Alam and Demir *Cybersecurity (2024)7:52*
https://doi.org/10.1186/s42400-024-00241-9

Cybersecurity

RESEARCH

Open Access

SIFT: Sifting file types—application of explainable artificial intelligence in cyber forensics

Shahid Alam[1*] and Alper Kamil Demir[2]

Abstract

Artificial Intelligence (AI) is being applied to improve the efficiency of software systems used in various domains, especially in the health and forensic sciences. Explainable AI (XAI) is one of the fields of AI that interprets and explains the methods used in AI. One of the techniques used in XAI to provide such interpretations is by computing the relevance of the input features to the output of an AI model. File fragment classification is one of the vital issues of file carving in Cyber Forensics (CF) and becomes challenging when the filesystem *metadata is missing*. Other major challenges it faces are: *proliferation of file formats*, *file embeddings*, *automation*, We leverage and utilize interpretations provided by XAI to optimize the classification of file fragments and propose a novel sifting approach, named SIFT (Sifting File Types). SIFT employs TF-IDF to assign weight to a byte (feature), which is used to select features from a file fragment. Threshold-based LIME and SHAP (the two XAI techniques) feature relevance values are computed for the selected features to optimize file fragment classification. To improve multinomial classification, a Multilayer Perceptron model is developed and optimized with five hidden layers, each layer with $i \times n$ neurons, where i = the layer number and n = the total number of classes in the dataset. When tested with 47,482 samples of 20 file types (classes), SIFT achieves a detection rate of 82.1% and outperforms the other state-of-the-art techniques by at least 10%. To the best of our knowledge, this is the first effort of applying XAI in CF for optimizing file fragment classification.

Keywords Explainable artificial intelligence, Deep learning, Cyber forensics, File fragment classification

Introduction and motivation

Cyber Forensics (CF) is the science of gathering digital testimony to inspect traces of cybercrimes and cyberattacks. File carving is one of the most important processes of CF that covers the identification, preservation, and extraction of files from intentionally or unintentionally corrupted or compromised data storage devices (Boiko et al. 2023). Often, cybercriminals attempt to erase any evidence that prosecutes them. For example, they format the data storage devices. As a result, in such cases, the traditional reconstruction approaches based on file system meta-data fail, unfortunately. Cyber investigators generally resolve this challenge by file carving that reproduces files from data storage devices based on the raw content type. The file carving process requires recovering the corrupted file from fragments of raw binary files without using meta-data that forms and utilizes the base of the file system during routine operation. Thereafter, file fragment classification (also known as file fragment type identification (Mittal et al. 2020)) is an essential step in file carving due to the increasing need for sifting file types in the presence of law enforcement investigations of data storage devices (Skračić et al. 2023; Ghaleb et al. 2023; Haque and Tozal 2022).

*Correspondence:
Shahid Alam
sha.alam@uoh.edu.sa
[1] Department of Information Security, College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia
[2] Computer Engineering Department, Adana Alparslan Turkes Science and Technology University, Adana, Turkey

The physical and data structure and logical rules to store, manage, and retrieve the fragments of raw binary data and their file names on a storage device are called file systems. To express the files, the file system contains meta-data (or meta-information) that provides information about the actual data. Therefore, the file system will also preserve the physical locations of the file fragments on data storage devices. Principally, a file system allocates the first several sectors (disc blocks) of a data storage device to store the meta-data indicating the overall data storage space, file attributes, and their formation. The remaining sectors keep the raw binary (real content) of the files. A sector is the indivisible physical data unit on a data storage device with a typical size of 512 or 4096 bytes. As a deduction, a file might be spread in fragments at distinct sectors having different physical addresses on a data storage device.

Due to data storage and file system failures, formatting, or erasing the evidence on data storage devices deliberately, the meta-data of a file system may be unavailable on a storage device. File carving becomes useful in such cases to rescue files on a data storage device in a piece or complete without the existence of meta-data. This could be achieved by analyzing and classifying the raw binary data of file fragments stored and located at sectors of the data storage device. After the file fragment types are classified, ordering and merging the relevant file fragments procedure to resurrect the initial file(s) is applied as the next step. As a consequence, it is essential to design and develop automated methods and tools for accurate file fragment classification. In our work, we propose, present, evaluate, and detail an AI-based approach, specifically leveraging a new emerging subdomain of AI, called Explainable Artificial Intelligence.

The rise of AI as a disruptive technology has been revolutionizing the world recently (Păvăloaia and Necula 2023). However, AI's becoming more competent and being used in very critical decisions with expressive human intervention is increasingly bringing trust issues essentially (Kaplan et al. 2023; Langer et al. 2023). Naturally, humans are required to understand, reproduce, and manipulate the decision-making processes of AI systems. As a result, there is an increasing need to expose the decision-making processes of AI systems so that it is more straightforward, understandable, and explainable. Correspondingly, Explainable Artificial Intelligence (XAI) is a new and prominent research domain intended to self-explain the reasoning behind decisions and predictions of AI systems (Hassija et al. 2023; Ali et al. 2023). XAI hopes to help users of AI-powered systems appear more understandable and transparent. Traditional AI systems seem to be the *black box* where even the designers can not explain why the AI system provided the conclusive decision. Explanation clarifies the decisions made by a black-box model where it is more intuitive for humans. Moreover, an explanation of the decisions made increases the potential reliability of the AI systems for final agreement.

Traditionally, XAI involves explaining or interpreting the predictions of recently developed Deep Learning (DL) models using diverse rule-based and visualization-based techniques (Kaur et al. 2022; Vilone and Longo 2021). Thereby, advances in theory, applications, and trends in XAI have been discovering and developing computational approaches in the XAI domain for these AI models recently (Górriz et al. 2023). In general, from an explainability point of view, these XAI techniques can be divided into three dimensions using a categorization system: (i) data explainability, (ii) model explainability, (iii) post-hoc explainability along with assessment of explanations axes (Ali et al. 2023). Every dimension of elucidating or revealing the decision-making mechanisms of AI systems plays an important role in explainablity. Data explainability compiles and study data to offer insight into that data. Model explainability spells out the internal structures and running algorithms of the AI systems. On the other hand, post-hoc explainability refers to methods used to explain the decision of the AI system. For example, post-hoc explainability illuminates the significant features using several kinds of explanation for the outcome of the AI model. Furthermore, several assessment methods and their suspicions can be used to evaluate the explanations. In this study, we utilized two post-hoc explainability techniques, namely LIME (Ribeiro et al. 2016) and SHAP (Lundberg and Lee 2017) for file fragment classification.

Various post-hoc explainability methods are classified into six important groups: (i) attribution methods, (ii) visualization methods, (iii) example-based explanation methods, (iv) game theory methods, (v) knowledge extraction methods, and (vi) neural methods (Ali et al. 2023). LIME is a class of attribution methods, whereas SHAP is a class of game theory methods. A multitude number of attribution methods depends on pixel corporation to investigate which pixel of a training input image is relevant from the standpoint of model activating assuredly in the context of image processing. The kind of analysis achieved can be categorized as either a local or a global method. Local explainers only explain a specific decision whereas global explainers are those that give a rationale for the whole datasets (Adadi and Berrada 2018). The major reasons for using LIME and SHAP in this paper are that they provide local explanations and are model-agnostic.

Most of the previous research on file fragment classification can be categorized into (i) signature, (ii) statistical, (ii) artificial intelligence (AI), or (iv) hybrid-based landscape (Sester et al. 2021). The file type is related by a signature that has unique, individualistic, and evidentiary attributes related to a file type. Comparison of known to unknown file fragment classification methods are applied in signature-based approaches. Statistical techniques are leveraged in the second class of approaches by utilizing the characteristic features of the file content. On the other hand, AI-based techniques principally utilize computational intelligence such as machines and models. Hybrid-based techniques apply an ensemble of these three techniques.

In this study, we urge a novel AI-based file fragment classification method. At first, we preprocess the files in the dataset to part the file fragments and basic raw features of them. Afterward, the Term Frequency and Inverse Document Frequency (TF-IDF) (Manning et al. 2010) technique is applied for feature selection. Specifically, each raw feature is designated with a weight depending on its TF-IDF, and the features holding positive weights are chosen. Then we apply two XAI techniques, LIME and SHAP, to gather the most decisive (relevant) features among the selected features. Finally, these relevant features are used to train and test a Multilayer Perceptron (MLP) (Hornik et al. 1989) classifier to categorize the file fragments into file types. MLP is the most common and practical model (Heidari et al. 2016). The results show that this approach is feasible and able to achieve better outcomes.

The major differences between the techniques explored in this paper and other AI-based works (Haque and Tozal 2022; Bhatt et al. 2020; Chen et al. 2018; Wang et al. 2018) are (1) Lossless feature extraction. (2) Adaption of TF-IDF and two XAI techniques LIME and SHAP to estimate inter-Classes and intra-Classes information gain of a feature. Given these new revisions, our study harvests encouraging results as regards other works. There are only two research works (Mahajan et al. 2021; Hall et al. 2022) that use LIME to explain the predictions of an AI model in CF but don't use such explanation (LIME values) to optimize the predictions/classification. To the best of our knowledge, this is the first effort of applying XAI within CF for optimizing file fragment classification.

The following are the major contributions of this paper:

- We propose a novel method, named SIFT, to classify file fragments in the absence of metadata of the filesystem. TF-IDF technique and two XAI techniques, LIME and SHAP, are enriched as a feature selection and relevance, and multinomial classification with an MLP model is leveraged to train and test the effectiveness of SIFT.
- We randomly selected 20 file types, from a publicly available and more standardized dataset for cyber forensics research, and extracted 47,482 samples (fragments) from them. To keep the evaluation unbiased we selected 7 files from each file type and chose 512 bytes as the fragment size. We chose three state-of-the-art works in this area to compare SIFT with them. We observe that SIFT produces promising (better) results by at least 10%.

The rest of the paper is organized as follows. Section "Background" gives background information. Section "SIFT—system overview" describes the proposed method, SIFT, along with preprocessing, feature extraction, LIME and SHAP feature relevance, feature selection, and multinomial classification steps. Empirical evaluation with dataset collection, evaluation metrics, validation results with discussion, and comparative results are presented in Section "Empirical evaluation". Section "Related work" details the related work. In Section "Limitations and future work", limitations and future work are presented. Finally, Section "Conclusion" concludes our work.

Background

To make the reader familiar with the research presented in this paper here we provide some background on XAI, CF and TF-IDF.

Explainable artificial intelligence (XAI)

XAI, a subdomain of AI, targets transforming complex *black box* data, models, and decisions of AI algorithms and systems into easily explainable and evaluative notations and methods (Schwalbe and Finzel 2023; Saeed and Omlin 2023; Vilone and Longo 2021). A vast amount of techniques to deal with this issue have been proposed, developed, and tested, striving to specify the concept of explainability and its evaluation. XAI leads the users of AI systems to trust. Comprehending and accepting the decision process of an AI system is especially important in high-risk tasks for humans (Leichtmann et al. 2023). XAI aims to advance the AI literacy of humans. While AI literacy is tough to characterize, XAI seeks to describe the complex construct of AI systems by investigating various computational methods to satisfy the cognitive abilities of humans. While developing explainable methods, it is also important to involve techniques to measure goodness, satisfaction, mental models, curiosity, trust, and human-AI performance in the context of XAI (Hoffman et al. 2023). In brief, XAI explores methods that can provide clear, verifiable, and trustworthy explanations of decision-making processes of AI systems, bringing experts from various disciplines, including computer science, psychology, philosophy, and ethics.

Recently, XAI has been penetrating various application domains and tasks of AI systems (Islam et al. 2022). Cyber security is also one of the promising application domains of XAIs (Capuano et al. 2022). Application of XAI in the cyber security field broadly ranges from Intrusion Detection Systems to Malware detection, Phishing and Spam detection, botnet detection, Fraud detection, Zero-Day vulnerabilities, Digital Forensics, CryptoJacking, etc. A fresh field of Cyber security that requires exposing the XAI is file carving (Saxena et al. 2023; Dunsin et al. 2023). File fragment classification is one of the most important steps of file carving. Therefore, we study the application of XAI to file fragment classification problems of Cyber security with LIME and SHAP techniques of the XAI, freshly minted. To the best of our knowledge, this is the first initial study in the literature for such a problem.

