A File Fragment Classification Method Based on Grayscale Image

Tantan Xu
College of Computer, Hangzhou Dianzi University, Hangzhou, China
Email: xutan0@163.com

Ming Xu*, Yizhi Ren, Jian Xu, Haiping Zhang, Ning Zheng
College of Computer, Hangzhou Dianzi University, Hangzhou, China
Email: {mxu, renyz, jian.xu, zhanghp, nzheng@hdu.edu.cn}

***Abstract*—File fragment classification is an important and difficult problem in digital forensics. Previous works in this area mainly relied on specific byte sequences in file headers and footers, or statistical analysis and machine learning algorithms on data from the middle of the file. This paper introduces a new approach to classify file fragment based on grayscale image. The proposed method treats a file fragment as a grayscale image, and uses image classification method to classify file fragment. Furthermore, two models based on file-unbiased and type-unbiased are proposed to verify the validity of the proposed method. Compared with previous works, the experimental results are promising. An average classification accuracy of 39.7% in file-unbiased model and 54.7% in type-unbiased model are achieved on 29 file types.**

***Index Terms*—file fragment classification, digital forensic, grayscale image**

I. Introduction

There are too many file types in a computer. It is very easy to store various types of data files on the hard drives. However, in computer forensics field, the huge number of file types may bring troubles to investigators. It is very difficult to find out an unknown file type especially when the file data is not complete (due to the fact that it may be deleted and part of the file data has been overwritten), or is fragmented. According to Garfinkel [1], although today's modern operating systems try their best to avoid files to be fragmented, the fragmentized files are still very common. For example, there may be no large enough contiguous region on the media to store the file, or there may be no sufficient unallocated region to hold the appended data at the end of a file. In this work, a new approach for the problem of the file fragment classification is proposed. The proposed method regards a file fragment as a grayscale image, and uses image classification method to classify file fragment.

Perhaps the most commonly utlized file type identification program is the UNIX "file" command which mainly relies on the "libmagic" library [2]. It uses the magic numbers to recognize the file types. This method is rather accurate when given a complete file. However, when it comes to the file fragment, the command typically reports "ASCII text" or "data".

File extension is the fastest way to detect file type because there is no need to open the file. However, at the same time, it is easily spoofed by simply renaming the file extension. Most works on file fragment classification to date have attempted to solve this problem by using a combination of machine learning techniques and statistical analysis techniques [3]. They treat a file as a vector made up of a combination of n-gram [4], Hamming weight, Shannon entropy [5], mean byte value and so on [6-11]. Besides, there are other classification technologies used for file fragment classification such as byte frequency distribution (BFD) [12-14], the longest common subsequence [15], Kolmogorov Complexity [16-18] and rate of change between two consecutive bytes [19].

At a broader level, all data stored in drivers can be represented as a binary string made up of zeros and ones. This binary string can be reshaped into a matrix and regarded as a grayscale image. Therefore, the description of a grayscale image can be used for a description of the corresponding data fragment. The description of a picture has been well studied in the field of computer vision. GIST Descriptor [20, 21] which has performed excellently in scene classification and object classification is used in this paper to describe the grayscale images generated by file fragments. The descriptions of the images are then put into classification algorithms for training and testing. Ten-fold cross validation [22] is used to evaluate this approach. 29 file types are chosen from the publicly available corpus collected by Garfinkel el al. [23] based on how well-known they are to conduct experiments in file-unbiased model and type-unbiased model.

The rest of this paper is organized as follows. Section II provides a brief overview of related works done in this area. Section III introduces the method of generating grayscale images from file fragments and automatically classifying them. The experiments are detailed in section IV, and results and analyses are presented in section V. Section VI places conclusions and future works.

II. Related Work

doi:10.4304/jcp.9.8.1863-1870

McDaniel and Heydari [12] proposed an approach to generate "fingerprints" for files. They used three algorithms in their work: Byte Frequency Analysis (BFA), Byte Frequency Cross-Correlation (BFC) and the File Header/Trailer (FHT). The basic idea was using statistics frequency of ASCII code (0-255) in the file to classify files. Their corpus consisted of 120 files with 30 different types; only complete file instead of fragment was considered. According to their results, BFA and BFC showed inferior performance (27.50% true positive rate for BFA and 46% for BFC) compared with FHT (95.83%). However, further analysis showed that FHT was simply a variation on the magic numbers.

Li et al. [6] substantially revamped BFD using unigram counts of the prefix of a file. Their contributions focused on using a centroid or multiple centroids generated from BFD as the signature (fileprint in their work) of a file type. They collected a data set consisting of 8 different types. K-means clustering algorithm was applied to get models for every type. The authors only evaluated on fragments started at the beginning of files. Results showed that the method did not work well when classifying similar data types such as GIF and JPG.

Karresan and Shahmehri [24] proposed a method called OSCAR which was very similar to centroid idea mentioned above. They extended their work in article [19] with the idea called rate-of-change (RoC). RoC is the difference of the ASCII values between two consecutive bytes. Their work performed nearly perfectly on JPG but did poorly on other file types.

Veenman [25] used features made up of 1-gram counts, Shannon entropy and Kolmogorov complexity for linear discriminant analysis [26] to classify file fragments. He utilized a private data set consisting of 11 file types to do his experiments and achieved 45% for average classification accuracy. The fragment size he used was 4096 bytes.

Calhoun and Coles [15] achieved good accuracy of 88.3%. Their basic idea was similar to Veenman's. They employed more features than Veeman. The experiments they performed were on only 4 types (JPG, BMP, GIF and PDF). 2 of the 4 types were used to distinguish from each other in one run of their experiments.

Based on Kolmogorov Complexity, Axelsson [16, 17] proposed a method called Normalized Compression Distance (NCD) to solve the problem of file fragment classification. He applied k-nearest-neighbors (KNN) to measure the distances between different types. A publicly available data set collected by Garfinkel et al. was used in his experiments. The average classification accuracy they achieved was about 34% on 28 different file types.

