT [9] pre-trained model. To prepare the text to serve as an input to the pre-trained BERT model, we represent the file path as a sequence of words by removing dashes, slashes, colons, underscore, and periods. We limit the sequence of words to 64 as indicated in section 3.1.1.

3.3 Learning Algorithms

We use several learning algorithms that have been successful at short-text classification. We consider two broad approaches: i) Traditional machine learning models ii) and Neural network models.

3.3.1 Traditional ML on extracted features

We chose a set of standard ML algorithms with distinct functional forms: logistic regression, naïve Bayes, and boosted decision trees. We chose these machine learning models because of their interpretability and simplicity.

Logistic Regression This discriminative classification algorithm models the posterior probability $P(Y|X)$ of the class Y given the input features X by fitting a logistic curve to the relationship between X and Y. Model outputs are interpreted as probabilities of the occurrence of a class [30].

Naive Bayes Conditional probability model that assumes independence of features: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)$ representing some n features, it assigns to this instance probabilities $P(C_k \mid x_1, \ldots, x_n)$ for each of K possible outcomes or classes C_k. The problem with the above formulation is that if the number of features n is large or a feature can take on too many values, basing such a model on probability tables is infeasible. A reformulation of the model makes the problem tractable. Using Bayes' theorem and assuming independence of the feature variables, the conditional probability is decomposed as:

$$P(C_k \mid \mathbf{x}) = \frac{P(C_k)\, P(\mathbf{x} \mid C_k)}{P(\mathbf{x})}$$

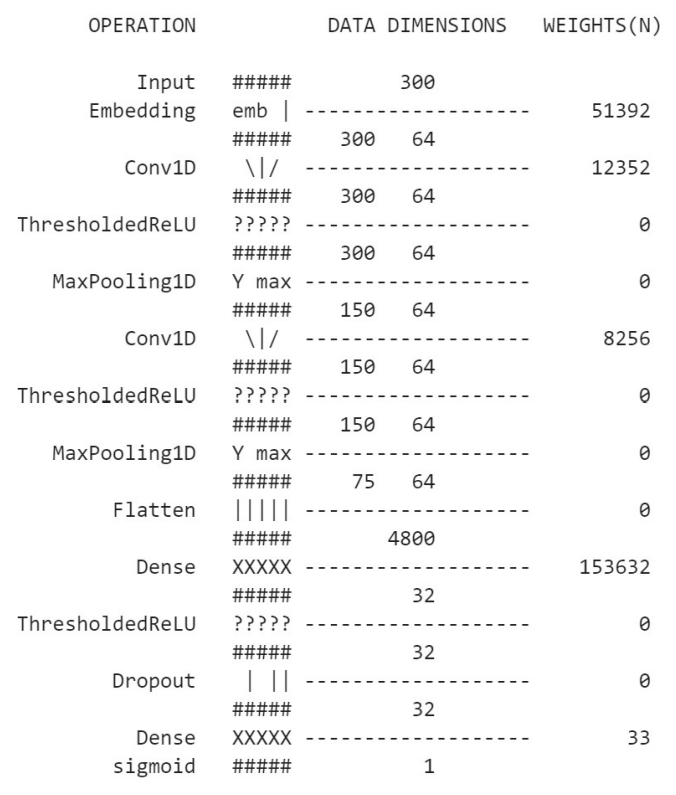

Fig. 2: Diagram of the deep neural network architecture with CNN layers used to train one of our CNN-based models. The above diagram indicates all data dimensions and the number of weights in each layer of our CNN model.

Boosted Decision Trees The base of the technique is tree ensembles. The algorithm trains each tree using a boosting process in which each subsequent tree is built with weighted instances misclassified by the previous tree [15]. A simple majority vote of the individual trees defines the classification of a new instance with a trained ensemble of trees.

3.3.2 Deep Neural Networks on Learned Embeddings

All neural network architectures start with an embedding layer representing each character by a 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. The variant of LSTM architecture used in our work is the standard "vanilla" architecture as used in [44].

Convolutional Neural Networks One-dimensional Convolutional Neural Networks (CNNs) are a good fit when the input is text, treated as a raw signal at the character level [48]. The CNN automatically learns 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) to make predictions. The algorithm learns the filters (also called kernels) during backpropagation. Figure 2 shows detailed information on the CNN-based architecture used in our experiments, including data dimensions and weights in each layer.