LIME first generates a dataset of perturbed data points, then calculates the sample weights using a kernel function and a distance function to calculate how far the sampled points are from the original point. It then uses a surrogate model (interpretable model) on the perturbed dataset using the sample weights. This trained model is then used to provide explanations (including LIME values) for each instance.

SHAP calculates the Shapley values for each feature of the dataset used to train and test the AI model that is to be interpreted. These values represent the impact of the feature in generating the prediction/output delivered by the AI model. Shapley values borrow the concept of the game theory field where the objective of the values is the contribution of each player to the game. One of the explainers available in SHAP is the TreeExplainer which is being used in SIFT. The TreeExplainer takes as an input a tree model such as RandomForest and DecisionTree etc, and uses the conditional expectation to estimate the effects and computes the Shapley values.

Cyber forensics (CF)

Every contact by a perpetrator leaves behind traces (Chisum and Turvey 2000). To make a case against the perpetrator these traces or pieces of evidence need to be found, collected, secured, studied, and analyzed. *Cyber forensics* (CF) (Alam 2022) uses scientific methods and expertise to gather and analyze pieces of evidence found in cyber devices that can be used in criminal or other investigations in a court of law. This evidence can be used for different purposes, such as *electronic discovery*, *intelligence*, and *administrative*. For example, the data collected from cyber devices can provide actionable intelligence. This intelligence can help accomplish different types of missions, such as securing national interests, decreasing or eliminating crimes like kidnapping and child exploitation, etc. Electronic discovery is the process of searching, finding, and securing any electronic data later to be used in a civil or criminal forensic case.

File system

A storage media or device stores information as blocks of raw data (bytes). There is no particular organization or access control to this raw data. A block or sector is the smallest storage unit on a device with a typical size of 512 or 4096 bytes. Filesystem organizes this raw data into files and folders for ease of management, storage, and retrieval of information. The first few sectors of a file: contain meta information, such as owner, size access rights, and creation time about files, and keep information about the overall storage space, files, and organization. The remaining sectors store the actual content of the files. A generally high-level structure of a storage device is shown in Fig. 1. The boot block mostly contains the information to boot the device. A superblock is the metadata repository. File system data structures keep information about the files and their data blocks. Data blocks contain the actual contents of the files. It is not necessary for data blocks belonging to a file to be contiguous. For example, the first two blocks of `a.pdf` are followed by one block of `b.png`, an unused block, one block of `a.pdf`, and one block of `a.png`, and so on.

File carving

In cyber forensics *File Carving* (Alam 2022) is the process of mining and extracting files from a storage device. In general, files are present in the form of raw bytes, i.e., there is no metadata information available about the files, and the filesystem that created the files is damaged. A file is generally identified by the header. A fragmented

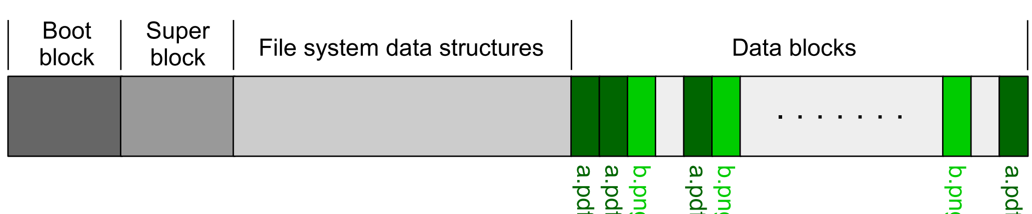

Fig. 1 High-level structure of a storage device

file is much more difficult to extract than a continuous file. A file is stored and retrieved as blocks (fragments – can be of size 512 or 4096 bytes) of raw bytes. To make files portable across different platforms files are encoded in standard formats. For example, a PNG file type stores bitmap images using lossless compression. To successfully extract a file from a storage device it is necessary to identify the fragment types of the file. After the fragment types are identified, the next step is to reconstruct the file by properly merging the fragments.

File fragment classification

The main focus of the research done in this paper is to assist *File Carving* by successfully identifying fragment types of the file, also called *File Fragment Classification*. Every file is encoded in a standard format or type, such as DOC, HTML, PDF, SWF, PNG, GIF, XML, etc. The type of a fragment extracted from a file is the same as the type of the file. The problem of fragment classification is: *to successfully classify a fragment out of several different types* (*classes*) *of fragments*. This is the first and foremost step during file carving. There are different methods used for fragment classification, including signature-based (McDaniel and Heydari 2003; Thi et al. 2017; Garfinkel 2006; Garfinkel et al. 2010; Garfinkel and McCarrin 2015), statistical (McDaniel and Heydari 2003; Dhanalakshmi and Chellappan 2009; Beebe et al. 2016), machine learning (Axelsson 2010; Conti et al. 2010b; Li et al. 2011; Veenman 2007; Conti et al. 2010a; Bhatt et al. 2020), and image-based (Xu et al. 2014).

Fragment classification challenges

Classifying a file fragment successfully is a challenging task because of the following reasons.

1. *Missing Metadata*—A filesystem contains metadata that expresses the actual filesystem and contains information about the location of fragments and attributes of each file, etc. If this metadata is missing due to damage to the device or format operations, then it becomes challenging to recover files on a storage device for forensics analysis.
2. *Proliferation of file formats*—Too many file types (each type is taken as a class) make it difficult to classify and lead to a multinomial classification problem. It becomes difficult to put a lower and upper bound for distances between classes required for successful feature selection and classification with a given accuracy. Class imbalance problem arises when data across the classes are imbalanced. This problem is aggravated when carrying out multinomial classification.
3. *File embeddings*—An image is generally first compressed and then stored. Therefore, part (block) of a zipped (compressed) file may contain similar patterns as an image file, especially if they are using the same compression types. During classification, this may make block(s) of a zipped file get detected/classified as an image file type and vice versa. SWF is an Adobe file format and may contain images to create animations. Such file types may also contain similar patterns as an image file and hence a block of an SWF file may get detected as a block from an image file and vice versa. The same is true for PDF and PPT (Microsoft PowerPoint) file types containing embedded images.
4. *Automation*—A digital forensic and incidence response professional can look through (using a hex editor) a piece of binary data and identify the type of data it carries. This requires experience which can be very helpful in various forensic tasks, such as decoding memory dumps, reverse engineering malware, data recovery, and so on. The main problem with manual examination is that it does not scale. Therefore, we need automated tools to perform fragment classification. Such a tool: should be accurate; and fast enough to handle large data; the error rates should be reliable; and should produce clear results.

Term frequency and inverse document frequency (TF-IDF)

Term Frequency (TF) is the relative frequency of a term in a document. Inverse Document Frequency (IDF) is the measure of the commonality or rarity of a term across all documents. The product of TF and IDF is used to assign weight to the term. This weight indicates the importance of the term in the corpus (set of documents). In information retrieval TF-IDF measures how important a term is inside a document with respect to a corpus.

The TF-IDF algorithm was first proposed by Sparck (1972) and consists of the following three items.

- $TF(t, d) \rightarrow$ number of times the term t occurs in document d
- $N \rightarrow$ total number of documents in a given corpus D
- $DF(t) \rightarrow$ number of documents containing the term t

The TF-IDF of the term t is computed as

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

and

$$IDF(t, D) = \log \frac{N}{DF(t)}$$

As an example usage, for identifying keywords we take into account not only how many times a keyword occurs in a document but also how frequently the keyword occurs in other documents in a corpus. For retrieving keywords, we can use the above algorithm by selecting the keyword t with the largest value of TF-IDF in a given corpus D. In the next iteration, we select the keyword with the next largest value of TF-IDF and so on.

SIFT—system overview

SIFT extracts fragments from the dataset with their raw features. These features are then sifted through to select the most important features. Sifting examines thoroughly to isolate the most important features by using statistical and XAI techniques. TF-IDF is used to assign weight to each feature according to its importance. Two of the popular techniques of XAI, SHAP, and LIME, are used to find the most relevant features. These weighted and relevant features are then used by a classifier to classify the file types of the fragments. Figure 2 shows a high-level component overview of the proposed system SIFT. The following sections further explain each of these components in detail.

Preprocessing and feature extraction

SIFT first reads each file in a dataset and then preprocesses the file as follows. It excludes files with size < 2 × fragment size and also removes the duplicate files. After this fragments are extracted at the byte level from each file. Each fragment is of the same size S. To cater to resource-constrained devices such as embedded systems and IoT, S can vary from 2^5–2^{12} = { 32, 64, 128, 256, 512, 1024, 2048, 4096 }. These devices generally store information in a flash ROM whose size is in the kilobyte range. The data from this flash ROM is transmitted to the edge/gateway/cloud to be stored for later use. The sector sizes of the filesystem, e.g., FAT 12/16—TinyOS (TinyOS 2023), for these flash ROMs typically range from 32–128 bytes.

Figure 3 shows an example of a fragment extracted from one of the dataset files used in this paper. The left column shows the address/location (in decimal), and