Fitzgerald et al. [27] represented file fragments as feature vectors consisting of unigram and bigram counts, as well as other statistical measurements such as Shannon entropy, the average contiguity between bytes to classify file fragments. The classification algorithm they used was SVM. They achieved an average classification accuracy of 49.1% on 24 file types in their experiments.

Conti et al. [28] seems to have taken the first step to make fragments visualization. They treated fragments from files as grayscale images. However, they did not give an automated approach to classify fragments through grayscale images they got. This paper extends their work and provides a preliminary solution to this classification problem.

Nataraj et al. [29, 30] proposed a visualization method for malware classification by using binary texture analysis. They only needed to consider the executable file type (EXE). While in this paper, different file types are involved. They treated an entire malware file as a grayscale image. However, what we face are file fragments instead of the whole file.

III. The Proposed Method

A. Transforming Fragment into Grayscale Image

As mentioned above, all data stored in drives can be represented as a binary string made up of zeros and ones. Considering a grayscale image of 256 gray-level, every 8 bits in the data can be viewed as a pixel in a grayscale image. Every grayscale image has its width and height. Therefore, it is necessary to reshape "pixels" into a 2D array to make them constitute an image instead of a line. Let *grayscale*(*i*,*j*) represents the gray value of *i*th row and *j*th column in the grayscale image after reshaping. The data in a fragment before reshaping is stored in a one-dimensional array *fragment*. The corresponding reshaping formula is defined as follows:

$$grayscale\ (i, j) = fragment(i \times width + j) \quad (1)$$

Here *width* is the width of the grayscale image after reshaping (the most appropriate width of the grayscale image will be discussed in experiments). Figure 1 shows the steps to convert file fragment to grayscale image.

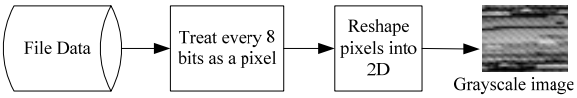

Figure 1. Converting File Fragment to Grayscale image

B. Describing Image

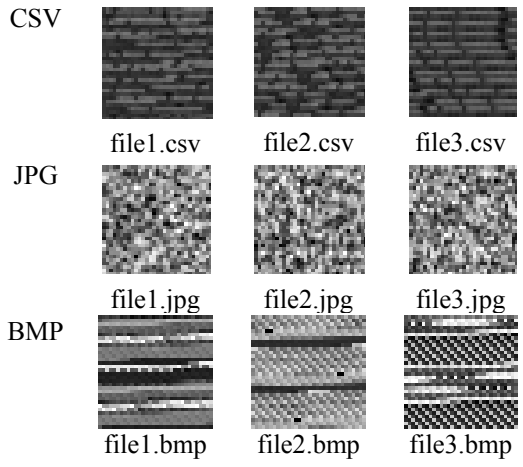

Figure 2. Grayscale Images Examples

Figure 2 shows nine grayscale image examples from different file types. All of the images are converted from 1024 bytes fragments with 32 pixels (32 bytes in file fragments) in width. These images can be classified into correct types easily by human eyes. Therefore, if computers can view pictures like human beings, file fragments

may be classified based on their corresponding descriptions of grayscale images.

Fortunately, the descriptions of images have been well studied in the field of computer vision. In [20], Oliva and Torralba proposed a computational model of the recognition of the real-world scene. They proposed a series of perceptual dimensions that represent the dominant spatial structure of a scene. To compute grayscale image's features, GIST [21] which uses wavelet decomposition to analyze images is used in this paper.

GIST feature is commonly used in image recognition and large scale image search. The following is a brief introduction of the GIST feature vector. A more detailed explanation can be found in [20]. The location of every image is represented by the output of filters tuned to different orientations and scales. In this work, a commonly used steerable pyramid with 4 scales and 8 orientations is applied to the grayscale image. Then, the local representation of an image $I^L(x)$ is given by:

$$I^L(x) = \{i_1(x), i_2(x), \ldots, i_j(x), \ldots, i_N(x)\} \quad (2)$$

Here $i_j(x)$ is the representation of *j*th sub-band and *N* is the total number of sub-bands. In order to obtain the global image properties while keeping some local information, the mean magnitude value of the local features *mmv*(*x*) is computed and averaged over large spatial regions:

$$mmv(x) = \sum_{x'} I^L(x')aw(x' - x) \quad (3)$$

Here *aw*(*x*) is the averaging window. The resulting representation is down sampled to have a spatial resolution of $M \times M$ pixels. Both *N* and *M* are the default values in GIST descriptor where $N = 32$ and $M = 4$. At last, the size of feature vector *mmv* acquired is $M \times M \times N = 512$.

C. Finding the Best Classifying Algorithm

After obtaining the descriptions of file fragments, the vectors are put into different classification algorithms to find the best classification algorithm which fits for file fragment classification. In this work, 6 commonly used classification algorithms are evaluated. The 6 classification algorithms and their brief introductions are listed as follows.

KNN: K-Nearest Neighbors algorithm is one of the simplest algorithms in machine learning field. It is easy to understand this algorithm: an unknown instance is classified by a majority vote of its neighbors, with the instance being attached to the class most common among its K nearest neighbors.

Naïve Bayes: Bayes classification is a statistics based algorithm. It predicts the possible attribute of an unknown instance and selects the maximum likelihood class as its class based on bayes thorem. To simplify the complexity, Naïve Bayes assumes all attributes have an independent effect on the specific class.