Long Short-Term Memory network This flexible architecture generalizes manual feature extraction via n-grams, for example, but instead learns dependencies of one or

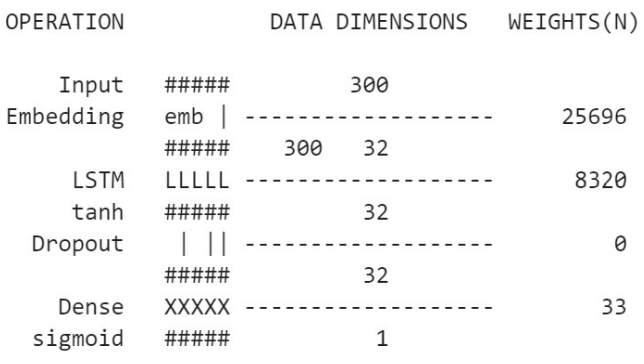

Fig. 3: Diagram of the deep neural network architecture with LSTM layer used to train one of our LSTM-based models. The above diagram indicates all data dimensions and the number of weights in each layer of our LSTM model .

multiple characters, whether in succession or with arbitrary separation. The long short-term memory network (LSTM) layer is an implicit feature extraction instead of explicit feature extraction (e.g., n-grams) used in other approaches. Rather than represent file paths explicitly as a bag of n-grams, for example, the LSTM learns patterns of tokens that maximize the performance of the second classification layer. Figure 3 shows detailed information on the LSTM-based architecture used in our experiments, including data dimensions and weights in each layer.

Transformer-based model Transformer [41] is a model architecture that dismisses recurrence and relies solely on an attention mechanism to derive global dependencies between input and output. BERT's model architecture is a multilayer bidirectional Transformer [9]. BERT's framework comprises two steps: pre-training and fine-tuning. Pre-trained BERT models are trained on unlabeled data over different pre-training tasks and can be easily fine-tuned to several downstream tasks. We utilize pre-trained `bert-base-uncased` from the Hugging Face Transformer library [7]. We use `BertForSequenceClassification` class from the same library for fine-tuning, which is the downstream task suitable for our classification problem. We fine-tune the BERT model using the hyperparameters suggested by [9].

3.4 File Path-Based CSAM Classifiers

Our work investigates four approaches for CSAM file path classification.

1 **Bag-of-words** This approach encodes the file path string into a vector of words. The weights of the words are attributed using TF-IDF. We utilize the resulting vectors as input to traditional machine learning classifiers (logistic regression, boosted decision trees, and Naive Bayes).

2 **Character n-grams** A list of character sequences on size N encodes the file path. The weights of the sequences are attributed using TF-IDF. We use the resulting vectors of the character sequences with traditional machine learning classifiers (logistic regression, boosted decision trees, and Naive Bayes);

3 **Character-based neural networks** We use sequences of encoded characters with a convolutional neural network (CNN) and a long short-term memory network (LSTM).

4 **Pre-trained BERT model** The `bert-base-uncased` pre-trained model is fine-tuned for downstream sequence classification task .

7. https://huggingface.co/bert-base-uncased

4 Model Evaluation

We present our results for all our classifiers in Table 2. All performance metrics were measured using a 10-fold cross-validation on the Project VIC data set.

For each of our classifiers, we report the mean and the standard deviation over the folds for 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' generalizations by looking into the standard deviations over the cross-validation folds.

4.1 Traditional Machine Learning Models

There are significant advantages of traditional machine learning models compared to deep neural networks. Understanding how well these models perform can help scientists and investigators leverage such models' most remarkable characteristic: feature interpretability. The most relevant predictive tokens, or n-grams, can give clues about vocabulary words in the data set and utilize them in other CSAM detection systems. Table 2 shows that the model trained with bag-of-words and bag-of-n-grams operates in similar AUC and accuracy ranges. When analyzing recall rates of traditional models, we note that both naive Bayes models have the highest average rates and lowest standard deviations. The naive Bayes with bag-of-n-grams features presents the best recall of all traditional models, of about 0.91. Among the other models trained using bag-of-n-grams, naive Bayes shows a much smaller recall standard deviation (σ = 0.085) when compared to logistic regression (σ = 0.20) and boosted decision trees(σ = 0.20).

Although evaluating CSAM classification models relies on recall rates, precision can become the most significant metric when deploying a model in an environment that potentially analyzes hundreds or thousands of file systems and, consequently, millions of file paths. The burden of having several thousands of false positives can result in an inefficient process and potentially delay investigations and the discovery of true positives. The AUC metric captures the ability of a classifier to operate with high recall when low false positive rates are necessary. By analyzing the traditional models' AUC, we observe that boosted decision trees perform better than the two other techniques.

4.2 Deep Neural Networks and Transformers-based Models

We achieved the best performance across all categories with deep neural network architecture. We trained three architectures: a layered CNN, an LSTM-based model, and BERT. The LSTM model achieves results very similar to the fine-tuned BERT model. Both models present accuracy above

TABLE 2: Model evaluation. Experiments with traditional machine learning and neural networks using Project VIC's data set. We evaluate the AUC-ROC, accuracy, precision, and recall. We measured results across 10-fold in a cross-validation setting. We report the mean (μ) and the standard deviation (σ) for each metric.