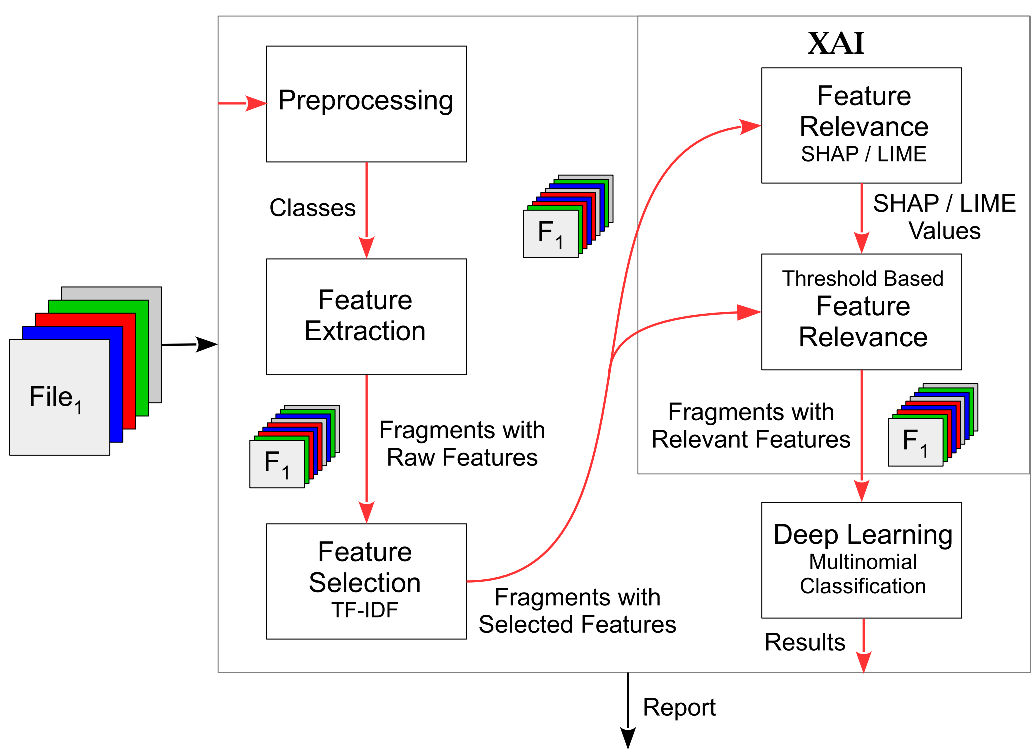

Fig. 2 Overview of the proposed system SIFT

```
4096 20 B9 5C 8F 49 26 B1 77 8F 49 C6 D6 9E 3B DB 31 A0 F0 84 96 7C 92 A1 8D 2A C7 24 33 D3 1B F7 28
4128 C3 5C 04 D0 DB 46 99 19 47 3D 89 C2 9E 82 FA 26 51 A2 C6 21 1C 7A 95 28 59 55 59 91 2B 25 CA 4C
4160 1E 01 D2 17 50 90 5E BA 8F D4 4C A0 E0 03 54 3A 53 95 0E 94 C4 B6 2D 3E C9 E0 CB 74 F0 04 16 46
4192 EA 27 4F D8 61 F4 DE 78 02 AF B5 0A C6 13 5B 7B 8F AF 1D E3 09 12 D3 90 28 E3 09 CE 41 0F 1C 3C
4224 81 01 96 D3 79 C2 6A 42 4D C6 93 32 A7 35 23 CA 5A 1E F6 6C C7 A8 52 18 6C 5D 58 49 D7 FB E6 FF
4256 F0 D9 F1 40 E2 81 C4 03 89 07 12 13 12 9F F5 F6 C0 03 FE 2E 48 FB D4 55 04 56 22 6A CC FC 99 AE
4288 22 A2 2A 3C 4F 75 15 7D BD 75 B5 77 96 AE F4 84 3C B6 AE D4 46 EA 87 AE 62 21 41 C6 D6 15 91 53
4320 F8 7C 5D BA A2 3A 7A 4E 87 AE B8 13 FB 96 15 46 1B 3C CA F3 96 15 4D 88 64 97 15 7D 08 78 49 9B
4352 AC 22 94 1C 71 A2 C9 8A 6B BE B4 5D 56 8C 22 E9 80 30 65 C5 13 3A 55 A3 BA 1A 6A 22 B3 6B 54 57
4384 63 45 91 F8 BC 5D BA 62 14 99 E5 56 5D 15 5F EF 3A DB CE D2 15 06 32 38 35 4C 57 B4 D8 D8 D1 A6
4416 2B 46 C9 34 6C 5D 29 9A A1 A0 A5 AB 58 33 C0 93 4C 57 B6 3C EC D9 CE D2 55 24 A1 24 2D 5D C5 99
4448 67 65 1C 75 65 95 49 04 E9 BA 54 18 75 B3 C7 30 FB 1A EB C8 7E DE B7 4A 2C 7A 39 E8 A5 D2 B4 30
4480 85 D9 5B 97 0A 2D 94 31 73 31 3D 9A 8A 9E 57 0A 3E AB 69 FE AE 4F 45 D0 63 D4 E8 B8 53 34 06 E4
4512 DF EE 14 8D A1 F7 7D A7 D0 C2 E8 F9 B8 53 34 06 7C 6B 77 0A 7D E6 1D 63 97 CA 5E BB 82 6D 67 5D
4544 2B 53 9F 6D 5F 2B 66 63 5E 2B 65 C5 D1 A0 06 BB 56 E8 25 35 6E D7 8A C5 B1 AF 15 06 3A 92 EC 6B
4576 25 A2 E8 1C 23 EC 5A 89 28 41 86 16 FC 5A 61 62 13 4C F8 B5 42 BD 0C A8 DF AE 15 4D 2E 74 60 D7
```

Fig. 3 Example of a fragment, of size = 512 bytes, extracted from one of the dataset files used in this paper

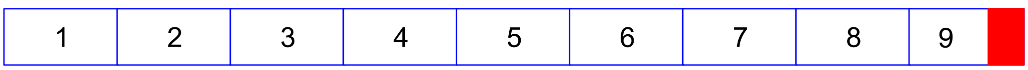

Fig. 4 Fragment extraction. As an example, there are 9 fragments of a file shown here. 8 are complete, whereas the last one is a partial fragment, filled to make it complete

the right column shows the byte value stored in hex. The size of the fragment shown in Fig. 3 is 4096–4608 = 512 bytes. There are only 256 different values at the byte level. Therefore, the byte value ranges from 0×00–0×FF. A total of S number of raw features are extracted for each fragment in a file. The first fragment contains the header information that identifies the file type, therefore SIFT excludes the first fragment of a file. The last fragment of a file may be of a different size. This last fragment is filled with bytes from a randomly chosen fragment of the file as shown in Fig. 4. This way we make sure that all the fragments of all the files are of equal size. These steps for extracting raw fragments from a list of files (dataset) are listed in Algorithms 1 and 2. Equation 1 formally defines this set of raw fragments.

Algorithm 1 Algorithm for extracting fragments from a list of files.

```
Pre–conditions: files (List of files); fragmentSize (Size of the fragment)
Post–conditions: Returns a dictionary of extracted fragments
procedure ExtractFragments(files, fragmentSize)
    fragments = dict()                         ; initialize the dictionary of fragments
    for all file ∈ files do                    ; iterate through the list of files
        fragments = ExtractFromFile(file.contents(), fragmentSize) ; extract fragments from the file
        samples[file.name] = fragments         ; store the extracted fragments under the filename
    end for
    return fragments                           ; return the dictionary of fragments
end procedure
```

Algorithm 2 Algorithm for extracting fragments from a file.

```
Pre–conditions: fragmentSize (Size of the fragment); fileContents (Contents of the file as bytes)
Post–conditions: Returns a list of extracted fragments
procedure ExtractFromFile(fileContents, fragmentSize)
    fileFragments = list()                          ; initialize the list of fragments
    c = n = fragmentSize                            ; c=start:n=end → start of the second fragment
    numberOfFragments = ⌊len(fileContents) / fragmentSize⌋   ; number of fragments in the fileContents
    for i = 1 to numberOfFragments do               ; extract these fragments from the fileContents
        n = n + fragmentSize                        ; reset the end pointer
        fragment = fileContents[c:n]                ; extract the slice (fragment)
        fileFragments.append(fragment)              ; append the fragment to the list of fragments
        c = n                                       ; reset the start pointer
    end for
    remainder = len(fileContents) % fragmentSize    ; calculate the size of any left over bytes
    if remainder > 0 then                           ; extract the left over bytes
        n = c + remainder                           ; c is start and now n is end of the left over bytes
        i = random(0, numberOfFragments - 1)        ; randomly select a fragment
        ac = i × fragmentSize                       ; start of the selected fragment
        an = ac + fragmentSize - remainder          ; end of the slice in the selected fragment
        fragment = fileContents[c:n] + fileContents[ac:an]   ; left over bytes + the slice
        fileFragments.append(fragment)              ; append the fragment to the list of fragments
    end if
    return fileFragments                            ; return the list of fragments
end procedure
```

We define a fragment of bytes as $f = \{ b_1, b_2, b_3, \ldots, b_S \}$, where S is the size of the fragment. Let $M = \{ m_1, m_2, m_3, \ldots, m_N \}$: where N = number of files in a dataset; and m_a is the number of fragments extracted from file a. A file in a dataset (set of files) is defined as $file = \{ f_1, f_2, f_3, \ldots, f_m \}$, where $m \in M$. Then, we define the set of fragments with extracted (raw) features F from a dataset as follows.

$$F = \bigcup_{i=1}^{N} \bigcup_{j=1}^{m \in M} \{f_{ij}, i\} \tag{1}$$

where f_{ij} is the jth fragment extracted from the ith file. We add an extra byte i at the end of each fragment to use as the Class (file type) label.

Feature selection

As shown in Fig. 3 a fragment of size S consists of S number of bytes whose value ranges from 0×00–0×FF. Therefore, for a fragment, we select a total of 256 features, and a weight is assigned to each of these bytes according to their importance in the fragment.

Term Frequency and Inverse Document Frequency (TF-IDF) (Manning et al. 2010) is often considered an empirical method in data mining to separate relevant features in a set of data. TF-IDF computes the information gain of a term (in our case a byte) weighted by its occurrence of probability. We explain in the following, how we adopt the TF-IDF weighting method and assign weight to a byte (feature). We define TF-IDF of a byte $b_j \in f$ as follows:

$$TF_j = \frac{fb_j}{R} \qquad and \qquad IDF_j = log\left(\frac{\sum_{n=1}^{N} m_n}{K_j}\right)$$

where, fb_j is the number of times (frequency) byte b_j appears in a fragment f; and K_j is the number of all the fragments with b_j in it.

Based on these definitions, we assign weight to a byte b_j as follows:

$$W_j = TF_j \times IDF_j \tag{2}$$

We build a vector of the fragments with selected features FS in the form of a matrix as follows.

$$FS = \bigcup_{i=1}^{N} \bigcup_{j=1}^{S+1} \begin{cases} W_j & \text{if } W_j > 0 \\ 0 & \text{otherwise} \\ i & \text{if } j = S+1 \end{cases} \tag{3}$$

The weight W_j ranges from 0–1. We only keep b_j if $W_j > 0$. For example, we noticed that the byte 0×FB occurs several times in many fragments of type (Class)

EPS, in the dataset used in this paper. A total of 20 Classes are part of the dataset used in this paper and are listed in Table 1. The byte 0×FB gets a score > 0.98 for Class EPS and mostly 0 or < 0.25 for the rest of the Classes. Similarly, the byte 0×30 gets a score > 0.98 for the Classes EPS, PS, and PDF, and mostly 0 or < 0.30 for the rest of the Classes. This indicates that the feature selection scheme we presented in Eq. 3 has the potential of successfully separating important features from the raw features computed in Eq. 1. This in turn helps a classifier correctly predict the Class (file type) of a fragment.

XAI—feature relevance

XAI is one of the branches of AI that interprets and explains the methods used in AI. One of the techniques used in XAI to provide such interpretations is by computing the relevance of the input features to the output of an AI model. SIFT uses LIME and SHAP, two popular XAI model-agnostic techniques to compute the relevance of input features to its output. Being model-agnostic, SHAP and LIME need to be initialized with the training and testing data. This training and testing data should be chosen from the dataset that is to be used to train and test the model of SIFT. We can either use the whole dataset or choose a part that is chosen randomly from the whole dataset.

LIME and SHAP feature relevance values

We initialize LIME with the following parameters: training data; testing data; features; class names; and Ridge Regression the interpretable model to be used as a surrogate. We initialize SHAP with the following parameters: training data; testing data; and RandomForest as the ensemble tree model. Then we use LIME and SHAP to compute the relevance value RV of a feature f, i.e., RV_f for each sample in the testing data. We compute the mean relevance value (also called LIME and SHAP value in this paper) LV (LIME value) and SV (SHAP value) of each feature f for a class c in the testing data as follows.

$$LV_{f,c} = \frac{\sum_{n=1}^{NS} RV_{f,n}}{NS}$$

and

$$SV_{f,c} = \frac{\sum_{n=1}^{NS} RV_{f,n}}{NS}$$

where NS = number of samples in class c and $RV_{f,n}$ is the relevance value of feature f in sample n.

Similarly, the LIME (LVs) and SHAP (SVs) values are computed for each feature in all the classes (in this paper for 20 classes) in the testing data as follows.

$$LVs = \bigcup_{f=1}^{NF} \bigcup_{c=1}^{NC} \{LV_{f,c}\} \tag{4}$$

and

$$SVs = \bigcup_{f=1}^{NF} \bigcup_{c=1}^{NC} \{SV_{f,c}\} \tag{5}$$

where NF = number of selected input features in the dataset and NC = number of classes in the dataset.

XAI—threshold-based feature relevance

The LIME and SHAP feature relevance values computed above are used to remove non-relevance features from the dataset based on a threshold. For each of the LIME and SHAP values separate thresholds, T_{lime} and T_{shap} respectively, are computed. The motivation behind these threshold values is to mark irrelevant features from the selected features and obtain the most relevant features for the dataset. Another motivation is to reduce the number of features and improve the computing time of the classification, which in the case of DL is a substantial improvement. These two thresholds are computed as follows.

We randomly choose a subset (part) of the dataset that we call D to compute the LIME and SHAP feature relevance values using the Eqs. 4 and 5. The higher value represents more relevance of an input feature to the model's output. To try different threshold values, we pick a range t_1–t_2 between the minimum and maximum relevance values (computed in Eqs. 4 and 5) that allow us to mark 10–30% irrelevant features from the selected features computed in Eq. 3. We divide D into 80% training and 20% testing and perform several experiments of fragment classification using a (in our case MLP) model by choosing different threshold values in the range t_1–t_2. We picked the threshold value that gave us the best results. This process was repeated for both LIME and SHAP and we get two threshold values T_{lime} for LIME values and T_{shap} for SHAP values.