SVM: A Support Vector Machine is a supervised learning model with related learning algorithm that analyzes data and recognizes patterns, used for classification and regression analysis. A support machine creates a hyperplane or series of hyperplanes in a high- or infinite-dimensional space. An excellent segmentation is implemented by the hyperplane that achieves the largest distance between different classes, since in general the larger the margin the better the result is. In this paper, three commonly used SVM algorithms: LIBSVM, LIBLINEAR and SMO are involved.

Decision Tree: A decision tree is a tool helps to make decisions. It utilizes a tree-like graph or model of decisions and their possible results, including chance event outcomes, resource consumption, and utility. Decision trees are frequently employed in operations research, specifically in decision analysis, to assist a strategy most likely to achieve the target. A well-known J48 decision tree algorithm is used in this paper.

D. Classifying Step

The file fragment classification process can be summarized in the following steps:

(1) For every file fragment do{
  (a) Reading every byte from file fragment and writing it into a one-dimensional array;
  (b) Converting the one-dimensional array into a two-dimensional matrix to get a grayscale image using formula (1);
  (c) Computing a 512-dimensional GIST feature to describe the grayscale image by using GIST Descriptor;
}
(2) Using a classification algorithm to classify all of the 512-dimensional vectors.

IV. Experimental Setup

A. Data Set

TABLE I
FILE TYPES USED IN EXPERIMENTS

BMP	CSV	DOC	DOCX	EPS	GIF
GZ	HTML	JAVA	JPG	LOG	PDF
PNG	PPS	PPT	PPTX	PS	PUB
RTF	SQL	SWF	TEX	TTF	TXT
WP	XBM	XLS	XLSX	XML	

Most of the previous research used private data sets. This is not good for promoting friendly competition between approaches in the field. Therefore, Axelsson's step is followed in this paper to use a freely available corpus. The data set used for training and testing in this paper is generated from the freely available corpus of forensic research data collected by Garfinkel et al. [23](the govdocs1 data set described in section 4 of the cited paper). That corpus contains 1 million files of different file types. Among previous works based on this corpus, Axelsson selected 28 file types and Fitzgerald et al. selected 24 file types. While in this paper, 29 file types which cover all file types except for JAR, ZIP and JS in previous works are selected to evaluate the method proposed. In addition, two more common types LOG and WP are added to data set in this paper. The reasons that JAR, ZIP and JS type files are not selected are detailed later. To make the results more convincible, it is ensured that the selected file extension and its claimed type are consistent when select-

ing file from the corpus. Table I shows the selected 29 file types used in this paper.

There are no JAR extension files in the govdocs1 data set. 17 files with extension HTML and other 17 files with extension TXT are claimed to be JAR files in govdocs1. Further investigation shows that the 17 files with extension HTML are actually HTML files, and the 17 files with extension TXT are actually text document that maybe change log for some JAR files. As for ZIP files, there are 10 files whose extension are ZIP. Of the 10 files, only 1 file is a ZIP file with a DOC file compressed in it. The extra 9 files consist of 1 DOCX file, 4 XLSX files and 4 PPTX files. When searching for files claimed to be ZIP files in govdocs1, 26 files with HTML extension are found. All the files are checked to make sure their type and their extension are consistent. The only remaining 1 ZIP file is not enough to do experiments. Such is the case of JS type.

B. Fragment

The fragment size adopted in this paper is 1024 bytes. There are three reasons prompt the authors to employ 1024 bytes as fragment size. To begin with, it is difficult to analyze the texture of a grayscale obtained from a fragment less than 1024 bytes because its texture is not clear. In addition to that, larger block size will help improve disk I/O performance when using large files, therefore, file systems commonly used today have a larger block size than 1024 bytes instead of 512 bytes in disk sector size. One more thing, the DFRWS'13 data sniffing challenge [31] also chose 1024 bytes as the minimum block size.

When generating fragments, the first and last fragments of each file are omitted, as the first fragment frequently contains header information, and the last fragment might not be 1024 bytes in length. Calhoun and Coles [15], Conti et al. [10], Fitzgerald et al. [27] also removed the first and last fragments.

C. File-unbiased and Type-unbiased Model

When conducting experiments, 2 different models are designed. The first model is called file-unbiased model. That is, an equal number of fragments are randomly selected from every file. 10 fragments per file are randomly chosen in this paper, except for where the file is not sufficiently long. The second model is type-unbiased model. In this model, an equal number of fragments are randomly selected from every type. In this work, more than 10 files that altogether consist of more than 8000 fragments are randomly selected from govdocs1. Of the over 8000 fragments, 1000 fragments per type are randomly selected.

D. Experiments

The classification experiments in this paper are conducted in WEKA. WEKA is a collection of machine learning algorithms for data mining tasks. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes [32]. All classification algorithms used in this paper can be found in WEKA. Ten-fold cross validation [22] provided by WEKA is used to evaluate the approach proposed in this work.

Four experiments are conducted in both file-unbiased model and type-unbiased model. The first experiment is to find the best classification algorithm that fits for file fragment classification. After determining the most suitable classification algorithm, the effect of grayscale image width on classification accuracy is discussed. Then the image description vectors are reduced in their dimensions using PCA [33]. This experiment is based upon the idea that there may be some redundant information in the image description. At the same time, dimensionality reduction can reduce the time required for classification. At last, grid searching is applied to figure out the optimal combination of classification algorithm parameters and feature dimensions. The reason that the 4 experiments are all conducted in both models is to make the two models to verify each other.

V. Results and Analyses

The first experiment is to find the most suitable classification algorithm. Table II shows the performance of 6 classification algorithms mentioned above in file-unbiased model. All classification algorithms use default configuration in WEKA. The width of grayscale images is 32 pixels (32 bytes in file fragments). Results show that KNN is really fast to build the model required for classification because of its simplicity. As can be seen from the table, the performance of KNN is the best in most cases. Therefore, it's obvious that KNN is the most suitable algorithm for file fragment classification or at least for the situation in this paper. Results in type-unbiased model also support this conclusion. The results table is not listed here for the limited space. The conclusion that KNN is the most suitable algorithm for file fragment classification is similar to the conclusion in paper [7].