Model	AUC		Accuracy		Precision		Recall	
	μ	σ	μ	σ	μ	σ	μ	σ
BoW Logistic Regression	0.967	0.035	0.922	0.062	0.904	0.090	0.787	0.202
BoW Naive Bayes	0.972	0.011	0.927	0.032	0.875	0.070	0.859	0.114
BoW Boosted Trees	0.982	0.013	0.934	0.062	0.903	0.096	0.827	0.203
N-grams Logistic Regression	0.980	0.021	0.931	0.060	0.919	0.088	0.793	0.202
N-grams Naive Bayes	0.958	0.023	0.929	0.032	0.839	0.083	0.913	0.085
N-grams Boosted Trees	0.983	0.015	0.931	0.060	0.906	0.094	0.822	0.203
Character-based CNN	**0.990**	0.011	**0.968**	0.019	**0.938**	0.034	**0.943**	0.060
Character-based LSTM	0.982	0.017	0.953	0.044	0.937	0.064	0.903	0.129
BERT	0.987	0.013	0.955	0.035	0.934	0.048	0.896	0.100

0.95, precision over 0.93, and recall $\approx$ 0.9. However, our CNN model consistently outperforms all the other models in mean performance metrics across all folds and the lowest standard deviation.

4.3 Comparison with previous works

Although several previous works have proposed using file paths for CSAM detection, they lack rigorous methodology in which test data correctly emulates the data during deployment. It is hard to evaluate the actual model performance without a test data set with a similar distribution to the data during deployment. In [2], although authors achieved a recall rate of 0.98, the training and testing data utilized in the experiments do not accurately represent data in a deployment scenario. CSAM and non-CSAM file paths come from entirely different data sources and do not accurately represent file paths' data distribution in real deployment scenarios. In [10], the authors propose a sound methodology for collecting CSAM and non-CSAM file paths from a pool of Windows disk images. However, accuracy, precision, and recall rates equal to 1.00 indicate model overfitting. A third work focuses on detecting CSAM [34] using file names. Once again, it does not provide a testing scenario that reasonably represents a deployment setting. The work explores only traditional machine learning techniques, achieving a maximum accuracy of 0.97.

5 Model Evaluation with Adversarial Examples

In classical machine learning applications, we assume the underlying data distribution is stationary at test time. However, a testing pipeline of models to detect illegal activities should anticipate an intelligent, adaptive adversary actively manipulating data. Perpetrators purposely add typos and modifications to file identifiers [37] to evade blocklists and machine learning-based detection mechanisms. As illustrated in Figure 4, we modify our test data set to simulate an adversary actively changing the file paths to elude the classifiers.

Our CSAM file path detector assumes that file paths contain information about file contents; therefore, we can detect CSAM files by only analyzing file paths. Perpetrators often share CSAM files, using file names to identify file contents, a practice common in peer-to-peer systems [14], [27]. In such a case, the adversary wants to make the maximum possible changes without compromising others' ability to search the file.

The proposed adversarial analysis has limitations, as criminals could adopt alternate coding methods or randomly name their CSAM files to evade detection. However, the alarming reality is that the online search for CSAM content has significantly increased in recent years, with CSAM content being openly advertised online [8]. Given the widespread nature of this issue, it is reasonable to assume that adversaries will opt for subtle modifications and obfuscation in the file paths rather than completely concealing them. Our experiments incorporate the most commonly observed adversarial techniques in evading text-based classification, as documented in previous works [7], [25]. It is important to acknowledge that this is not an exhaustive list of possible attacks but provides a solid foundation for understanding the threat landscape.

5.1 Threat Model

In our threat model, the attacker is unaware of the model architecture and parameters and cannot access the confidence scores predicted by the model. The attacker attempts to cause an integrity violation in the model by modifying the input under bounded perturbation size [17]. The only knowledge the adversary has about the model is the input space and the output space. Notably, the attacker cannot immediately observe the output for a given input, so it cannot directly optimize for a worst-case outcome. But,

8. https://www.thorn.org/child-sexual-exploitation-and-technology/

since the attacker generally understands that file names are being monitored, they use randomly applied heuristics that attempt to evade the model, as illustrated by the random character replacement attack in Figure 4.

We assume the adversary has access to a list of CSAM and non-CSAM trigger words, similar to the adversarial attacks proposed in [25] to evade spam email detection models. Using odds ratio, we create a list of trigger words, i.e., words highly correlated with CSAM and non-CSAM file paths. Odds ratio is a widely used technique in information retrieval for feature selection and interpretation of text classification models [29]. The CSAM word spacing attack in Figure 4 illustrates the type of attack.