Using the two threshold values T_{lime} and T_{shap} computed above we remove the irrelevant features from the dataset and get the set of final fragments with threshold-based relevant features for LIME (FRL) and SHAP (FRS) as follows:

$$FRL = \{v \mid v \in LVs \wedge v > T_{lime}\} \tag{6}$$

and

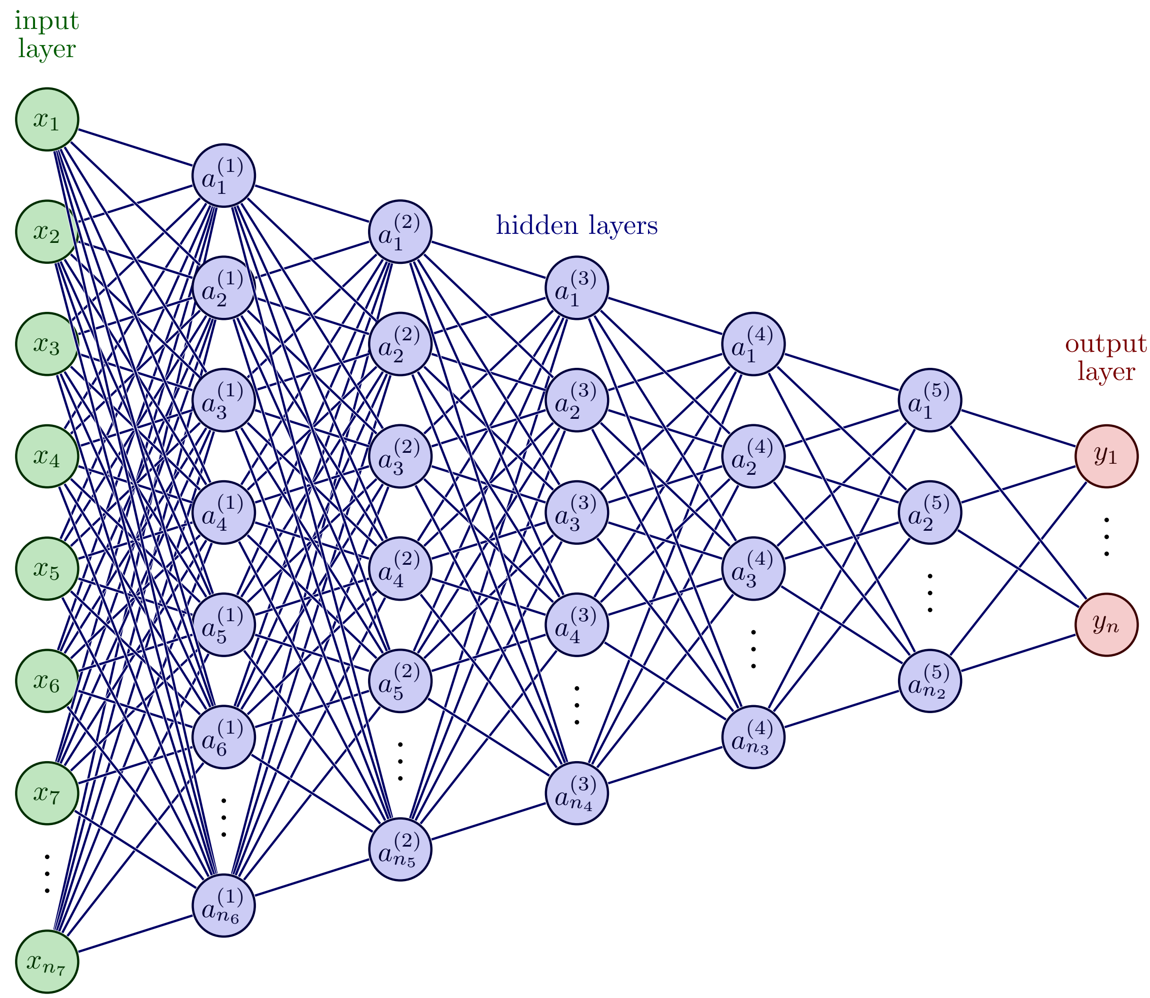

Fig. 5 A high-level architecture of the MLP artificial neural network developed in the paper, where n = total number of classes in the dataset, i = layer number, and $n_i = i \times n$

$$FRS = \{v \mid v \in SVs \wedge v > T_{shap}\} \tag{7}$$

This set of final fragments with threshold-based relevant features is used for training a model for classifying file fragments.

Multinomial classification with deep learning

One of the main challenges of file fragment classification is multinomial classification. The other main challenge is the class imbalance problem. The dataset used in this paper presents both of these challenges. To overcome these challenges to some extent we use a model for file fragment classification. We develop and build the model using *Multilayer Perceptron* (MLP) (Hornik et al. 1989), one of the most common and practical models (Heidari et al. 2016), with one input, one output, and five hidden layers. The developed MLP artificial neural network architecture is shown in Fig. 5. The reason for using five hidden layers is because of the complexity, such as the large number of file types (classes) and class imbalance, of the data found in most of the file fragment classification problems. To improve multinomial classification, the MLP model is optimized with hidden layers, each layer with $i \times n$ neurons, where i = the layer number and n = the total number of classes in the dataset. We use an adaptive learning rate algorithm *RMSProp* (Root Mean Square Propagation) (Hinton et al. 2012) for optimizing the learning process. RMSProp addresses some issues with the stochastic gradient descent method in training deep neural networks.

One of the major performance optimizations of neural networks is tuning the hyperparameters, such as the batch size, number of hidden layers, and number of epochs. The tuning depends on the type and complexity

Table 1 Fragment distribution of the 20 file types (Classes). There are a total of 47,482 fragments (samples) each of size 512 bytes

Classes (file type)	Number of files	Number of fragments
csv	7	889
dbase3	7	66
doc	7	2420
eps	7	5110
gif	7	701
gz	7	4470
jpg	7	924
html	7	613
kmz	7	1381
log	7	5346
pdf	7	2787
png	7	2704
ppt	7	3534
ps	7	3622
swf	7	1991
text	7	4671
txt	7	1374
unk	7	3264
xls	7	813
xml	7	802

Table 2 Results of 10-fold cross-validation of the RandomForest classifier

File type (class)	TPR	FPR	Precision	F-Measure
csv	0.991	0.000	0.991	0.995
dbase3	1.000	0.000	1.000	1.000
doc	0.724	0.008	0.724	0.776
eps	0.989	0.000	0.989	0.992
gif	0.357	0.000	0.357	0.524
gz	0.889	0.133	0.889	0.560
html	0.977	0.000	0.977	0.976
jpg	0.000	0.000	0.000	0.000
kmz	0.938	0.000	0.938	0.968
log	0.999	0.000	0.999	0.999
pdf	0.463	0.003	0.463	0.611
png	0.720	0.042	0.720	0.597
ppt	0.492	0.014	0.492	0.592
ps	0.996	0.002	0.996	0.988
swf	0.124	0.008	0.124	0.190
text	0.765	0.003	0.765	0.852
txt	0.932	0.000	0.932	0.960
unk	0.995	0.005	0.995	0.965
xls	0.812	0.001	0.812	0.875
xml	0.965	0.000	0.965	0.977
Weighted Avg	0.798	0.018	0.798	0.793

of the dataset. *Batch size* is the number of samples that are propagated through the network. It can be a subset or whole of the dataset. In the case of a subset, the whole dataset is divided into subsets, and with each iteration, each subset is propagated through the network until all the samples have been propagated. *Hidden layers* refers to a set of neurons, that makes neural networks deep and enable them to learn complex data representations. *Epochs* is the complete training of neural networks on all the datasets exactly once. The value of epochs can range from 1 to ∞. They are the fundamental part of the training of neural networks. We compute and set these parameters experimentally on a subset of the main dataset. We randomly chose a subset of the samples from the main dataset and trained the model using this subset. Then, for setting the number of epochs we try different values and set the final epochs to the value when the model achieves more than 99% accuracy on this training data. Values of other hyperparameters are set using the same technique. These same values of the hyperparameters are then used for training the model with the main dataset.

In the next few sections, we present an empirical evaluation to analyze the correctness and efficiency of SIFT.

Empirical evaluation

We carried out an empirical evaluation to assess the performance of SIFT. This section presents the dataset, evaluation metrics, empirical study, obtained results, and analysis. We also compare SIFT with three other state-of-the-art file fragment classification techniques. All experiments were run on an IntelÂ® Core(TM) i-7-4510U CPU @ 2.00 GHz with 8 GB of RAM, running Windows 8.1.

Dataset

To carry out different experiments we selected a publicly available dataset (Garfinkel 2024) for cyber forensics research. From this dataset, we randomly collected 20 file types and extracted 47,482 samples (fragments) from these files. The distribution of this dataset is shown in Table 1. To make sure the evaluation is unbiased, we selected the same number (seven) of files from each Class. We chose 512 bytes as the size of a fragment for our experiments. The reason for choosing this value is as follows. The researchers are divided between 512 (Axelsson 2010; Beebe et al. 2013, 2016; Calhoun and Coles 2008; Fitzgerald et al. 2012; Catanzaro et al. 2008; Sportiello and Zanero 2012) or 4096 (Karresand and Shahmehri 2006b, a; Li et al. 2011; Penrose et al.

Table 3 Weight assigned according to Eq. 2 to bytes in the fragments of some of the classes. To save space, we only show specific bytes whose weights are much higher than other bytes

Class	Byte value in hex	Symbol	Description	Weight assigned (averaged)
CSV	0×2C	,	Comma	0.663
CSV	0×22	"	Double quotes	0.335
DBASE3	0×20		Space	0.939
EPS	0×48	0	Zero	0.462
XML	0×3C	<	Open angled bracket	0.185
XML	0×3E	>	Close angled bracket	0.181
XLS	0×40	@		0.206
LOG	0×3A	:	Colon	0.160

2013; Sportiello and Zanero 2011; Veenman 2007) bytes as the size of a fragment. According to Penrose et al. (2013) all the hard drive manufacturers have used 4096 bytes as their sector size since 2011, therefore this is the right size to choose. However, Axelsson et al. (2010) makes an observation that 512 bytes are a conservative choice. Out of these two choices we chose the conservative value, i.e., 512 bytes.

Evaluation metrics

We drive our evaluation metrics from the confusion matrix. Confusion matrix (Fawcett 2006) is used for measuring the performance of a classification model. Items in a confusion matrix belong to an original Class from a trusted set of pre-classified items (ground truth values) and items that are classified by the model (predicted values). This allows us to compare our technique against the ground truth.

We use four metrics, True Positive Rate (**TPR**), False Positive Rate (**FPR**), **Precision**, and **F-Measure**, defined as follows:

$$TPR = \frac{TP}{P} \qquad and \qquad FPR = \frac{FP}{N}$$
$$Precision = \frac{TP}{TP + FP}$$
$$F - Measure = \frac{2TP}{2TP + FP + FN}$$

where, TP the true positive is the number of fragments classified as positive. FP the false positive is the number of fragments wrongly classified as positive. FN the false negative is the number of fragments wrongly classified as negative. P is the total number of fragments in the positive Class. N is the total number of fragments in the other Classes.

Empirical study and results

During the empirical study, we carried out three different experiments, firstly to obtain the threshold-based LIME and SHAP feature relevance values from the selected features, secondly for tuning the neural networks hyperparameters, and thirdly to conduct performance evaluation of SIFT.

Threshold based LIME and SHAP feature relevance values

The main dataset used in this paper consists of 47,482 samples as shown in Table 1. We first extracted the raw features from these samples and performed feature selection to obtain the samples with selected features listed in Eq. 3. We then trained a RandomForest classifier on these samples and performed 10-fold cross-validation. The results of this classification are shown in Table 2.

The RandomForest classifier was able to positively classify fragments of the 8 classes with a TPR > 95%. Weights were assigned using Eq. 2 to bytes in the fragments belonging to some of these and other classes. These weights are shown in Table 3. To save space, we only show specific bytes whose weights are much higher than other bytes. The table lists the average weight of all the fragments belonging to a class. As an example, the class CSV contains 889 fragments as shown in Table 1. Each of these 889 fragments contains 512 bytes and each of these bytes is assigned 889 different weights. Table 3 lists the average of these 889 weights for the class CSV and similarly for other classes. These bytes occur much more often in the fragments belonging to a specific class than any other class. This means these are the features (bytes) that helped successfully classify these fragments.

As shown in Table 2 we obtained a weighted average TPR of 79.8% and FPR of 1.8%. These results are better when compared to other state-of-the-art works discussed in Section "Comparison with other works". We still want to improve these results especially the FPR, and TPR of some of the file types, such as *html* and *swf*. The low results for these file types are because of the multinomial classification and to some extent class imbalance problems. The other major reason for these results is because of the problem of file embeddings as discussed before. To overcome some of these problems and provide a solution we apply XAI and compute threshold-based LIME and SHAP feature relevance values as follows.