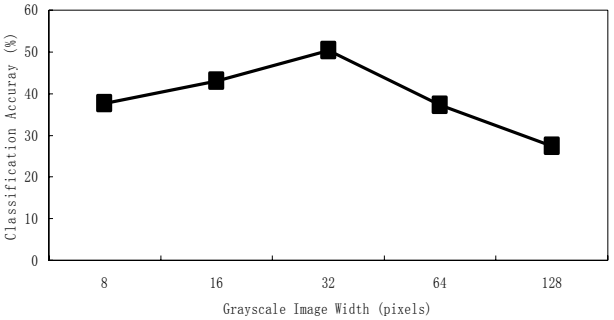

Figure 3. Classification Accuracy vs. Grayscale Image Width in Type-unbiased Model

To evaluate the effect of grayscale image width on classification accuracy, experiment with different grayscale image widths is conducted. The classification algorithm used in this experiment is KNN which performs best in the former experiment. Figure 3 shows 32 pixels (32 bytes in file fragments) is the most appropriate width of the grayscale image when the fragment size is 1024 bytes in type-unbiased model. Such is the case of file-unbiased model.

GIST produces a feature vector of 512 dimensions wh-

TABLE II
PERFORMANCE OF 6 CLASSIFICATION ALGORITHMS IN FILE-UNBIASED MODEL

Classifier	TPR(%)	FPR(%)	Precision(%)	Recall(%)	ROC Area	Time to build model(seconds)
KNN	37.1	2.4	37.2	37.1	0.68	0.01
Naïve Bayes	27.8	2.6	25.3	27.8	0.84	0.58
LIBSVM	14.2	3.3	18.5	14.2	0.56	202.17
LIBLINEAR	28.3	2.7	25.6	28.3	0.63	38.32
SMO	33.7	2.5	35.1	33.7	0.85	17.63
J48	22.0	2.9	22.9	22.0	0.61	46.20

TABLE III
USING GRID SEARCHING TO FIGURE OUT THE OPTIMAL COMBINATION IN FILE-UNBIASED MODEL

Dimensions	2	3	4	5	6	7	8	**9**	10	11	12	13	14	15
K	10	7	5	6	1	5	7	**1**	1	1	1	1	1	1
Classification Accuracy (%)	29.6	36.2	38.8	38.2	38.3	39.6	39.3	**39.7**	39.4	39.4	38.9	39.1	38.9	36.7

TABLE IV
USING GRID SEARCHING TO FIGURE OUT THE OPTIMAL COMBINATION IN TYPE-UNBIASED MODEL

Dimensions	2	3	4	5	6	7	8	**9**	10	11	12	13	14	15
K	10	9	10	9	9	9	8	**7**	5	7	7	6	9	1
Classification Accuracy (%)	40.3	49.5	52.1	51.7	53.0	54.5	54.6	**54.7**	54.4	53.7	54.0	53.9	53.1	51.9

en describing a grayscale image. Figure 4 shows classification accuracy after dimensionality reduction using PCA in file-unbiased model. The classification algorithm used is KNN. The width of grayscale images is 32 pixels (32 bytes in file fragments). As can be seen from the figure, when the feature dimensions exceed 15, the increment of the dimensions does not seem to bring the improvement of classification accuracy. This conclusion will be used in the next experiment to find the optimal combination of classification algorithm parameters and feature dimensions. The same phenomenon can be seen in type-unbiased model.

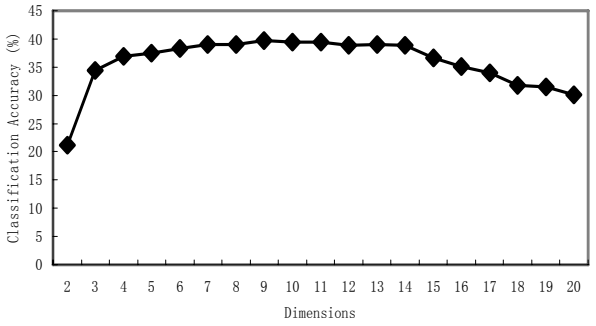

Figure 4. Classification Accuracy vs. Feature Dimensions in file-unbiased model

In the final experiment, all results in both models are presented for comparison and analyses. This experiment is to figure out the optimal combination of classification algorithm parameters and feature dimensions by using grid searching in both two models. The classification algorithm used is KNN with *K* ranging from 1 to 10. The feature dimensions in this experiment increase from 2 to 15. Table III shows that the best average classification accuracy acquired is 39.7% with *K* being 1 in 9 dimensions in file-unbiased model. While in type-unbiased model, the best average classification accuracy is 54.7% with *K* being 7 in 9 dimensions (see table IV).

Figure 5 and figure 6 show the confusion matrix when using the optimal combination in the two models. The classification accuracy in this paper is significantly better than random chance ($1/29 \approx 3.4\%$). The preliminary results in this paper are promising when comparing with previous work. Axelsson [16] achieved best average classification accuracy of 34% on 28 file types using file-unbiased model, while utilizing the same model, the approach in this paper achieved an average classification accuracy of 39.7% on 29 file types. Veenman [25] achieved an average accuracy of 45%, but only on 11 file types. The model they used is similar to type-unbiased model. Fitzgerald et al. [27] achieved best average classification accuracy of 49.1% on 24 file types using type-unbiased model. While in this paper, an average classification accuracy of 54.7% on 29 file types is achieved using the same model.