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 for all keywords. The Odds ratio of a word w is computed as:

$$\text{Odds Ratio} = \frac{\text{odds of } w \text{ appear in CSAM file}}{\text{odds of } w \text{ appear in non-CSAM}}$$

The CSAM lexicon comprises all keywords with Odds Ratio greater than two. An analogous process identifies the non-CSAM trigger words. We assume the list of trigger words is available to the adversary.

5.2 Random Character Replacement

The adversarial examples are generated by randomly selecting a position in the file path string and substituting the character in the chosen position with a random alphanumeric character. This technique has been previously used to attack language models [3]. We evaluate our models for three character replacement rates: 10%, 15%, and 20% of file path length.

5.3 Homoglyph Replacement

The homoglyph replacement attack [20] obfuscates words by modifying words in a text while keeping them readable. This attack replaces characters of CSAM trigger words in file paths with homoglyphs, i.e., a character with identical or similar shapes. We utilize the homoglyph dictionary from [7] to make the character replacements. This attack is also known as leetspeak attack [19].

5.4 Synonym Replacement

In [25], authors presented this attack as an effective way to evade spam classification models. The synonym replacement attack finds trigger words in CSAM file paths and replaces them with synonyms. We use `nltk` module to find word synonyms. This attack intends to modify the file path without changing its meaning.

5.5 CSAM Word Spacing

In [25], authors propose the spacing attack. The spacing attack adds spaces between characters of trigger words. In our attack, since file paths comprise our data, we add an underscore "_" between every character of words from the CSAM trigger word list. Intuitively, the text parser and n-gram sequences of the trigger words would not recognize the new modified word, while a human could still read and recognize the keyword.

5.6 non-CSAM Word Injection

Manipulating the file paths by adding words more likely to appear in non-CSAM file paths was adapted from the *ham word injection attack* presented in [25]. This attack consists of selecting a word from the non-CSAM trigger word list and injecting this word into the file path. We evaluate our models when injecting one, two, and three non-CSAM words in the file paths.

5.7 Experimental Results

We evaluate the impact of adversarial modifications in test samples on the model's performance. We perform adversarial modifications in the test fold of the cross-validation on the Project VIC data set. Details on how we conducted cross-validation are in section 3.1. We are interested in i) understanding which machine learning techniques are more robust when an adversary modifies the data at test time ii) and how much the performance of the models changes.

When evaluating models under attack, the precision of the models will not change (There is no manipulation of non-CSAM files during the attack). The attack's objective is to reduce model recall or increase the incidence of false negatives; consequently, we only assess changes in recall rates.

Under the random character replacement attack, 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 same percentage range for perturbing text strings [21]. Since most file paths have a length between 40 and 200 characters, this changes 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 3 demonstrates the variation in recall rates for different categories of adversarial attacks. In the presence of file paths generated via the character replacement attack, both variations of logistic regression models will present a slight decay in recall rates when 10% of the file path is modified. We see the recall rates decrease as we increase the percentage of modified file path. Logistic regression combined with bad-of-words text representation achieves a decrease of almost 25 points percentage in the recall rate, compared to the original recall when 20% of the file path is modified. Boosted tree models also see a sharp decrease in recall rates as we increase the percentage of modified file path. Compared to the original recall rate of 0.83, A drop of 16 points percentage is observed in the lowest level of the attack, when 10% of the file path is modified. Recall rates get to 0.43 at the highest level of the attack when 20% of the file path is modified. At a recall rate of 0.43, more than half of the modified file paths evade the classifier. CNN, LSTM, and BERT models also present decreases in recall rates as we increase the number of modifications in the file paths. The BERT model shows the smallest overall decrease of the three models. The naive Bayes model presents surprising results. The recall rates did not decrease for any level of the attack. In addition, we notice a slight recall increase as the number of modifications in the file paths increased. The