For computing the LIME and SHAP feature relevance values we randomly chose a subset of samples from the main dataset. This subset contained 60 samples from each of the 20 classes, i.e., a total of 1200 ($60 \times 20 = 1200$) samples. We computed the LIME and SHAP feature relevance values as described in Section "XAI—feature relevance" and listed in Eqs. 4 and 5 using

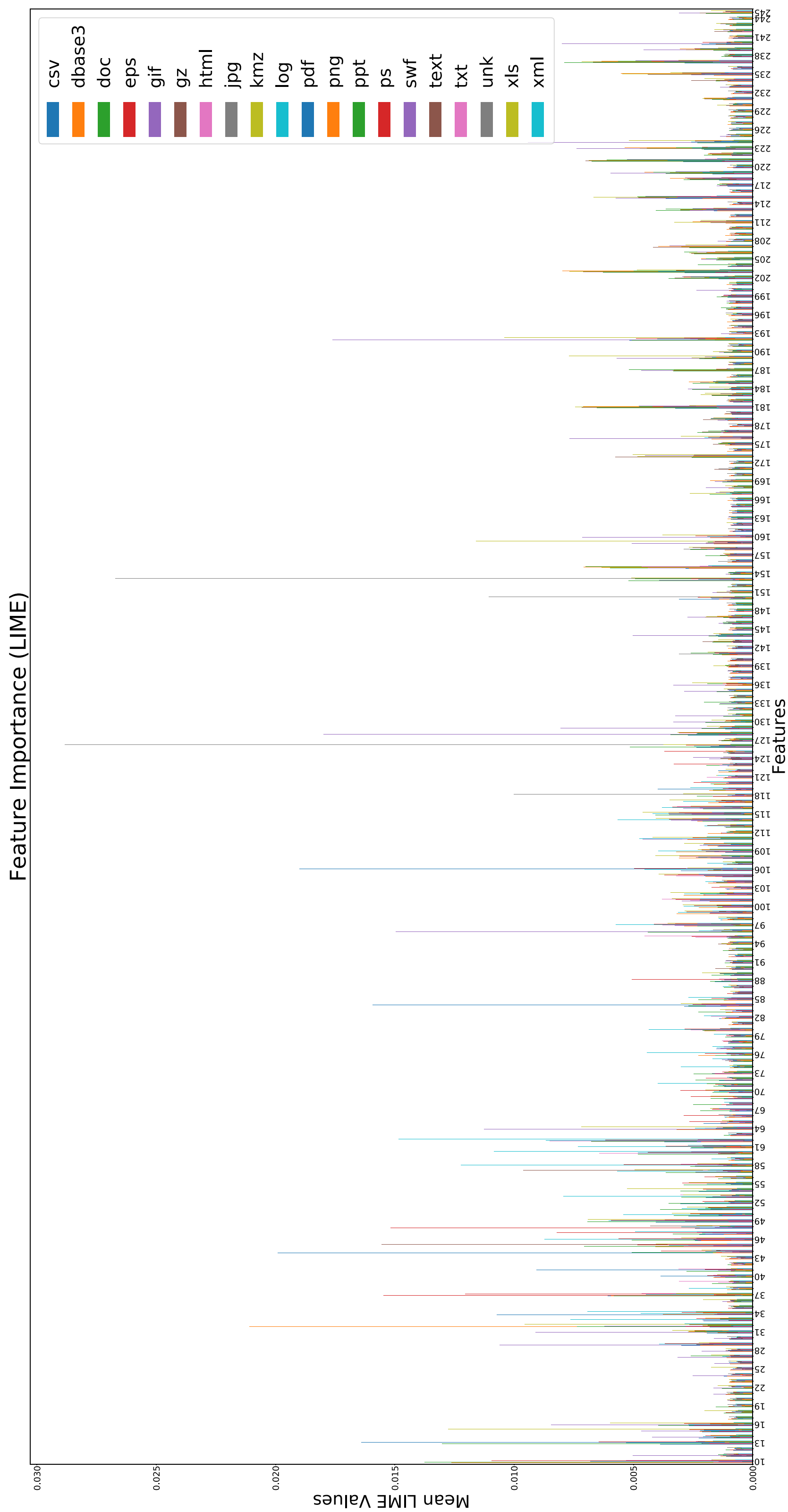

Fig. 6 LIME feature relevance values for the set of 1200 samples

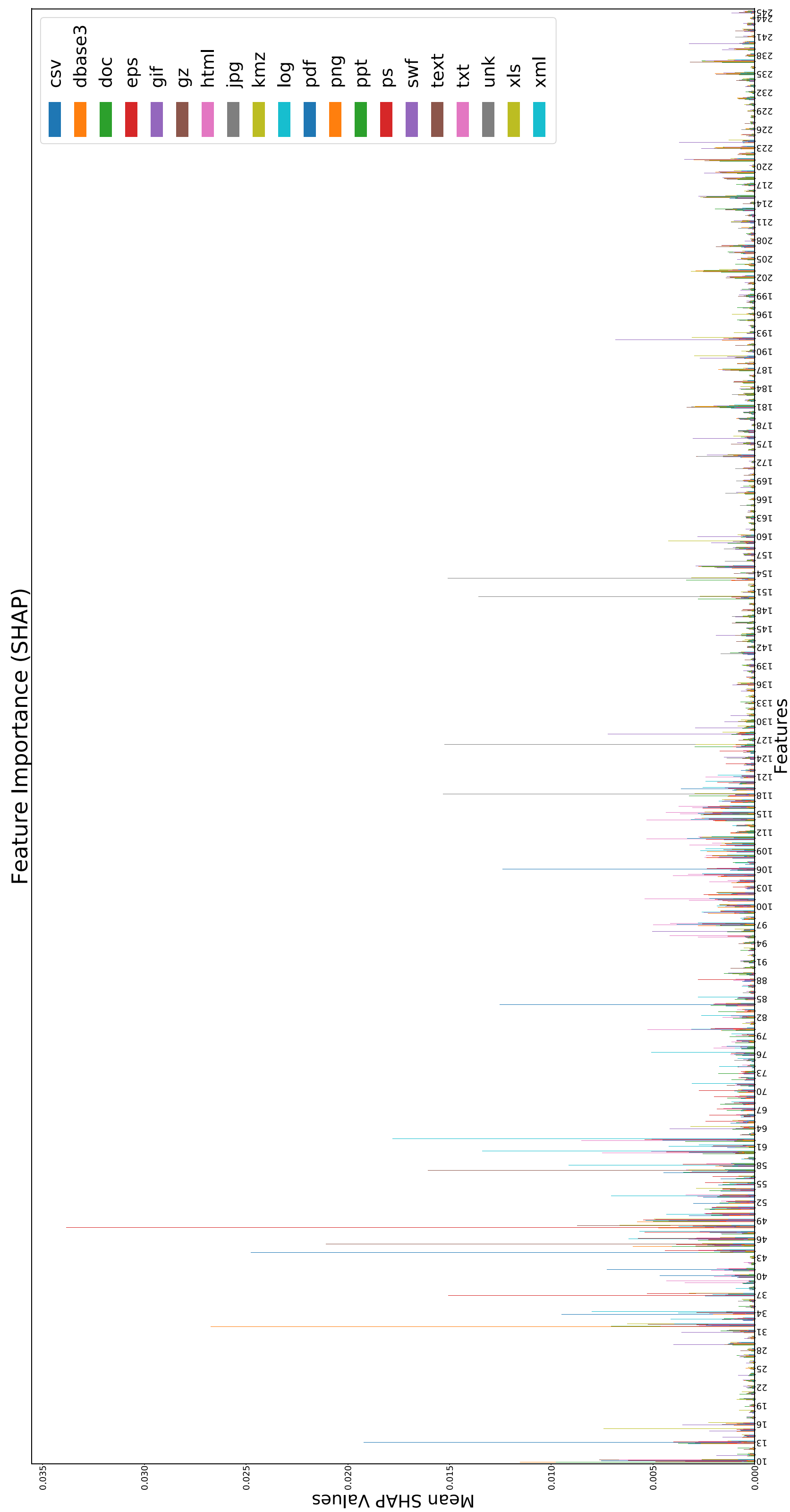

Fig. 7 SHAP feature relevance values for the set of 1200 samples

Table 4 Results of MLP model using the final dataset of 47,482 fragments (samples) with threshold-based LIME relevant features

File type (class)	TPR	FPR	Precision	F-Measure
csv	1.000	0.000	0.989	0.994
dbase3	1.000	0.000	1.000	1.000
doc	0.864	0.011	0.801	0.831
eps	0.985	0.000	0.992	0.988
gif	0.656	0.005	0.632	0.644
gz	0.610	0.020	0.764	0.678
html	0.989	0.000	0.989	0.989
jpg	0.450	0.012	0.341	0.388
kmz	0.938	0.000	0.985	0.961
log	0.998	0.000	0.998	0.998
pdf	0.592	0.020	0.631	0.611
png	0.694	0.032	0.526	0.599
ppt	0.639	0.034	0.601	0.619
ps	0.992	0.001	0.980	0.986
swf	0.415	0.028	0.395	0.405
text	0.833	0.011	0.889	0.860
txt	0.989	0.001	0.963	0.975
unk	0.982	0.001	0.987	0.985
xls	0.802	0.001	0.905	0.851
xml	0.981	0.000	0.981	0.981
Weighted Avg	0.821	0.009	0.818	0.817

Table 5 Results of MLP model using the final dataset of 47,482 fragments (samples) with threshold-based SHAP relevant features

File type (class)	TPR	FPR	Precision	F-Measure
csv	1.000	0.000	0.997	0.995
dbase3	1.000	0.000	0.875	0.933
doc	0.852	0.009	0.822	0.837
eps	0.988	0.001	0.990	0.989
gif	0.625	0.006	0.577	0.601
gz	0.707	0.032	0.702	0.705
html	1.000	0.000	0.989	0.994
jpg	0.351	0.005	0.495	0.412
kmz	0.923	0.000	0.966	0.944
log	1.000	0.000	0.997	0.999
pdf	0.613	0.024	0.595	0.604
png	0.679	0.023	0.597	0.635
ppt	0.593	0.018	0.725	0.652
ps	0.989	0.000	0.991	0.990
swf	0.470	0.029	0.416	0.442
text	0.861	0.018	0.883	0.847
txt	0.985	0.001	0.955	0.970
unk	0.978	0.000	0.991	0.984
xls	0.820	0.001	0.926	0.870
xml	0.968	0.000	0.987	0.977
Weighted Avg	0.820	0.008	0.821	0.820

these 1200 samples. Figures 6 and 7 show these LIME and SHAP values for each of the 20 classes in the dataset.

For the 20 classes LIME and SHAP feature relevance values are in the range of 0–0.034 as shown in feature relevance graphs in Figs. 6 and 7. There are 120 out of 236, i.e., 50.85%, features that have a value < 0.0025 for each of the 20 classes in both LIME and SHAP feature relevance graphs. There are 116 out of 236, i.e., 49.15%, features that have a value > 0.0025 for some of the 20 classes. That means both these techniques need a different threshold value to mark irrelevant features as described in Section "XAI—threshold-based feature relevance". We perform different experiments using the 1200 samples chosen above as described in section "XAI—threshold-based feature relevance" and computed the two threshold values as $T_{lime} = 0.00075$ and $T_{shap} = 0.00015$. Using these two threshold values we obtained the fragments (samples) with threshold-based relevant features for LIME and SHAP as listed in Eqs. 6 and 7. These experiments were conducted using the MLP model as described in "Multinomial classification with deep learning".

Neural networks hyperparameters

During the experiments carried out above we also fine-tuned the different MLP hyperparameters for the dataset used in this paper. The *batch size* and *epochs* were set to 189 and 200 respectively and 5 *hidden layers* were created to train the MLP model.

Performance evaluation of SIFT

The final dataset of 47,482 fragments (samples) with threshold-based relevant features was divided into 80% training and 20% testing data as input to our MLP model as described in section "Multinomial classification with deep learning" for multinomial classification of these file fragments. The time of training the MLP model with threshold-based LIME relevant features was 612.62 s (0.016 s per sample—total training samples 37,985) and with threshold-based SHAP relevant features was 493.44 s (0.013 s per sample). The training time per sample indicates that SIFT is scalable and can efficiently handle a much larger dataset. To validate this in future we will evaluate SIFT with a much larger dataset. The obtained results are shown in Tables 4 and 5.

Analysis

The confusion matrices of the MLP model using the final dataset of 47,482 fragments (samples) with threshold-based LIME and SHAP relevant features are shown in Figs. 8 and 9 respectively.

