Azerbaijan Journal of High Performance Computing, Vol 1, Issue 1, 2018, pp. 42-50
https://doi.org/10.32010/26166127.2018.1.1.42.50

ISE: An Intelligent and Efficient Steganalysis Engine for Image Database in Big Data Systems

Mayank Tiwary[1], Pritish Mishra[1], Mohammad S. Obaidat[2], Deepak Puthal[3]

[1] SAP Lab, Bangalore, India, {mayank.tiwary, pritishmishra}@sap.com,
[2] Department of ECE, Nazarbayev University, Astana, Kazakhstan;] King Abdullah II School of Information Technology, The University of Jordan, Jordan, Ministry of Education Overseas Distinguished Professor at University of Science and Technology Beijing, China, msobaidat@gmail.com
[3] University of Technology Sydney, Australia, deepak.puthal@uts.edu.au

*Correspondence: Mohammad S. Obaidat, The University of Jordan, Jordan, Ministry of Education Overseas Distinguished Professor at University of Science and Technology Beijing, msobaidat@gmail.com

Abstract

The aim of this work is to design a faster and artificially intelligent steganalysis engine, which is able to secure the image databases from any infected image in big data environment. The proposed Intelligent Steganalysis Engine (ISE) for image database in big data makes use of three steps, which are image estimation, feature generation and classification. In the first step, five new images are estimated from the original image, for computing 438 features and then these data images are passed through a classifier for final prediction of a stego image. The engine is designed based on Map-Reduce programming approach to cope with big data. The actual experiments were performed on the Big Data Hadoop by taking standard image data set. In the first two steps, the images are processed in both spatial and DCT domain. During these steps the implementations of image estimation and feature extraction algorithms become very much computationally intensive and seek a huge amount of time. The results obtained are compared with previously reported six similar works and an inference has been drawn for appropriate use of feature set and classifier pair.

Keywords: Big Data, HPC, Steganalysis, Map, GPU, Hadoop, CUDA

1. Introduction and Background Information

Steganography is the art of hiding secret information in audio, video, images, text, etc. The current work deals with proposing algorithms(ISE) to detect stego images in the big data as shown in Figure 1. The proposed engine is based on Map and Reduce programming approach, where in Mapping phase, 438 features are generated and in Reduction phase the classification of the generated features is done to predict the given image as stego or non-stego. Graphics Processing Units (GPU) are specialized multi core processors, which are designed to solve image processing and video processing operations faster and provide a good parallel platform to researchers. In ISE, the computationally intensive operations are offloaded to GPU.

In [1], Lyu and Farid proposed a universal steganalyzer for gray scale images using first and high order wavelets. Harmsen et al. [2] in their work proposed a methodology to efficiently detect any steganographic method which adds noise in the spatial domain by using histogram characteristic function and center of mass for that histogram. Fridrich [3] in his work used 23 first and second order features and ANN for stego image prediction. In their work, they decompressed a given image cropped by 4 pixels and compressed it. This cropping image along with original image were used to generate features, which can classify F5, Outguess and Steghide. Shi et al. [4] generated a set of features based on the differences in the JPEG 2-D array using Markov's random process. In [4], the features were derived from the transaction probability matrix. In [5], the authors used multi domain features to detect steganography. In [5], features were collected from both DCT and spatial domain for efficient classification process. In [6], the authors classified Jsteg and F5 steganographic techniques using wavelet domain features. Westfeld [7] designed a detector for color images, which has its basis in the effect that the embedding algorithm has on the occurrences of close pairs of colors.

Mei-Ching Chen et al. [8], used alpha trimmed mean filter to generate a set of 25 features for steganalysis, which can help in the classification of F5, outguess and steghide steganographic techniques. Wenqiong Yu et al. [9] presented a total of 9 functions to generate a set of 168 features based on center of mass of histogram characteristic function, which helped in the prediction of the five steganographic techniques including MB1, MB2, Jphide, Outguess and Steghide. In [9], the authors generated features from both DCT and spatial domain. In [10], the authors