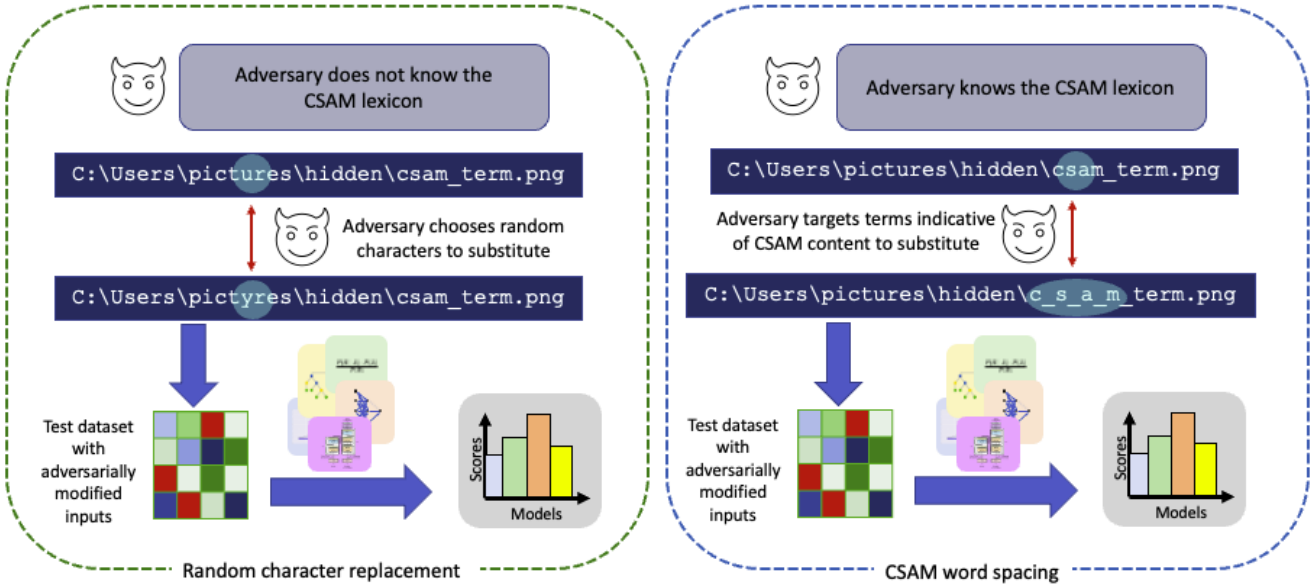

Fig. 4: Example of adversarial inputs generation. We generate adversarial inputs based on several different adversarial attacks. Here, we illustrate two attacks: (1) In the random character replacement attack, the adversary chooses random positions in the file path string and replaces the character with a randomly chosen character. (2) The CSAM word spacing attack allows the adversary access to a CSAM lexicon. The adversary adds spacing between characters in words in the CSAM lexicon.

randomness added to the CSAM file paths helped the naive Bayes classifier identify better which file paths were CSAM.

The homoglyph attack is typical in scenarios where an adversary wants to evade spam email classifiers [8]. We evaluate the performance of all models to understand which models are more resilient to this kind of attack. Logistic regression and boosted tree models all see a significant reduction in recall rates. We observe that n-grams boosted tress recall rate achieve 0.40 under the homoglyph attack, which leads us to conclude that this model is susceptible to this attack. CNN and LSTM models also present a decline in recall rate, going down to 0.71 and 0.74, respectively. Once again, the n-grams naive Bayes model performs best under attack, showing a recall rate of 0.88, followed by BERT, which presented a recall rate 0.84.

The synonym attack was the attack that overall impacted less the models' performance. Most models did not see a decrease, and several models experienced a small increase in recall rates. The spacing attack affected logistic regression and boosted trees, decreasing the recall rates of these models by more than 20 points. CNN, LSTM, and BERT suffered decreases in recall rates of approximately 10 points percentage. N-grams naive Bayes presented, once again, the smallest decrease in recall rates, with a reduction of 3 points percentage.

Non-CSAM word injection was the attack that most impacted the n-gram naive Bayes model. By injecting only one word highly correlated with non-CSAM files, we already see a drop in the recall rate to 0.80. As we increase the number of words, the recall rate decreases to 0.49 when injecting three non-CSAM words into a file path. Logistic regression boosted trees, and BERT models also presented a reduction in recall rates by adding non-CSAM words. Even though we see a decrease in recall rates in all models, CNN and LSTM models showed the most stable behavior under this attack.

By exploring multiple attacks and evaluating the performance of different models under these attacks, we can identify which learning techniques produce models sensitive to the attacks, such as logistic regression and boosted trees, and which models are more resilient, such as n-gram naive Bayes and CNN models. Although the attacks explored in our experiments were primarily aimed at file paths containing English words and sequences of alphanumerical characters, adapting the proposed attacks to include other alphabets and special characters is straightforward. We leave the improvement of the performance of models under non-CSAM word injection attacks and attack generalization to multiple languages as a future extension of this work.

6 Model Evaluation with File Paths from Common Crawl

Our training data set perfectly represents the deployment scenario for model deployment: our model is currently used to identify CSAM files in apprehended hard drives. Our training data comprises positive and negative examples from apprehended hard drives. The training data contains only file paths from hard drives that were *suspect* in the first place.