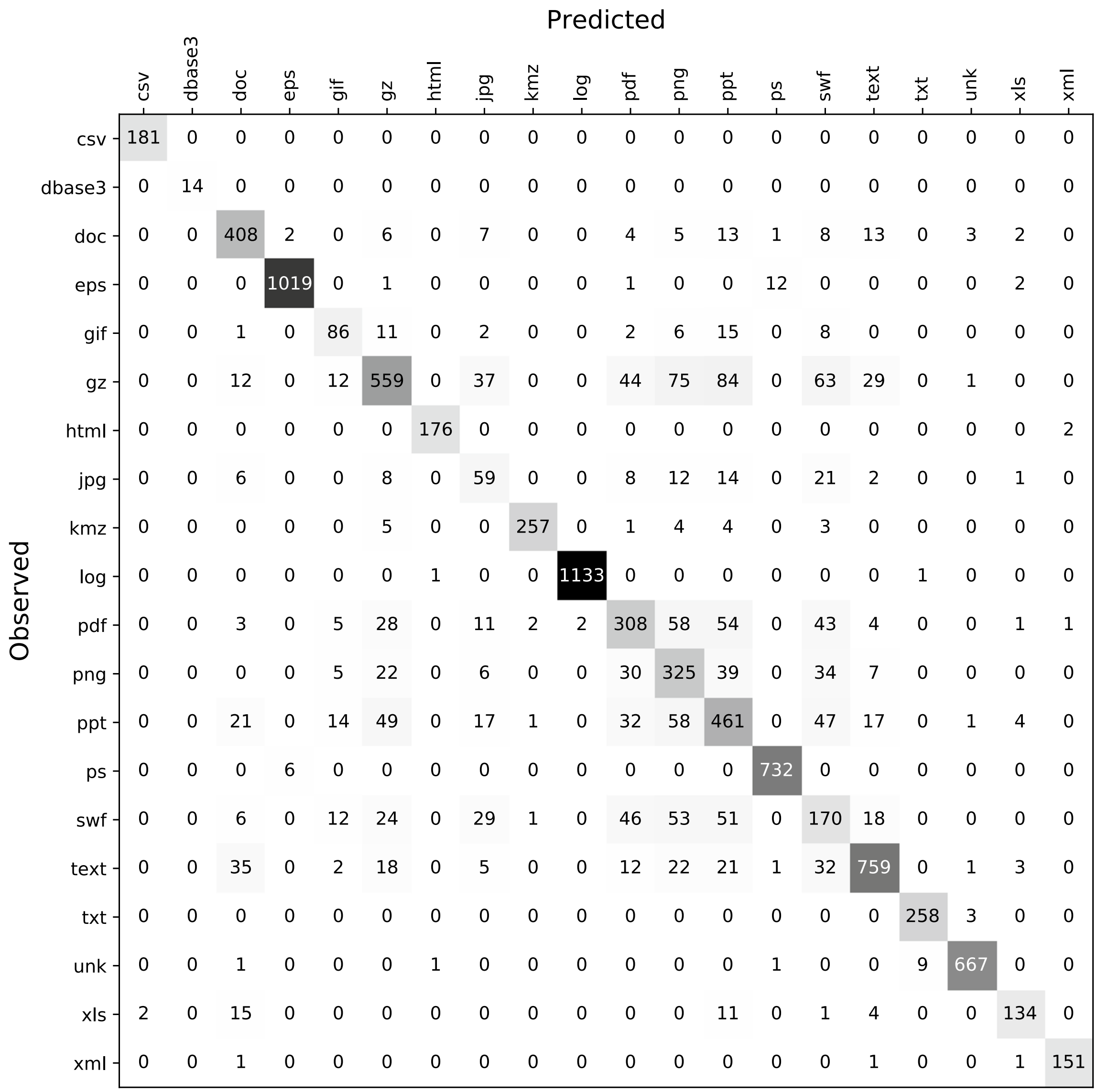

Fig. 8 Confusion matrix of MLP model using the final dataset of 47,482 fragments (samples) with threshold-based LIME relevant features

The testing data contained a total of 9497 samples (20% of 47,482). Here we define and compute another metric from the confusion matrix as follows:

$$Accuracy = \frac{\text{Total correctly predicted samples}}{\text{Total number of samples}}$$

From the confusion matrices (Figs. 8 and 9) we compute the SIFT overall accuracy with LIME and SHAP values as follows:

$$\text{Accuracy with LIME} = \frac{7857}{9497} \times 100 = 82.37\%$$

and

$$\text{Accuracy with SHAP} = \frac{7941}{9497} \times 100 = 83.62\%$$

SHAP provided a slightly better accuracy than LIME. As we can see from Tables 4 and 5 the TPR of LIME (82.1%) is slightly better than the TPR of SHAP (82%) but the FPR of SHAP (0.8%) is slightly better than LIME (0.9%). FPR has a little edge over TPR when computing the overall accuracy of a model. The accuracy of SIFT without any LIME or SHAP values (RandomForest classification – Table 2 is 79.77%. This accuracy is computed from the confusion matrix of the RandomForest classifier not shown in the paper. When comparing the results of SIFT without XAI and with XAI and DL, applying XAI

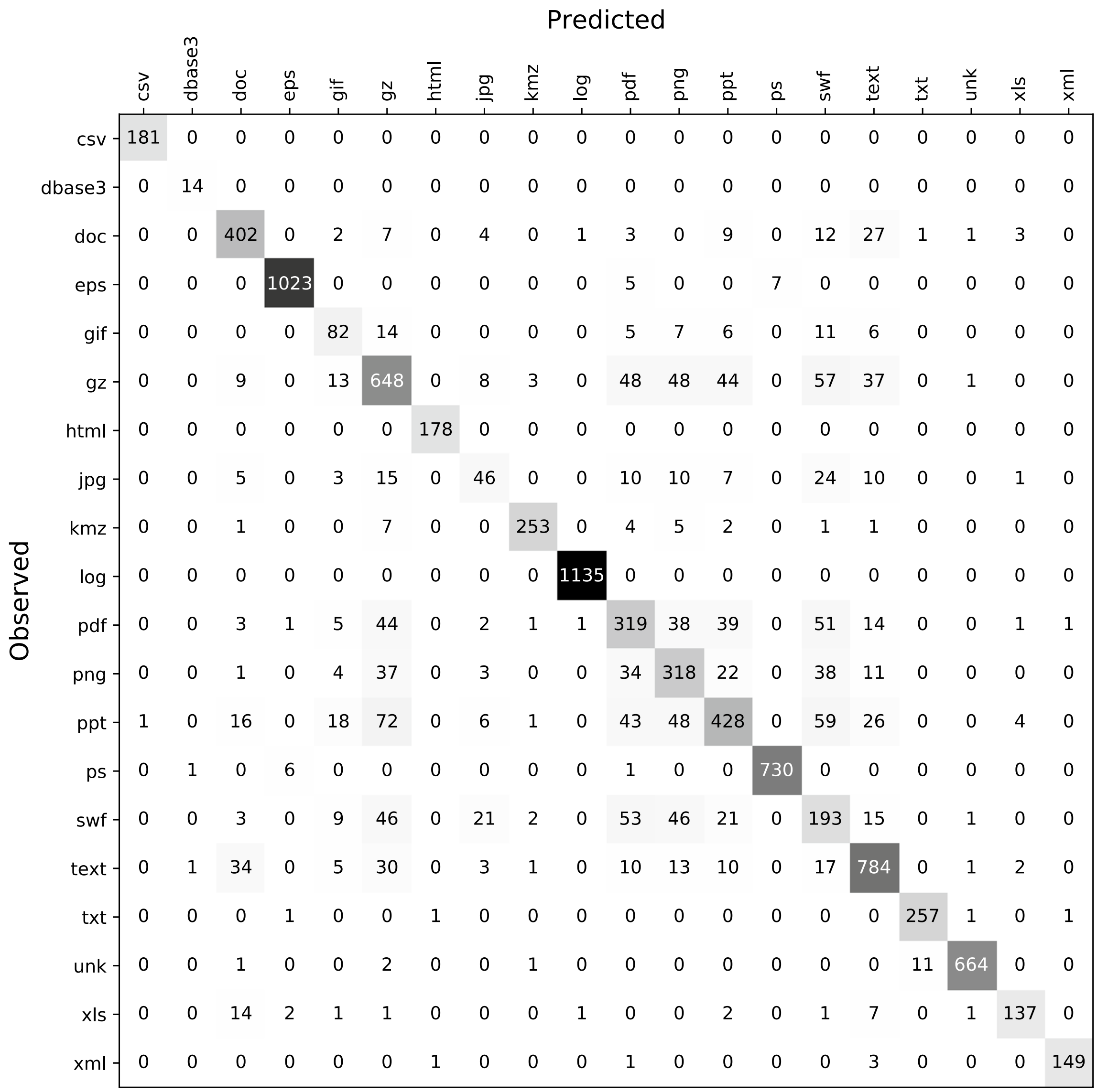

Fig. 9 Confusion matrix of MLP model using the final dataset of 47,482 fragments (samples) with threshold-based SHAP relevant features

Table 6 Distribution of the TPR results of SIFT at the local (class) level using different classifiers

TPR (%)	RandomForest	MLP with LIME values	MLP with SHAP values
98–100	CSV, DBASE3, EPS, LOG, PS, UNK	CSV, DBASE3, EPS, HTML, LOG, PS, TXT, UNK, XML	CSV, DBASE3, EPS, HTML, LOG, PS, TXT
91–97	HTML, KMZ, TXT, XML	KMZ	KMZ, UNK, XML
71–90	DOC, GZ, PNG, TEXT, XLS	DOC, TEXT, XLS	DOC, TEXT, XLS
40–70	PDF, PPT	GIF, GZ, JPG, PDF, PNG, PPT, SWF	GIF, GZ, PDF, PNG, PPT, SWF
0–39	GIF, JPG, SWF	–	JPG

with DL improved the overall accuracy of SIFT by 3.85%, the TPR by 2.3%, and the FPR by 1%. These may seem small improvements but they reinforce the claim that applying XAI, SIFT can solve to a certain extent some of the challenges of file fragment classification. These are overall performance improvements. We did not apply

Table 7 The updates (increase/decrease → ±) in accuracy at local (class) level after applying XAI (LIME and SHAP values)

Class	RandomForest accuracy (%)	MLP with LIME values accuracy	MLP with SHAP values accuracy
GIF	35.66	66.15% (+ 30.49%)	62.60% (+ 26.94%)
GZ	88.98	61.03% (− 27.95%)	71.44% (− 17.54%)
JPG	00.00	50.43% (+ 50.43%)	35.11% (+ 35.11%)
SWF	12.36	41.46% (+ 29.10%)	47.07% (+ 34.71%)

global interpretations provided by LIME and SHAP. In the future, we would like to apply global interpretations provided by some of the XAI techniques to improve the overall results and especially further improve the multinomial classification issue.

Distribution of the TPR results of SIFT at the local (class) level using RandomForest (i.e., without XAI) and MLP with LIME and SHAP values (i.e., with XAI) are shown in Table 6. We can see that improvements in classification are significant when XAI is applied with DL. More classes are predicted in the range of 98–100%, and no classes < 40% except one class JPG with SHAP, with LIME and SHAP values. There are more classes predicted in the range 40–70% that depict that applying XAI and DL SIFT can solve some of the challenges, especially the multinomial classification and file embedding problems up to a certain extent.

The updates (increase/decrease) in accuracy at the class level after applying XAI and are shown in Table 7. Here also, LIME and SHAP are the two XAI techniques and provide significant improvements in accuracy (25–50%) at the local class level. There is only one class GZ where the accuracy dropped. 29.1% in LIME (Fig. 8) and 21.8% in SHAP (Fig. 9) of the fragments of GZ are predicted as type PDF (4.8% & 5.3%), PNG (8.2% & 5.3%), PPT (9.2% & 4.9%), and SWF (6.9% & 6.3%). GZ (gzip) type of file consists of compressed data. In general, an image (i.e., file types PNG and SWF, etc) is first compressed and then stored in the file. The file types PDF and PPT (Microsoft PowerPoint) may also contain images. The compressed fragments found in these file types (classes) are the main reason why SIFT with LIME and SHAP predicted them as class GZ. This is just an observation and may need further research. In the future, we will look into the respective LIME and SHAP values and specific interpretations of the (MLP) model to know the reasons why there is a decline and then improve such a prediction.