It can be seen from confusion matrixes in figure 5 and figure 6 that the method proposed in this paper performs nearly perfectly in some types such as TTF and XBM. Generally speaking, the classifier here performs better on low entropy file fragments than high entropy file fragments, which reflect findings in previous works. The goal of this paper is to test the feasibility of the idea that whether the type of a file fragment can be detected by description of its corresponding grayscale image. The preliminary results to this end show that the approach is promising. However, the method which is used to describe file grayscale images is taken from computer vision without any optimization. It is worth careful optimization to make the method more suitable for file fragment classification in the future.

VI. Conclusions and Future Work

Classifying file fragment is an important and difficult problem in digital forensics. In this work, a new approach that treats binary data of a file fragment as a grayscale image is explored to deal with the problem of file fragment classification. To describe the grayscale image obtained from a file fragment, GIST which is borrowed from computer vision is used without any optimization. The preliminary results in this paper are promising. Hence, it is worth careful optimization when describing the file grayscale image to make the method more suitable for file fragment classification in the future.

GIST utilized here is borrowed from computer vision that is suited for recognition of real-world scenes. Therefore, another area of future work is to find out a more targeted way [34, 35] to classify grayscale images obtained from original binary data fragments. For example, lossy compression [36] may be used to extract the domi-

	gz	jpg	swf	docx	xlsx	pptx	ppt	pps	png	pdf	doc	tex	eps	sql	java	html	txt	gif	wp	rtf	csv	ps	pub	xls	log	xml	bmp	ttf	xbm
gz	4.0	18.0	12.0	12.0	9.0	8.0	3.0	10.0	11.0	3.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0
jpg	12.0	22.0	7.0	9.0	3.0	9.0	5.0	8.0	9.0	3.0	4.0	1.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	1.0	4.0	0.0	0.0	0.0	0.0	0.0
swf	8.0	7.0	21.0	7.0	8.0	8.0	5.0	8.0	15.0	4.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	5.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
docx	12.0	15.0	6.0	12.0	7.0	8.0	4.0	3.0	14.0	2.0	4.0	0.0	2.0	0.0	0.0	1.0	0.0	5.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	1.0	0.0	0.0
xlsx	8.0	9.0	7.0	14.0	7.0	9.0	7.0	9.0	9.0	4.0	2.0	0.0	1.0	1.0	0.0	1.0	1.0	4.0	2.0	0.0	0.0	1.0	0.0	3.0	0.0	0.0	1.0	0.0	0.0
pptx	14.0	12.0	8.0	11.0	8.0	14.0	7.0	3.0	11.0	1.0	2.0	0.0	0.0	0.0	0.0	1.0	0.0	4.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
ppt	6.0	8.0	4.0	4.0	9.0	4.0	11.0	10.0	5.0	4.0	6.0	0.0	3.0	1.0	1.0	1.0	0.0	5.0	8.0	0.0	1.0	1.0	3.0	3.0	1.0	0.0	1.0	0.0	0.0
pps	8.0	7.0	8.0	5.0	10.0	5.0	14.0	13.0	6.0	3.0	4.0	0.0	1.0	1.0	0.0	1.0	0.0	3.0	3.0	1.0	1.0	1.0	3.0	2.0	0.0	0.0	0.0	0.0	0.0
png	8.0	10.0	10.0	10.0	5.0	9.0	9.0	10.0	15.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
pdf	7.0	10.0	7.0	2.0	6.0	3.0	7.0	2.0	8.0	22.0	2.0	9.0	1.0	0.0	1.0	1.0	1.0	1.0	2.0	1.0	2.0	2.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
doc	2.0	3.0	2.0	7.0	0.0	0.0	4.0	4.0	0.0	2.0	25.0	16.0	0.0	1.0	3.0	4.0	7.0	1.0	4.0	4.0	1.0	3.0	0.0	5.0	1.0	0.0	1.0	0.0	0.0
tex	0.0	1.5	0.0	1.5	0.0	0.0	0.0	0.0	0.0	3.0	13.4	31.3	0.0	1.5	1.5	1.5	19.4	0.0	4.5	11.9	3.0	3.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0
eps	0.0	0.0	0.0	3.0	1.0	0.0	3.0	1.0	0.0	4.0	3.0	2.0	43.0	1.0	2.0	11.0	2.0	0.0	1.0	2.0	1.0	9.0	3.0	1.0	3.0	1.0	3.0	0.0	0.0
sql	0.0	0.0	0.0	0.0	2.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	3.0	52.0	2.0	13.0	3.0	1.0	1.0	3.0	4.0	2.0	0.0	1.0	6.0	2.0	0.0	0.0	1.0
java	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	2.0	3.0	3.0	54.0	9.0	6.0	0.0	1.0	0.0	6.0	0.0	5.0	0.0	6.0	2.0	0.0	0.0	0.0
html	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	7.0	1.0	7.0	9.0	34.0	13.0	0.0	3.0	7.0	1.0	0.0	0.0	2.0	4.0	8.0	0.0	0.0	0.0
txt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	4.0	10.0	0.0	2.0	4.0	11.0	48.0	0.0	0.0	10.0	4.0	1.0	2.0	0.0	1.0	2.0	0.0	0.0	0.0
gif	4.0	9.0	7.0	4.0	5.0	8.0	1.0	5.0	6.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	45.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
wp	1.0	0.0	3.0	1.0	0.0	2.0	3.0	2.0	1.0	0.0	3.0	3.0	1.0	5.0	7.0	1.0	3.0	2.0	47.0	2.0	1.0	6.0	2.0	3.0	1.0	0.0	0.0	0.0	0.0
rtf	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	8.0	0.0	5.0	1.0	6.0	8.0	0.0	1.0	46.0	2.0	3.0	1.0	1.0	9.0	1.0	1.0	0.0	0.0
csv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	9.0	9.0	5.0	4.0	0.0	2.0	5.0	39.0	1.0	4.0	1.0	18.0	0.0	0.0	0.0	0.0
ps	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	2.0	3.0	3.0	5.0	2.0	0.0	4.0	1.0	0.0	6.0	5.0	1.0	57.0	0.0	0.0	1.0	3.0	1.0	0.0	0.0
pub	0.0	3.6	1.8	5.4	0.0	0.0	3.6	0.0	1.8	0.0	0.0	0.0	0.0	0.0	7.1	0.0	3.6	0.0	0.0	0.0	7.1	0.0	64.3	0.0	0.0	0.0	1.8	0.0	0.0
xls	2.0	1.0	0.0	2.0	2.0	2.0	4.0	3.0	0.0	1.0	9.0	0.0	3.0	0.0	0.0	2.0	0.0	0.0	2.0	0.0	1.0	0.0	0.0	64.0	0.0	2.0	0.0	0.0	0.0
log	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	6.0	1.0	2.0	0.0	0.0	2.0	4.0	7.0	2.0	0.0	0.0	72.0	1.0	0.0	0.0	0.0
xml	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	2.0	1.0	3.0	0.0	8.0	2.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	77.0	0.0	0.0	0.0
bmp	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	2.0	1.0	1.0	0.0	2.0	0.0	0.0	1.0	3.0	0.0	3.0	1.0	0.0	0.0	81.0	0.0	0.0
ttf	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0
xbm	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0