The Common Crawl dataset, containing text data from publicly available web sources, is used in our experiment to evaluate our models' false positive rate. We extract file paths from this dataset[9], assuming they are benign. It's important to understand our models' performance with 'suspect' hard drives and benign file paths. The false positive rate (FPR) is

9. https://commoncrawl.org

TABLE 3: Recall model performance evaluation in the presence of adversarial examples. We evaluate changes in the recall rate of several machine learning models under the following attacks: random character replacement, homoglyph replacement, synonym replacement, CSAM word spacing, and non-CSAM word injection. For all experiments, we report the average recall.

	Character replacement						non-CSAM word injection		
Model	10%	15%	20%	homoglyph	synonym	spacing	one word	two words	three words
BoW Logistic Regression	0.75	0.66	0.54	0.61	0.81	0.62	0.78	0.71	0.63
BoW Naive Bayes	0.85	0.83	0.80	0.81	0.90	0.81	0.80	0.68	0.58
BoW Boosted Trees	0.66	0.55	0.43	0.42	0.85	0.42	0.78	0.71	0.69
N-grams Logistic Regression	0.78	0.74	0.69	0.64	0.83	0.63	0.80	0.73	0.65
N-grams Naive Bayes	**0.92**	**0.93**	**0.94**	**0.88**	**0.92**	**0.88**	0.80	0.64	0.49
N-grams Boosted Trees	0.69	0.61	0.53	0.40	0.85	0.39	0.80	0.73	0.66
Character-based CNN	0.86	0.83	0.79	0.71	0.91	0.80	**0.89**	**0.86**	**0.84**
Character-based LSTM	0.82	0.79	0.76	0.76	0.90	0.77	0.87	0.84	0.81
BERT	0.85	0.83	0.82	0.84	0.91	0.77	0.83	0.74	0.65

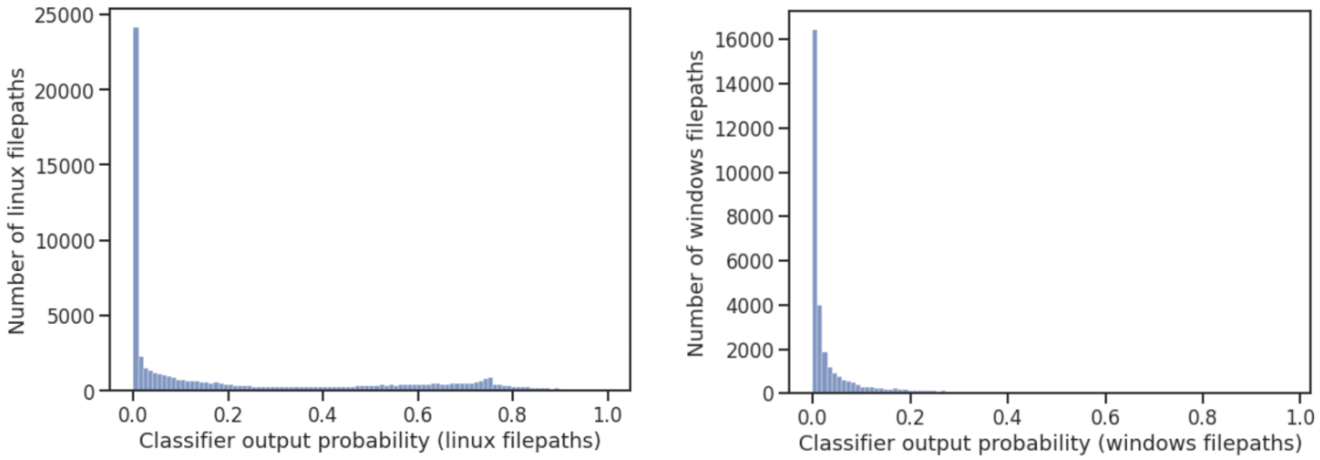

Fig. 5: Model confidence scores when evaluating file paths from the common crawl data set. Our best-performing model, a CNN-based model, exhibits a low number of files with a confidence score of over 0.2. The false positive rate (FPR) is 0.03 for a confidence threshold of 0.5 but achieves an FPR of 0.002 for a confidence threshold of 0.9. The model presents a higher FPR on Linux file paths, where at a confidence level of 0.5, it exhibits an FPR of 0.24. However, it drops significantly for a higher confidence threshold, achieving an FPR of 0.008 at a confidence level 0.9. At this confidence level, out of 73k file paths from the Linux set, only 584 would be identified as CSAM by our model.

our metric, as it triggers human data review. A poorly understood or calibrated FPR can burden content moderators and risk system deployment.

6.1 Common Crawl data set

We constructed a data set of benign samples using the publicly available common crawl data set. We collected data from the Common Crawl index CC-MAIN-2021-10. The WARC files utilized to construct our data set are:

- Linux file paths: we parsed the first 200 WARCs (00000-00199, inclusive), resulting in over 73k unique paths.
- Windows file paths: we parsed 11821 WARCS (00000-12000, inclusive), resulting in 32K unique paths.