The two XAI techniques LIME and SHAP are model agnostic, i.e., they can be used with any model. That means LIME and SHAP feature relevance values computed are generated just once and can be used with any model (DL or any other ML model). To validate and test this claim we carried out another experiment and used the SHAP feature relevance values with Random Forest to classify the 20 file types. The accuracy achieved was 79.9%. Random Forest without SHAP achieved an accuracy of 79.7%. Random Forest with XAI improved by 0.2% over Random Forest without XAI. This indicates that XAI is able to improve (although small but still an improvement) the performance of not only a DL (MLP) but also a ML model, and this also verifies that LIME and SHAP are model agnostic.

Moreover, we only used LIME and SHAP local interpretations, and using these values we are able to improve the performance relative to other works (compared in Section "Comparison with other works") that use other techniques for feature selection/importance. Using the same values we are also able to further improve the results presented in Table 2. Improvements at the local (class) level are more significant as shown in Tables 6 and 7. This indicates and corroborates that XAI is capable of enhancing the performance of an AI model.

Comparison with other works

To compare our technique we chose three state-of-the-art techniques in this area. The first (Haque and Tozal 2022) and the most recent from the year 2022, the second (Bhatt et al. 2020) and last (Wang et al. 2018) from the

Table 8 Comparison of the model SIFT proposed in this paper and three other state-of-the-art models

Model	TPR (%)	Number of fragments	Size of each fragment (byted)	Number of classes	Techniques used
SIFT	82.1	47,482	512	20	Adapted TF-IDF for assigning weights; XAI for feature relevance; MLP for classification
Haque and Tozal (2022)	72	87,500	4096	35	Byte2Vec embeddings; extension of Word2Vec and Doc2Vec; k-nearest neighbors for classification
Bhatt et al. (2020)	67	14,000	512	14	Ten features, such as entropy, bigram distribution, hamming weight, mean byte value, etc; support vector machine for classification
Wang et al. (2018)	61	270,000	512	18	Continuous sequence (n–gram) of bytes of different sizes; support vector machine for classification

years 2020 and 2018, respectively. The reasons for selecting them are: (1) They perform file fragment classification and are published in IEEE, Elsevier, and MDPI journals. (2) They select their samples from the same dataset (Garfinkel 2024) as used in this paper. (3) They also employ machine learning to improve performance and perform automated classification.

Table 8 provides a comparison of SIFT with the other three techniques. SIFT outperforms the others by 10–19%. (1) One of the major differences between SIFT and others is that SIFT uses a single byte as a separate feature, i.e., a total of 256 (0×00–0×FF) features. We also call this a lossless feature (information) extraction, i.e., there is no loss of information. (2) The other major difference is the technique used to estimate inter-classes and intra-classes information gain of a feature. For this purpose, SIFT adapts TF-IDF to compute and assign weight to each byte (feature) in a fragment (sample) and then applies two XAI techniques LIME and SHAP to compute the input feature relevance for selecting important and relevant features. (3) For classification SIFT uses a model MLP that trains on the dataset with relevant features. With these major differences and approaches, SIFT produces promising (better) results.

As discussed in Section "Fragment classification challenges", a large number of classes in the dataset increases the complexity of multinomial classification and this effects the TPR of the classifier. All the works compared use more than 13 classes for training and testing, which depicts that this is a real challenge in file fragment classification, and also presents a fair comparison with SIFT that uses 20 classes. To mitigate this challenge we use TF-IDF and two XAI techniques LIME and SHAP, and also develop a deep learning model MLP by fine tuning its hyperparameters to optimize multinomial classification. We also conducted experiments using different number of classes. When tested with 18 and 14 classes SIFT achieved a TPR of 83% and 86.9% respectively. This shows SIFT outweighs the other two works that use a similar number of classes.

Size of each fragment used in all the works compared are the same, 512 bytes, as SIFT except (Haque and Tozal 2022) uses 4096 bytes. As discussed in Section "Dataset", researchers are divided on the size of fragment to use. We choose the conservative size 512 bytes because of the legacy storage systems still being used. Also for comparison we choose works that use the same size. But we also want to include one of the works that use the fragment size 4096 bytes being used in modern storage systems. As we can see using the conservative size SIFT achieves better results.

Related work

Binary file fragments have been explored in various scientific contexts, including digital forensic analysis, reverse engineering, and fuzzing, among many others. In this section, we briefly highlight recent research works on file fragment classification in the context of file carving in digital forensics. There are multi-fold ways that one could organize a taxonomy of file fragment classification. We divide these works into three popular categories (Sester et al. 2021). There are several works that have used XAI to explain the predictions of an AI model in CF but none of them have used such explanation to optimize the predictions/classification. At the end of this section we also present a short discussion on such works.

Signature based approaches

Known signatures in file headers and footers are exclusively useful in file carving. Nevertheless, this approach assumes that file clusters remain consecutively. In case of file fragmentation, file clusters can be separated, and the order can be disrupted such that distinctly file carving will fail. Signature-based techniques use the potential embedded signatures (Sester et al. 2021). Similarly, Roussev et al. (2012) suggested the adoption of *sdhash* real-time digital forensics and triage. Breitinger et al. (2013) used typically similarity-preserving hashing (SPH). Consequently, Lillis et al. (2017) boosted the lookup speed by way of hierarchical Bloom filter tress.

Earlier, Garfinkel et al. (2006) and Dandass et al. (2008) urged the use of hash-values for fragments to identify individual files with the same fragments. A few modifications on MD5 and SHA1 to CRC32 hashing algorithms were also blessed to measure hash values. Besides, Garfinkel et al. (2010) investigated a faster design to match master files and image files together by using maps.

Statistical approaches

Conti et al. (2010b; a) pick statistical features, like Shannon entropy, chi-square, hamming weight, and arithmetic mean to resolve the low-level binary data. In each group, 1000 fragments, where fragment size is 1 KB, are analyzed. Statistical features are detected in agreement with the distribution of data fragments by primitive fragment class. It occurred that the bitmap samples exhibit little clustering, but the high entropy, text, encoded, and machine code primitive types are more densely clustered.

Calhoun et al. (2008) use statistical features, like entropy and frequency of ASCII codes to sift graphic files, JPG, GIF, etc. Promising results (83% accuracy) are obtained. However, the results are only applicable to graphic types. Veenman et al. (2007) use statistical features, such as histogram, and entropy to classify disc images. A dataset of 450 MB is collected from the Internet and used. They carried out multi-class and two-class perception experiments with 0.45 overall accuracy which is quite modest. The results indicate that ZIP files were classified with only 18% accuracy while HTML and JPEG files came out with 98% accuracy.

Karresand et al. (2006b) introduced Oscar which computes the divergence of the ASCII values in the seam of two successive bytes as a scale of change to classify file types. As far as is know, Oscar only outperforms well on JPG file types. Representing the mean and standard deviation of the byte frequency distribution of distinct file types, called Centroids, is the fundamental base for the Oscar approach. A weighted quadratic distance metric is assigned with the distance between the centroid and sample data fragments. When the distance falls below a threshold, the sample is classified as possibly associated with the modeled file type. Besides, Li et al. (2005) extract a 1-gram binary distribution for file fragment classification on files gathered from the Internet utilizing a general search of a file type on Google. Results are promising when they are realized by using a one-centroid and multi-centroid file-type model. Ultimately, McDaniel et al. (2003) proposed a file fingerprint for file type detection. They extract the byte frequency analysis, byte frequency correlation, and file header/trailer information to produce the file fingerprint.

Artificial intelligence (AI) based approaches

AI-based file fragment classification approaches have been emerging recently. In Ghaleb et al. (2023), Ghaleb et al. use convolutional neural networks (CNN) with an accuracy of 79%. Lie et al. (2023) also use CNN for file fragment classification using bit shift and n-Gram embeddings. Recurrent and convolutional neural networks (RCNN) have established that ByteCRNN resulted with 71.1% average accuracy on 512-byte fragments and 83.9% average accuracy on 4096-byte fragments (Skračić et al. 2023). Zhu et al. (2023) also used CNN along with LSTM that achieved an average accuracy of 66.5% and 78.6% for 512-byte and 4096-byte file fragments, respectively.

Haque et al. (2022) introduced a model that broadens Word2Vec and Doc2Vec embeddings to bytes and fragments. The Byte2Vec name is given to this model. 4096 bytes fragment sizes are separated from each file. Byte2Vec embeddings are used to vectorize these fragments. The k-nearest neighbor classification is applied afterward. Byte2Vev models achieved an accuracy of 72% and a TPR of 72% during the tests.

Bhatt et al. (2020) introduced a hierarchical machine-learning-based model for file fragment classification. SVM is used as a base classifier. Entropy and bigram distribution, hamming weight, mean byte value, etc., a total of ten features, are extracted from each fragment. The proposed approach with the SVM model achieved an accuracy of 67.78% and a TPR of 67%.

Chen et al. (2018) introduced an approach. At first, a fragment is turned into a grayscale image for extracting high-dimensional features. Afterward, a convolution neural network model is used for the classification of fragments. Experiments on models showed an accuracy of 70.9%.

Wang et al. (2018) leveraged sparse coding as automatic feature extraction. Features corresponding to how well these can be used to reconstruct the original data are extracted by sparse coding. Based on this principle, a continuous sequence of bytes (n-grams) of distinct sizes is used, and the method showed an accuracy of 61.31% and a TPR of 60.99%.

Only special types of fragments are classified by the majority of the previous studies, such as graphic types (JPG, GIF, PNG, etc.). However, a few of the applied approaches do not perform well for high entropy fragments, as they do not have apparent patterns to attain. The approach proposed in this study does not have such constraints because of lossless feature extraction and application of XAI techniques, LIME and SHAP, for selecting important and relevant features that make it feasible to successfully classify different fragment types but a few.

XAI in CF

A detailed review on research works that have used XAI to explain the predictions of an AI model in CF is presented in Alam and Altiparmak (2024). Here we present and discuss few of these recent works.

Afzaliseresht et al. (2019) present an XAI model for analyzing security event logs, which can be used during forensics investigation of security events. After mining temporal patterns to discover sequential events from a log file, storeytelling is used to present this sequence of events in a human readable format. This reduces the efforts of humans to interpret events.

Mahajan et al. (2021) use LIME to interpret and evaluate AI models for toxic comment classification. The experiments and results concluded that XAI techniques such as LIME are important in selecting the best model.

Jayakumar et al. (2022) present a method to enhance the interpretability of deepfake detection models. The method visually explains why a deppfake detection model classifies a video as a deepfake. This plays a crucial role in the decision-making process of juries in CF investigations.

Hall et al. (2022) evaluate and interpret different AI models using LIME. These models were trained to classify file types. After classification the results were input to LIME for explanation. Most of the time LIME was able to explain the classification results but sometime failed because of the feature interaction. Here LIME is used to explain the classification results, whereas we have used LIME and SHAP to optimize classification.

Bouter et al. (2023) propose a method for visualizing and interpreting predictions of deepfake video data for forensic analysis. This method allows a forensic analyst to intuitively interact with the model and hence helps the analyst thoroughly explain and evaluate the model. This aids the analyst in making a decision if the video is manipulated or not. The explanation about this decision can be presented in a court of law as a piece of trustworthy evidence.

Limitations and future work

Computer files are often embedded with other files, such as images, PNG and JPG, etc., embedded in PDF and PPT file types. Some fragments (image type) of these files will be classified as the other Class (image). In this case, sometimes SIFT is not able to correctly identify these fragment types. In the future, we will look into the respective LIME and SHAP values and specific interpretations of the (MLP) model to know the reasons why there is a decline and then improve such predictions.

The number distributions of different types of files are not the same in GovDocs, for example, when the number of files is very small, it will affect the accuracy of the final classification results. This paper does not optimize the dataset itself. Therefore, the bias may affect the accuracy of our model. In future work, we can focus on the optimization of datasets and models to further improve classification accuracy.

When applying the XAI techniques, LIME and SHAP, we only used their local interpretations, i.e., local feature relevance values. In the future, we would like to apply global interpretations provided by some of the XAI techniques to improve the overall results and especially improve the multinomial classification issue.

To explore the real world impact of our research, as a future work we will implement the proposed technique in this paper as part of a tool that provides a complete file recovery. To test the scalability of SIFT in future we will evaluate SIFT with a much larger dataset.