Figure 5. Confusion Matrix When Using Optimal Combination of Classification Algorithm Parameters and Feature Dimensions in File-unbiased Model(%)

nant information of a file grayscale image. This is our ongoing work.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation Natural Science Foundation of China under Grant No.61070212, 61003195 and 61100194, the Zhejiang Province Natural Science Foundation Natural Science Foundation of China under Grant No.Y1090114 and LY12F02006, the Zhejiang Province key industrial projects in the priority themes of China under Grant No 2010C11050, the science and technology search planned projects of Zhejiang Province (No.2012C21040) , Scientific Research Fund of Zhejiang Provincial Education Department (Grant No. Y201120356).

REFERENCES

[1] S. L. Garfinkela, "Carving contiguous and fragmented files with fast object validation," in Digital Forensics Research Conference Pittsburgh, PA, 2007, pp. 2 - 12.

[2] I. F. Darwin, "Libmagic,", ftp://ftp.astron.com/pub/file, 2008

[3] V. Roussev and S. L. Garfinkel, "File Fragment Classification—The Case for Specialized Approaches," in Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering Berkeley, CA, 2009, pp. 3-14.

[4] M. Damashek, "Gauging similarity: with n-grams Language-independent categorization of text," Science, vol. 267, pp. 843 - 848, 1995.

[5] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, pp. 379 - 423, 1948.

[6] W. Li and K. W. S. J, "Fileprints: identifying file types by n-gram analysis," in Sixth Annual IEEE SMC Information Assurance Workshop United States Military Academy, West Point, NY, 2005, pp. 64-71.

[7] I. Ahmed, K. Lhee, H. Shin, and M. Hong, "Fast Content-Based File Type Identification," in nternational Confer-

ence on Digital Forensics. vol. Volume 361/2011 Orlando, FL, USA, 2011, pp. 65-75.
[8] S. J. Moody and R. F. Erbacher, "SÁDI – Statistical Analysis for Data type Identification," in Third International Workshop on Systematic Approaches to Digital Forensic Engineering Oakland, CA, 2008, pp. 41-54.
[9] I. Ahmed, K. Lhee, H. Shin, and M. Hong, "Fast File-type Identification," in ACM Symposium on Applied Computing, 2010, pp. 1601-1602.
[10] G. Conti, S. Bratus, A. Shubina, B. Sangster, R. Ragsdale, M. Supan, A. Lichtenberg, and R. Perez-Alemany, "Automated mapping of large binary objects using primitive fragment type classification," in Digital Forensics Research Conference Portland, Oregon, 2010, pp. S3 - S12.
[11] G. A. Hall and W. P. Davis, "Sliding Window Measurement for File Type Identification," in IEEE workshop on Information Assurance Workshop, 2006.
[12] M. McDaniel and M. H. Heydari, "Content Based File Type Detection Algorithms," in 36th Annual Hawaii International Conference on System Sciences Hawii, 2003.
[13] M. C. Amirani, M. Toorani and A. A. B. Shirazi, "A New Approach to Content-based File Type Detection," in IEEE Symposium on Computers and Communications Marrakech, 2008, pp. 1103-1108.
[14] I. Ahmed, K. Lhee, H. Shin, and M. Hong, "On Improving the Accuracy and Performance of Content-Based File Type Identification," in Information Security and Privacy. vol. Volume 5594/2009 Australia, 2009, pp. 44 - 59.
[15] W. C. Calhoun and D. Coles, "Predicting the types of file fragments," in Digital Forensics Research Conference Baltimore, MD, 2008, pp. S14 - S20.
[16] S. Axelsson, "The Normalised Compression Distance as a file fragment classifier," in Digital Forensics Research Conference University Place Hotel and Conference Center near Portland State University, 2010, pp. S24 - S31.
[17] S. Axelsson, "Using Normalized Compression Distance for Classifying File Fragments," in International Conference on Availability, Reliability, and Security Krakow, 2010, pp. 641 - 646.
[18] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," Problems of Information Transmission, vol. 1, pp. 3 - 11, 1965.
[19] M. Karresand and N. Shahmehri, "File Type Identification of Data Fragments by Their Binary Structure," in IEEE Workshop on Information Assurance West Point, NY, United states, 2006, pp. 140-147.
[20] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," International Journal of Computer Vision, vol. 42, pp. 145 - 175, 2001.
[21] "GIST Code", http://people.csail.mit.edu/torralba/code/spatialenvelope/.
[22] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in International Joint Conferences on Artificial Intelligence Montreal, Quebec, Canada, 1995, pp. 1137 - 1143.
[23] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing Science to Digital Forensics with Standardized Forensic Corpora," in Digital Forensics Research Conference Montreal, Canada, 2009, pp. S2 - S11.
[24] M. Karresand and N. Shahmehri, "Oscar—file type identification of binary data in disk clusters and ram pages," in Security and Privacy in Dynamic Environments Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006), 22–24 May 2006, Karlstad, Sweden. vol. Volume 201/2006, 2006, pp. 413-424.
[25] C. J. Veenman, "Statistical Disk Cluster Classification for File Carving," in Third International Symposium on Information Assurance and Security, 2007, pp. 393-398.
[26] R. A. Fisher, "The use of multiple measurements in taxonomic problems," Annals of Eugenics, vol. 7, pp. 179 - 188, 1936.
[27] S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn, "Using NLP techniques for file fragment classification," in Digital Forensics Research Conference Washington, DC USA, 2012, pp. S44-S49.
[28] G. Conti, S. Bratus, A. Shubina, A. Lichtenberg, R. Ragsdale, R. Perez-Alemany, B. Sangster, and A. M. Supan, "A Visual Study of Primitive Binary Fragment Types," in Black Hat USA, 2010.
[29] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware Images: Visualization and Automatic Classification," in 8th International Symposium on Visualization for Cyber Security Pittsburgh, Pennsylvania, 2011, pp. 1 - 7.
[30] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in 4th ACM workshop on Security and artificial intelligence Chicago, Illinois, USA, 2011, pp. 21 - 30.
[31] DFRWS, "The 2013 Data Sniffing Challenge," in Digital Forensics Research Conference Monterey, CA, 2013.
[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," SIGKDD Explorations, vol. 11, pp. 10 - 18, 2009.
[33] S. Wold, K. Esbensen and P. Geladi, "Principal Component Analysis," Chemometrics and Intelligent Laboratory Systems, vol. 2, pp. 37 - 52, 1987.
[34] Z. Wen, Y. Hu and W. Zhu, "Research on Feature Extraction of Halftone Image," Journal of Software, vol. 10, pp. 2575-2580, 2013.
[35] Y. Lan, Y. Zhang and H. Ren, "A Combinatorial K-View Based Algorithm for Texture Classification," Journal of Software, vol. 8, pp. 218-227, 2013.
[36] Y. Bian, L. Wu, C. Yang, and B. Li, "Framework of A Hybrid Image Compression Algorithm for View-dependent Visualization," Journal of Software, vol. 8, pp. 2208-2212, 2013.