We parsed the raw HTML, treating it as a Latin-encoded string. In each HTML, regular expression functions for identifying Windows and Linux file paths are the following:

```
Windows_file path_with_ext = r"([a-z]:\\
([a-z0-9() ]*\\)*[a-z0-9()]*\.(jpg|jpeg|
png|gif|mp4|mov|m4a|m4v|mpg|mpeg|wmv|avi|
flv|3gp|3gpp|3g2|3gp2|doc|docx|xls|xlsx|
ppt|pptx|pdf))"

Linux_file path_with_ext = r"(/([a-zA-Z0-9
()]*/)*[a-zA-Z0-9()]*\.(jpg|jpeg|png|gif|
```

```
mp4|mov|m4a|m4v|mpg|mpeg|wmv|avi|flv|3gp|
3gpp|3g2|3gp2|doc|docx|xls|xlsx|ppt|pptx|
pdf))"
```

After collecting the data set using the functions above, we filtered Windows file names to exclude ":\u002F". In Linux file names, we only keep the paths that begin with: /usr/, /home, /etc, /tmp, and /var.

6.2 Analysis

Evaluating model performance in independent data sets is essential to understand model generalization. We test our best-performing model against a data set containing only benign file paths. We measured the false positive rate for the best-performing model, CNN, at different confidence thresholds.

As we can observe from Figure 5, there is a small number of files for which the model attributes a confidence score above 0.8. For example, for Linux file paths, a decision threshold of 0.8 results in an FPR of $\approx$ 0.03, whereas a decision threshold of 0.95 results in an FPR of $\approx$ 0.001. For Windows file paths, a threshold of 0.8 prompts an FPR less than 0.01, while a decision threshold of 0.95 leads to an FPR less than 0.001. High decision thresholds are common design choices in detection systems, where only the high-confidence samples are flagged and sent for human review.

Based on this evaluation, we recommend carefully choosing the model threshold when using the model in general scenarios, i.e., scenarios where file paths do not come exclusively from suspect hard drives. The volume of data to be analyzed is also a decision factor when defining the model threshold.

7 Conclusion

This paper proposes a novel CSAM detection framework consisting of

- Machine learning models trained on file paths extracted from a real-world data set containing over 1 million file paths obtained in criminal investigations.
- Guidelines for model evaluation that account for data changes caused by adversarial data modification and variations in data distribution caused by limited access to training data.
- An assessment of false positive rates against file paths from common crawl data.

We highlight the framework's model assessment that accounts for adversarial interaction and changes in data distribution at test time. Previous works focus on proposals of model architecture and evaluate proposed models on data originating from the same distribution as the training data and do not consider potential data changes after deployment. Our work is the first in the CSAM literature to propose a model evaluation framework.

Because the proposed models utilize only file paths for CSAM identification, it avoids direct handling of CSAM content. This results in an easy-to-maintain detector with fewer legal restrictions for training data collection.

Our best classifiers achieve precision and recall rates over 0.90 in out-of-sample hard drives and maintain a low False Positive Rate (FPR).

It is also noteworthy how well the models can handle adversarial attacks and data distribution changes at test time. We evaluated our models under five distinct adversarial attacks, and we used Common Crawl data to assess the performance of our model on distinct data distributions.

Finally, by assessing the false positive rates of our models against file paths from common crawl data, we show that our proposed character-based Convolutional Neural Networks (CNNs) significantly reduce human evaluation needs, reducing the occurrence of false alarms.

Used alongside tools like PhotoDNA hash and computer vision techniques, our CSAM file path classifier creates a comprehensive toolset helping organizations combat CSAM distribution.

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] 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.

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

[7] Rob Dawson. Homoglyhps. *Codebox GitHub Repository*, 2022. https://github.com/codebox/homoglyph, Last accessed on 2023-01-30.

[8] Perry Deng, Cooper Linsky, and Matthew Wright. Weaponizing unicodes with deep learning-identifying homoglyphs with weakly labeled data. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Xiaoyu Du and Mark Scanlon. Methodology for the automated metadata-based classification of incriminating digital forensic artefacts. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–8, 2019.

[11] David Finkelhor, Heather Turner, and Deirdre Colburn. Prevalence of online sexual offenses against children in the us. *JAMA network open*, 5(10):e2234471–e2234471, 2022.

[12] National Center for Missing Exploited Children. Cybertipline 2022 report. Last accessed on 2023-09-23.

[13] Internet Watch Foundation. Cthe annual report 2021. Last accessed on 2023-09-23.