Conclusion

File carving is the practice of repairing damaged files on a storage media in part or whole without any filesystem information. An essential issue in file carving is the recognition of file fragment types. In this paper, we propose a novel file fragment type identification method based on the TF-IDF technique to assign a weight for each byte (feature) to select important features in a fragment. We used 512-byte segments. Then, we investigated three multinomial classifiers, namely Naive Bayes, Decision Tree, and Random Forest, to evaluate the performance on a popular and publicly available dataset by 10-fold cross-validation in terms of TPR, FPR, Precision, F-measure, and AUC metrics. Among these classifiers, Random Forest performs the best with our novel feature selection technique.

In this paper, we presented a novel sifting file types method, called SIFT. SIFT analyzed a total of 47,482 low-level binary file fragments belonging to 20 file types (classes). Our experimental results show that SIFT reaches a TPR of 82.1%. Compared to other state-of-the-art methods presented in Haque and Tozal (2022), Bhatt et al. (2020) and Wang et al. (2018) where they select their samples from the same dataset (Garfinkel 2024) as used in this paper, SIFT outperforms by 10–19%.

Acknowledgements
Not applicable.

Author contributions
Shahid Alam: idea development; system design, implementation, and testing; writing and editing of the paper. Alper Kamil Demir: background and literature review; writing and editing of the paper.

Funding
None.

Availability of data and materials
The dataset used in the paper is publicly available in Garfinkel (2024).

Declarations

Competing interests
The authors declare no potential competing interests.

Received: 2 January 2024 Accepted: 10 April 2024
Published online: 11 September 2024

References
Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6:52138–52160
AfzaliSeresht N, Liu Q, Miao Y (2019) An explainable intelligence model for security event analysis. In: AI 2019: Advances in Artificial Intelligence: 32nd Australasian Joint Conference, Adelaide, SA, Australia, December 2–5, 2019, Proceedings 32, Springer, pp 315–327

Alam S (2022) Cyber Security: Past Present and Future. Lambert Academic Publishing, London, UK
Alam S (2023) Sift—file fragment classification without metadata. In: 3rd International Conference on Computing and Information Technology (ICCIT), IEEE, pp 123–129
Alam S, Altiparmak Z (2024) XAI-CF–Examining the role of explainable artificial intelligence in cyber forensics. arXiv preprint arXiv:2402:02452
Ali S, Abuhmed T, El-Sappagh S et al (2023) Explainable artificial intelligence (xai): what we know and what is left to attain trustworthy artificial intelligence. Inform Fus 99:101805
Axelsson S (2010) The normalised compression distance as a file fragment classifier. Digit Investig 7:S24–S31
Beebe N, Liu L, Sun M (2016) Data type classification: Hierarchical class-to-type modeling. In: IFIP International Conference on Digital Forensics, Springer, pp 325–343
Beebe NL, Maddox LA, Liu L et al (2013) Sceadan: using concatenated n-gram vectors for improved file and data type classification. IEEE Trans Inf Forensics Secur 8(9):1519–1530
Bhatt M, Mishra A, Kabir MWU et al (2020) Hierarchy-based file fragment classification. Mach Learn Knowled Extract 2(3):216–232
Boiko M, Moskalenko V, Shovkoplias O (2023) Advanced file carving: ontology, models and methods. Radioelectron Comput Syst 1(3):204–216
Bouter MdLd, Pardo JL, Geradts Z, et al (2023) Protoexplorer: Interpretable forensic analysis of deepfake videos using prototype exploration and refinement. arXiv preprint arXiv:2309:11155
Breitinger F, Stivaktakis G, Baier H (2013) Frash: a framework to test algorithms of similarity hashing. Digit Investig 10:S50–S58
Calhoun WC, Coles D (2008) Predicting the types of file fragments. Digit Investig 5:S14–S20
Capuano N, Fenza G, Loia V et al (2022) Explainable artificial intelligence in cybersecurity: a survey. IEEE Access 10:93575–93600
Catanzaro B, Sundaram N, Keutzer K (2008) Fast support vector machine training and classification on graphics processors. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 104–111
Chen Q, Liao Q, Jiang ZL, et al (2018) File fragment classification using grayscale image conversion and deep learning in digital forensics. In: 2018 IEEE Security and Privacy Workshops (SPW), IEEE, pp 140–147
Chisum WJ, Turvey B (2000) Evidence dynamics: Locard's exchange principle & crime reconstruction. J Behav Profil 1(1):1–15
Conti G, Bratus S, Shubina A et al (2010) A visual study of primitive binary fragment types. White Paper, Black Hat USA
Conti G, Bratus S, Shubina A et al (2010) Automated mapping of large binary objects using primitive fragment type classification. Digit Investig 7:S3–S12
Dandass YS, Necaise NJ, Thomas SR (2008) An empirical analysis of disk sector hashes for data carving. J Digit Forensic Pract 2(2):95–104
Dhanalakshmi R, Chellappan C (2009) File format identification and information extraction. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, pp 1497–1501
Dunsin D, Ghanem MC, Ouazzane K, et al (2023) A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response. arXiv preprint arXiv:2309:07064
Fawcett T (2006) An Introduction to ROC Analysis. Pattern Recogn Lett 27(8):861–874
Fitzgerald S, Mathews G, Morris C et al (2012) Using nlp techniques for file fragment classification. Digit Investig 9:S44–S49
Garfinkel S (2024) GovDocs. Digital Corpora, http://downloads.digitalcorpora.org/corpora/files/govdocs1
Garfinkel S, Nelson A, White D et al (2010) Using purpose-built functions and block hashes to enable small block and sub-file forensics. Digit Investig 7:S13–S23
Garfinkel SL (2006) Forensic feature extraction and cross-drive analysis. Digit Investig 3:71–81
Garfinkel SL, McCarrin M (2015) Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. Digit Investig 14:S95–S105
Ghaleb M, Saaim K, Felemban M, et al (2023) File fragment classification using light-weight convolutional neural networks. arXiv preprint arXiv:2305:00656
Górriz J, Álvarez-Illán I, Álvarez-Marquina A et al (2023) Computational approaches to explainable artificial intelligence: advances in theory, applications and trends. Inform Fus 100:101945
Hall SW, Sakzad A, Minagar S (2022) A proof of concept implementation of explainable artificial intelligence (xai) in digital forensics. In: International Conference on Network and System Security, Springer, pp 66–85
Haque ME, Tozal ME (2022) Byte embeddings for file fragment classification. Futur Gener Comput Syst 127:448–461
Hassija V, Chamola V, Mahapatra A, et al (2023) Interpreting black-box models: a review on explainable artificial intelligence. Cogn Comput pp 1–30
Heidari E, Sobati MA, Movahedirad S (2016) Accurate prediction of nanofluid viscosity using a multilayer perceptron artificial neural network (mlp-ann). Chemom Intell Lab Syst 155:73–85
Hinton G, Srivastava N, Swersky K (2012) Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on 14(8):2
Hoffman RR, Mueller ST, Klein G et al (2023) Measures for explainable ai: explanation goodness, user satisfaction, mental models, curiosity, trust, and human-ai performance. Front Comput Sci 5:1096257
Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
Islam MR, Ahmed MU, Barua S et al (2022) A systematic review of explainable artificial intelligence in terms of different application domains and tasks. Appl Sci 12(3):1353
Jayakumar K, Skandhakumar N (2022) A visually interpretable forensic deepfake detection tool using anchors. In: 2022 7th international conference on information technology research (ICITR), IEEE, pp 1–6
Kaplan AD, Kessler TT, Brill JC et al (2023) Trust in artificial intelligence: meta-analytic findings. Hum Factors 65(2):337–359
Karresand M, Shahmehri N (2006a) File type identification of data fragments by their binary structure. In: Proceedings of the IEEE Information Assurance Workshop, IEEE, pp 140–147
Karresand M, Shahmehri N (2006b) Oscar–file type identification of binary data in disk clusters and ram pages. In: IFIP International Information Security Conference, Springer, pp 413–424
Kaur D, Uslu S, Rittichier KJ et al (2022) Trustworthy artificial intelligence: a review. ACM Comput Surv 55(2):1–38
Langer M, König CJ, Back C et al (2023) Trust in artificial intelligence: comparing trust processes between human and automated trustees in light of unfair bias. J Bus Psychol 38(3):493–508
Leichtmann B, Humer C, Hinterreiter A et al (2023) Effects of explainable artificial intelligence on trust and human behavior in a high-risk decision task. Comput Hum Behav 139:107539
Li Q, Ong A, Suganthan P, et al (2011) A novel support vector machine approach to high entropy data fragment classification. In: Proceedings of the South African Information Security Multi-Conf (SAISMC), University of Plymouth, pp 236–247
Li WJ, Wang K, Stolfo SJ, et al (2005) Fileprints: Identifying file types by n-gram analysis. In: Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, IEEE, pp 64–71
Lillis D, Breitinger F, Scanlon M (2017) Expediting mrsh-v2 approximate matching with hierarchical bloom filter trees. In: International Conference on Digital Forensics and Cyber Crime, Springer, pp 144–157
Liu W, Wang Y, Wu K, et al (2023) A byte sequence is worth an image: Cnn for file fragment classification using bit shift and n-gram embeddings. arXiv preprint arXiv:2304:06983
Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. Adv Neural Inform Process Syst. 30
Mahajan A, Shah D, Jafar G (2021) Explainable ai approach towards toxic comment classification. In: Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 2, Springer, pp 849–858
Manning C, Raghavan P, Schütze H (2010) Introduction to information retrieval. Nat Lang Eng 16(1):100–103
McDaniel M, Heydari MH (2003) Content based file type detection algorithms. In: 36th Annual Hawaii international conference on system sciences, 2003. Proceedings of the, IEEE, pp 10–pp
Mittal G, Korus P, Memon N (2020) Fifty: large-scale file fragment type identification using convolutional neural networks. IEEE Trans Inf Forensics Secur 16:28–41
Păvăloaia VD, Necula SC (2023) Artificial intelligence as a disruptive technology–a systematic literature review. Electronics 12(5):1102

Penrose P, Macfarlane R, Buchanan WJ (2013) Approaches to the classification of high entropy file fragments. Digit Investig 10(4):372–384
Ribeiro MT, Singh S, Guestrin C (2016) "Why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 1135–1144
Roussev V, Quates C (2012) Content triage with similarity digests: The m57 case study. Digit Investig 9:S60–S68
Saeed W, Omlin C (2023) Explainable ai (xai): a systematic meta-survey of current challenges and future opportunities. Knowl-Based Syst 263:110273
Saxena I, Usha G, Vinoth N, et al (2023) The future of artificial intelligence in digital forensics: A revolutionary approach. In: Artificial Intelligence and Blockchain in Digital Forensics. River Publishers, pp 133–151
Schwalbe G, Finzel B (2023) A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. Data Mining and Knowledge Discovery pp 1–59
Sester J, Hayes D, Scanlon M et al (2021) A comparative study of support vector machine and neural networks for file type identification using n-gram analysis. Forensic Sci Int Digital Investig 36:301121
Skračić K, Petrović J, Pale P (2023) Bytercnn: Enhancing file fragment type identification with recurrent and convolutional neural networks. IEEE Access
Sparck Jones K (1972) A statistical interpretation of term specificity and its application in retrieval. J Document 28(1):11–21
Sportiello L, Zanero S (2011) File block classification by support vector machine. In: 2011 Sixth international conference on availability, reliability and security, IEEE, pp 307–312
Sportiello L, Zanero S (2012) Context-based file block classification. In: IFIP international conference on digital forensics, Springer, pp 67–82
Thi NN, Cao VL, Le-Khac NA (2017) One-class collective anomaly detection based on lstm-rnns. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXVI, Springer, pp 73–85
TinyOS (2023) Open source operating system designed for low-power wireless devices. http://www.tinyos.net/, [Online; 2024/04/13 10:12:31]
Veenman CJ (2007) Statistical disk cluster classification for file carving. In: Third international symposium on information assurance and security, IEEE, pp 393–398
Vilone G, Longo L (2021) Notions of explainability and evaluation approaches for explainable artificial intelligence. Inform Fus 76:89–106
Wang F, Quach TT, Wheeler J et al (2018) Sparse coding for n-gram feature extraction and training for file fragment classification. IEEE Trans Inf Forensics Secur 13(10):2553–2562
Xu T, Xu M, Ren Y et al (2014) A file fragment classification method based on grayscale image. J Comput 9(8):1863–1870
Zhu N, Liu Y, Wang K, et al (2023) File fragment type identification based on cnn and lstm. In: Proceedings of the 2023 7th International Conference on Digital Signal Processing, pp 16–22

Publisher's Note