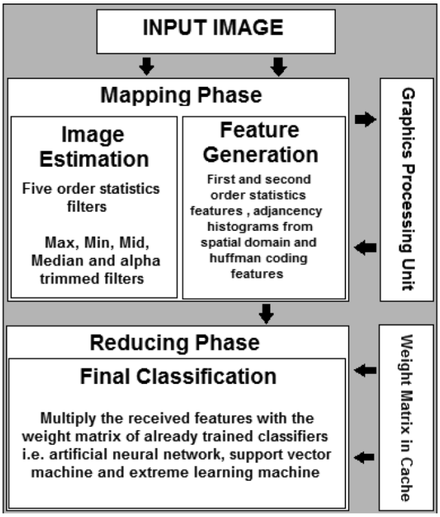

INPUT IMAGE
Mapping Phase
Image Estimation
Five order statistics filters
Max, Min, Mid, Median and alpha trimmed filters
Feature Generation
First and second order statistics features , adjancency histograms from spatial domain and huffman coding features
Graphics Processing Unit
Reducing Phase
Final Classification
Multiply the received features with the weight matrix of already trained classifiers i.e. artificial neural network, support vector machine and extreme learning machine
Weight Matrix in Cache

Figure 1. Proposed ISE Based on Map-Reduce Approach

used SVM and Neural Network Pattern Recognition Tool (NPR) for classification of stego and clean images. Zuzana Oplatkova et al. [11] used huffman encoding features for DCT compression, which showed a high variation between stego and clean images. The authors used ANN for classification, which resulted in a high classification accuracy for detecting Outguess and Steghide steganography techniques. Andrew D.ker [12] used the histogram characteristics function for calibrated image based on adjacency histogram instead of the usual histogram function which helps in the classification for LSB matching. In [14], the authors estimated MB steganography based on least square method generation. In this work, the authors tried to estimate the length of the embedded message. In [13], authors tried to increase the efficiency of classification based on first and second order features by introducing localized generalization error model. Xiaolong Li's [15] work is based on the steganalysis of LSB matching. In his work, he tried to increase the efficiency of detection by considering the ratio of the histograms DFT coefficient of a given image to the corresponding coefficient of the down sampled image. In [19], the authors generated 255 features from the DCT domain using 5 estimated images and classified the images as stego or non stego.

The key contributions of this work can be summarized as below.

- Applying five order statistics filters for estimating images, which generates 438 features using both spatial and transform domain for robustness.
- Offloading the computationally intensive operations of image estimation and feature generation phases to GPU's and structuring the whole process into Map and Reduce programming approach to deal with big data for faster execution.
- Detecting ten types of stegos including F5, Outguess, Steghide, MB1, MB2, LSB Matching, LSB Replacement, Jphide, Jsteg and PQ.
- Classifying the image to stego or non-stego using three machine learning classifiers, which are SVM, ANN and ELM to achieve a better accuracy.

2. Proposed Technique:

- Our proposed scheme consists of four key phases: image estimation, feature generation, reduction and classification phases. The mapping phase takes a single image as an input, where the key is the name of the image and value is the image matrix. The image matrix is further divided into three matrices each for Red, Green and Blue domain. These Red, Green and Blue matrices are sent to the GPU, where the GPU does the actual processing. In this phase, a set of four GPU kernels has been used which are called asynchronously to increase the overall performance. The four sets of GPU kernels are image estimator, first order statistics feature generator, second order statistics feature generator and spatial domain feature generator. The output of the image estimation kernel is fed asynchronously to the remaining three kernels. The communication to these 3 kernels is facilitated by the Multi-Process Service(MPS) [16] provided in Nvidia GPU cards. A description of the each major phase is given below.

2.1. Image Estimation:

- In this phase, the proposed engine is applied on five order statistics filters namely alpha trimmed mean, max, min, median and mid filter.

- Alpha Trimmed Mean Filter: This filter is used to discard the maximum and minimum values in the region Sxy of given pixels. It is used to remove combination of salt, pepper and Gaussian noise.
- Max and Min Filter: Max/Min filter is used to replace the value of a given pixel with the maximum/minimum of the gray level. Both the max and min filter are used to remove salt and pepper noises.
- Median Filter: It is a non-linear filter used to remove presence of both unipolar and bipolar noises. It replaces the value of a given pixel with the median of the gray level.
- Mid Filter: Mid filter is used to remove random filtered noise such as Gaussian or uniform noise. It replaces the value of a given pixel by calculating the mid-point between max and min filter.

2.2. Feature Generation

Feature generation phase is meant for extracting relevant features, which would be meaningful for data processing such as machine classification. In this phase, extraction of features takes place from original image and estimated images. A total set of 438 features are extracted from spatial and DCT domain using first order statistics features, second order statistics features, Huffman's coding features and features from spatial domain.