[14] Raphaël Fournier, Thibault Cholez, Matthieu Latapy, Isabelle Chrisment, Clémence Magnien, Olivier Festor, and Ivan Daniloff. Comparing pedophile activity in different p2p systems. *Social Sciences*, 3(3):314–325, 2014.

[15] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[16] 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.

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

[18] 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.

[19] Nora Hofer, Pascal Schöttle, Alexander Rietzler, and Sebastian Stabinger. Adversarial examples against a bert absa model–fooling bert with l33t, misspellign, and punctuation. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–6, 2021.

[20] Francisco Jáñez-Martino, Rocío Alaiz-Rodríguez, Víctor González-Castro, Eduardo Fidalgo, and Enrique Alegre. A review of spam email detection: analysis of spammer strategies and the dataset shift problem. *Artificial Intelligence Review*, pages 1–29, 2022.

[21] 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.

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

[23] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

[24] 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.

[25] Bhargav Kuchipudi, Ravi Teja Nannapaneni, and Qi Liao. Adversarial machine learning for spam filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–6, 2020.

[26] Camila Laranjeira da Silva, João Macedo, Sandra Avila, and Jefersson dos Santos. Seeing without looking: Analysis pipeline for child sexual abuse datasets. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 2189–2205, 2022.

[27] Matthieu Latapy, Clémence Magnien, and Raphaël Fournier. Quantifying paedophile activity in a large p2p system. *Information Processing & Management*, 49(1):248–263, 2013.

[28] 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.

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

[30] A. Y. Ng and M. I. Jordan. On discriminative vs. denerative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems 14, Proceedings of the 2001 NIPS conference*, pages 841–848. MIT Press, 2001.

[31] Vuong M Ngo, Christina Thorpe, Cach N Dang, and Susan Mckeever. Investigation, detection and prevention of online child sexual abuse materials: A comprehensive survey. In *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 707–713. IEEE, 2022.

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

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

[34] Alexander Panchenko, Richard Beaufort, Hubert Naets, and Cédrick Fairon. Towards detection of child sexual abuse media: categorization of the associated filenames. In *Advances in Information Retrieval: 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings 35*, pages 776–779. Springer, 2013.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

[45] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Detecting homoglyph attacks with a siamese neural network. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 22–28. IEEE, 2018.

[46] Jessica Woodhams, Juliane A. Kloess, Brendan Jose, and Catherine E. Hamilton-Giachritsis. Characteristics and behaviors of anonymous users of dark web platforms suspected of child sexual offenses. *Frontiers in Psychology*, 12, 2021.

[47] 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.

[48] 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.

Mayana Pereira Mayana Pereira is a Senior Data Scientist at Microsoft AI for Good Research Lab. Mayana is currently working towards her PhD degree in Electrical Engineering at the University of Brasilia. Her research is currently focused on the intersection of digital safety/cybersecurity/software security and artificial intelligence, as well as the impacts of privacy-preserving techniques in machine learning.

Rahul Dodhia Rahul Dodhia is the deputy director of the AI for Good Lab Microsoft. With his team of AI research scientists, he focuses on building AI solutions for the world's most pressing problems, with a focus on the Global South. After getting his PhD from Columbia University, where his research centered on mathematical psychology, he moved to the NASA Ames Research Center to continue work on hu[1]man memory and decision making.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3324275

Hyrum Anderson Hyrum Anderson is CTO at Robust Intelligence. He received his PhD in Electrical Engineering from University of Washington, with an emphasis on signal processing and machine learning, and BS and MS degrees in Electrical Engineering from Brigham Young University. Much of his career has been focused on defense and security, having directed research projects at MIT Lincoln Laboratory, Sandia National Laboratories, Mandiant, as Chief Scientist at Endgame (acquired by Elastic), and Principal Architect of Trustworthy Machine Learning at Microsoft.

Richard Brown Richard W. Brown is the Co-Founder and Director for a domestic / international child rescue organization called ”Project VIC International”. Project VIC champions a transformation in the approach to child exploitation investigations internationally by developing and adopting innovative technologies and victim-centric forensic workflows. Project VIC’s global outreach has resulted in a sharp increase in law enforcement’s ability to detect illegal video and imagery as it traverses global networks. Richard has had a long career in the areas of intelligence / law enforcement networks in first and developing countries. Rich’s work with the State Department, Homeland Security, United Nations, ICAC’s UNICEF, UNODC, FBI International and others have created opportunities to assist developing countries by donating and licensing technologies and training their specialists. Within the New Jersey State Police, he has held positions in the High Technology Crimes Unit, Electronic Surveillance Unit, Training Bureau, Internal Affairs, and retired as Chief of Intelligence Management for New Jersey. Most of his 25 years of law enforcement have been in the capacity of serving technology focused investigations or support of such investigations.