Tantan Xu received the B.S. degree in software engineering from the Guiyang University in 2011. He is currently a master candidate in Technology of Computer Application from the Hangzhou Dianzi University, P. R. China. His research interest includes Intelligent Information Processing System, and Digital Forensic.

Ming Xu is a Professor in the college of Computer, Hangzhou Dianzi University, P. R. China. He received the doctor degree in computer science and technology from the Zhejiang University in 2004. His research interests include Digital Forensics, Network Security, Social Network and Data Mining.

Yizhi Ren is a Lecturer in the college of Computer, Hangzhou Dianzi University, P. R. China. He received the doctor degree in computer science and technology from the Dalian University of Technology in 2011. His research interests include Network Security, Social Computing and Evolutionary game.

Jian Xu is a Professor in the college of Computer, Hangzhou Dianzi University, P. R. China. He received the doctor degree in computer science and technology from the Zhejiang University

in 2004. His research interests include Location-based Services, Mobile Computing, Distributed Database and Network Security.

Haiping Zhang is an Associate Professor in the college of Computer, Hangzhou Dianzi University, P. R. China. He received the master degree in Computer Software and Theory from the Hangzhou Dianzi University in 2005. His research interests include Digital Forensics, Network Security, Social Network and Corporate Information Technology.

Ning Zheng is a Professor in the college of Computer, Hangzhou Dianzi University, P. R. China. His research interests include Network Security, CAD, and CAM.