In first order and second order statistics, a set of histograms and co-occurrence matrices is used. We use histogram characteristic function (HCF) and center of mass (COM) for estimation of changes caused by embedding the data by different steganographic techniques in statistical model. COM is used to represent the energy distribution in HCF, whereas the HCF represents a histogram in frequency domain.

For each single feature in statistical models, the COM's are calculated for all estimated and original image and then the feature of estimated image is subtracted from the feature of the original image and is represented by the below equation:

$$\text{Feature} = |\text{COM}[\text{Original}] - \text{COM}[\text{estimated}]|$$

In first order and second order statistics, we use the below 6 functions:

- F1: This function is used to extract a feature using the histograms of all DCT coefficients. It results in one feature f1.
- F2: This function is used to represent the individual histograms of the first five AC coefficients. This results in a set of five basic features f2, f3, f4, f5 and f6.
- F3: This function is used to calculate total occurrences of all the AC coefficients from -5 to 5. This function results in 11 basic features from f7, … f17
- F4: This function is used to measure the variation of the neighboring blocks in DCT domain. Here only one basic feature f18 is extracted.
- F5: This function is used to measure the variation along the boundaries of DCT blocks. Here two basic features f19 and f20 are derived.
- F6: This feature is derived from the co-occurrence matrix. The function used to derive three basic features f21, f22 and f23.

Huffman's coding for compression features are also included in the feature set.

This includes features for 12 coefficients of the DCT matrix. The classes include the DC and AC coefficients for class 0 and class 1 that is DC class 0, AC class 0, DC class 1 and AC class 1. This totals to 48 features. Lastly, the features are collected from the spatial domain. In the spatial domain, a total of 45 features are extracted by using functions F7, F8 and F9.

- F7: This feature represents the histogram of the difference between the adjacent pixels of an image in spatial domain.
- F8: This feature calculates the histogram of the pixel difference at the boundary location of the DCT block.
- F9: This feature is derived from the co-occurrence matrix pixel difference of blocks in spatial domain.

There are a total of 45 features from the spatial domain using F7, F8 and F9. These features are the output of the mapping phase. The mapper outputs the key and value pairs to the reducer, where the key is the image name and value is an array containing these 438 features.

2.3. Reduction Phase

The main function of this phase is the classification. The reduction phase receives the input from the mapping phase;

it receives the image name as key and the value as its derived features. The classifiers in the reducer use the weight

matrix which is pre-determined based on training of the machine classifiers. If the image feature is classified as stego,

the reducer outputs the same key, and value as binary 0 or 1 to indicate stego or non-stego image. This output is finally written to the HDFS as final result.

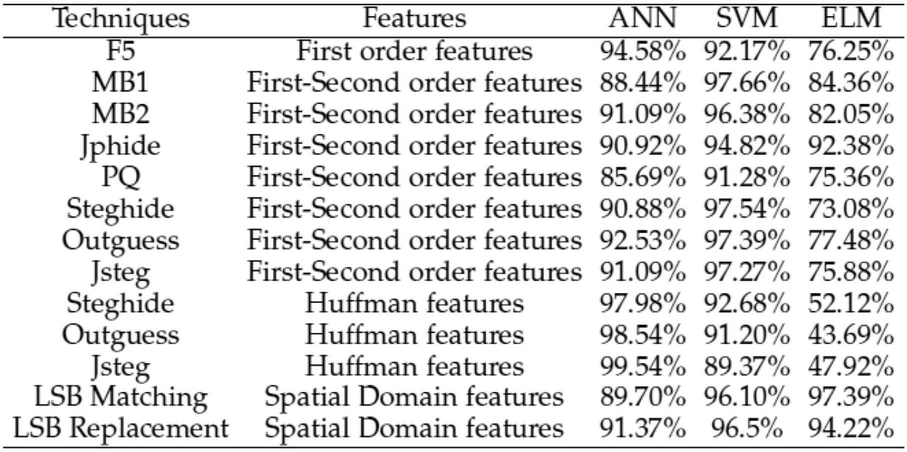
2.4. Classification Phase

For the classification process, three well known machine classifiers, which include ANN, SVM and ELM have been used. In the feature set, 4 subsets of features have been used, which are first order and second order statistics features, features from Huffman's encoding of DCT matrix, and features from spatial domain. These feature sub sets are used to detect different steganographic techniques. For each steganography technique, there can be one or more trained weight matrix obtained from the respective machine classifiers. These weight matrices are cached so that they can be accessed easily by every reducer which runs on different nodes in the Big Data cluster.