	gz	jpg	swf	docx	xlsx	pptx	ppt	pps	png	pdf	doc	tex	eps	sql	java	html	txt	gif	wp	rtf	csv	ps	pub	xls	log	xml	bmp	ttf	xbm
gz	33.1	18.4	4.3	6.8	9.3	6.4	2.6	4.0	2.0	6.5	2.5	0.0	0.0	0.0	0.0	0.0	0.0	2.4	0.0	0.0	0.0	0.0	1.7	0.0	0.0	0.0	0.0	0.0	0.0
jpg	23.8	23.1	8.0	7.8	6.7	7.3	3.1	3.1	2.9	7.3	3.5	0.0	0.0	0.0	0.0	0.0	0.0	1.8	0.0	0.0	0.0	0.0	1.6	0.0	0.0	0.0	0.0	0.0	0.0
swf	11.7	13.6	25.4	4.9	5.6	3.9	4.0	4.1	12.5	4.9	2.6	0.2	0.0	0.0	0.9	0.3	0.2	1.7	1.7	0.0	0.6	0.1	1.1	0.0	0.0	0.0	0.0	0.0	0.0
docx	14.7	15.1	7.1	17.5	8.0	15.3	2.5	2.8	4.2	4.3	2.4	0.0	0.3	0.0	0.2	0.2	0.0	1.3	1.6	0.0	0.0	0.3	1.7	0.0	0.5	0.0	0.0	0.0	0.0
xlsx	17.0	15.4	9.1	7.8	14.9	6.3	2.9	3.9	4.3	6.8	2.7	0.0	0.0	0.0	0.2	0.1	0.4	3.4	1.5	0.2	0.2	0.0	1.8	0.4	0.3	0.3	0.1	0.0	0.0
pptx	16.3	13.4	7.3	18.6	8.4	15.4	2.6	3.6	3.7	4.7	2.3	0.0	0.1	0.0	0.0	0.0	0.1	1.6	0.6	0.1	0.1	0.0	1.0	0.0	0.1	0.0	0.0	0.0	0.0
ppt	11.2	10.9	7.4	4.9	5.5	4.4	14.9	5.9	12.7	4.6	2.4	0.5	0.1	0.0	0.4	2.0	1.2	1.5	3.6	0.9	0.0	0.5	1.5	1.2	0.8	0.1	0.9	0.0	0.0
pps	15.1	10.8	7.9	8.0	7.9	6.3	7.6	11.2	5.1	4.4	2.8	0.1	0.1	0.0	0.2	0.2	0.4	2.2	3.6	0.2	0.6	0.6	2.1	1.0	0.6	0.1	0.9	0.0	0.0
png	6.3	9.0	15.9	4.2	4.9	4.4	10.4	3.3	33.7	3.5	1.7	0.0	0.0	0.0	0.0	0.5	0.2	0.9	0.2	0.0	0.0	0.1	0.8	0.0	0.0	0.0	0.0	0.0	0.0
pdf	18.6	17.5	9.3	6.1	6.9	4.8	2.7	4.1	5.1	12.0	3.4	0.1	0.4	0.2	1.4	0.5	1.5	1.5	1.1	0.0	0.7	0.3	1.2	0.0	0.2	0.2	0.2	0.0	0.0
doc	12.0	14.2	7.0	3.3	4.2	2.7	4.0	5.3	3.9	5.8	21.7	0.8	0.1	2.6	0.3	0.6	0.5	1.3	2.1	0.9	0.7	0.8	0.9	1.5	0.2	0.4	2.2	0.0	0.0
tex	0.0	0.0	0.6	0.0	0.6	0.0	1.3	0.0	0.6	0.0	0.6	45.0	0.0	0.0	6.3	30.6	5.6	0.0	1.9	1.9	1.3	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
eps	0.0	0.0	0.3	0.5	0.3	0.1	0.4	0.0	0.3	0.3	0.0	0.0	92.1	0.2	0.7	0.2	0.4	0.1	0.9	0.7	0.1	1.9	0.0	0.0	0.0	0.5	0.0	0.0	0.0
sql	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.2	0.0	0.0	90.8	2.5	0.4	0.2	0.0	0.0	0.6	2.2	0.0	0.1	0.0	2.3	0.6	0.0	0.0	0.0
java	0.0	0.0	0.2	0.0	0.0	0.2	0.2	0.0	0.2	0.1	0.1	0.2	0.2	5.7	59.4	4.4	3.5	0.0	0.6	0.6	14.9	0.4	0.6	0.0	1.8	6.4	0.3	0.0	0.0
html	0.0	0.0	0.1	0.2	0.1	0.1	0.2	0.0	0.0	0.2	0.2	6.5	0.0	0.7	2.4	80.6	1.8	0.0	0.1	4.6	0.4	0.8	0.0	0.2	0.1	0.7	0.0	0.0	0.0
txt	0.1	0.0	0.5	0.1	0.0	0.3	0.6	0.1	0.5	0.3	0.1	2.2	0.0	2.3	5.7	4.7	78.0	0.0	0.4	0.7	1.0	0.7	0.0	0.4	0.1	0.9	0.3	0.0	0.0
gif	7.3	6.9	3.6	2.7	6.4	2.3	1.4	1.7	1.8	4.4	1.5	0.0	0.0	0.0	0.0	0.0	0.0	59.4	0.1	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0
wp	0.0	0.1	2.0	0.4	0.3	0.5	2.7	2.7	0.9	0.5	1.8	1.1	0.1	0.6	5.5	0.8	1.2	0.5	68.4	0.5	3.0	1.7	0.1	0.8	2.4	0.5	0.9	0.0	0.0
rtf	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	1.9	0.0	1.8	0.6	6.7	0.3	0.0	0.4	79.4	1.0	2.6	0.1	0.5	3.3	0.2	0.9	0.0	0.0
csv	0.0	0.0	0.0	0.1	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.1	0.0	3.2	12.7	2.4	0.8	0.0	0.7	3.1	70.0	0.2	0.1	0.0	5.3	1.1	0.0	0.0	0.0
ps	0.1	0.3	0.6	0.4	0.2	0.5	0.8	0.4	0.5	0.7	0.4	1.8	1.9	0.1	1.2	3.9	1.6	0.0	1.0	1.7	1.4	74.3	0.7	0.1	0.7	0.3	4.4	0.0	0.0
pub	14.9	11.2	2.6	3.6	5.2	5.1	2.2	5.2	1.4	4.2	3.2	0.0	0.0	0.0	1.2	0.0	1.4	0.4	0.9	0.1	1.6	0.6	32.8	1.2	0.0	0.1	0.7	0.0	0.1
xls	0.2	0.0	0.2	0.2	0.2	0.0	1.4	0.7	0.1	0.0	1.2	0.0	0.0	0.1	0.0	0.1	0.0	0.0	1.5	0.5	0.1	0.1	0.7	90.3	0.1	1.7	0.6	0.0	0.0
log	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.6	1.5	0.0	0.2	0.0	0.0	1.0	3.2	0.0	0.0	0.0	91.3	0.1	0.0	0.0	0.0
xml	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.1	0.0	0.1	0.1	0.0	1.6	3.0	0.8	0.2	0.0	0.1	1.8	0.5	0.0	0.6	0.0	0.2	90.5	0.1	0.0	0.0
bmp	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.4	0.0	0.0	0.1	0.2	0.0	0.6	0.1	0.5	1.9	0.6	3.9	1.0	0.1	0.1	0.2	90.1	0.0	0.0
ttf	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0
xbm	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.6

Figure 6. Confusion Matrix When Using Optimal Combination of Classification Algorithm Parameters and Feature Dimensions in Type-unbiased Model(%)