The ELM classifier do not generate a weight matrix and the training and testing process occur simultaneously. However, the SVM and ANN generate weight matrix. These weight matrices are cached in distributed environment, which are used further by reducer during actual simulations. A total of 21 weight matrices were generated after training process, which ultimately result in making the proposed model robust and accurate.

3. Performance Evaluation

The experiments were performed on hadoop 4 node cluster where, GPU was attached with each node in the cluster. The CPU used was Intel Core i5 and GPU was NVIDIA's GT540M. The operating system used was Ubuntu 12.04 LTS. For testing the algorithm, the Apache Hadoop was used. For doing image processing in Hadoop, Java's buffered image library was used and a custom data type in Hadoop was developed which could easily read images. For integrating the power of GPU with the Hadoop cluster, Java Native Interface enabled Java Compute Unified Device Architecture(JCUDA) was used. The Hadoop's distributed cache was used to cache the weight matrices which is used in the reducing phase for final classifications. For big image data set, the images were downloaded from Break Our Water Marking System(BOWS) [17]. This source provides a set of 10000 images in PGM format, which were converted to JPEG format using specific compression ratio. After conversion the total data size is approximately equal to 1.5 GB. To further increase the size of data, copying the same data again and again to get a size of 15GB has been done to perform the experiments for big data. In the classification phase, three classifiers which include ANN, SVM and ELM have been used. The ANN was simulated in MATLAB and trained. Then, the SVM was simulated using LibSVM [18] library in Java. The ELM was simulated in the reducer method itself using the panrose inverse algorithm. The neural network classifier used in this work is a supervised multilayer feed forward network, which uses a single hidden layer containing four nodes. Back propagation algorithm was used to train the network. The whole experiment is carried out using Matlab R2013a. The total data set is divided into two sets. A training set and a testing set; each comprising of almost 50% of the total data.



Techniques	Features	ANN	SVM	ELM
F5	First order features	94.58%	92.17%	76.25%
MB1	First-Second order features	88.44%	97.66%	84.36%
MB2	First-Second order features	91.09%	96.38%	82.05%
Jphide	First-Second order features	90.92%	94.82%	92.38%
PQ	First-Second order features	85.69%	91.28%	75.36%
Steghide	First-Second order features	90.88%	97.54%	73.08%
Outguess	First-Second order features	92.53%	97.39%	77.48%
Jsteg	First-Second order features	91.09%	97.27%	75.88%
Steghide	Huffman features	97.98%	92.68%	52.12%
Outguess	Huffman features	98.54%	91.20%	43.69%
Jsteg	Huffman features	99.54%	89.37%	47.92%
LSB Matching	Spatial Domain features	89.70%	96.10%	97.39%
LSB Replacement	Spatial Domain features	91.37%	96.5%	94.22%

Figure 2. Classification accuracy for different classifiers with different feature sets

After the training is over, the network was tested against the test dataset to validate the learning process. The SVM classifier used here uses a Radial basis Kernel function as it works well for classification task. The set of images used in the training process for each steganographic algorithm and for each classifier was

30000. To increase the efficiency of the classifier, each steganographic technique training was done separately and a separate weight matrix was generated. For training in ANN, the weight matrix was copied from the neural network simulator after training. But in case of SVM, the LibSVM provides an easy interface for generating the weight matrix. These generated weight matrices, would be cached in the Hadoop's Distributed Cache. For testing the accuracy of the classification, large set of images were tested which include image data of size 15GB. The images for testing were divided into two parts. The first part of images was stego, where 2.341 KB and 0.65 KB of text was embedded and the rest of the images were clean. These testing sets were generated for each of the ten steganographic methods. In case of ELM, some known features set were cached in the Hadoop Distributed cache and used in the classification process. These set of features used in the classification process includes features of known 30000 images. The features for ELM are limited and include first order features, second order features and features from spatial domain.

After extensive simulation of the proposed model, we achieved the results as shown in Figure 2. Figure 3 represents the CPU-GPU time comparison graph. Here CPU time represents the total time taken to process a given set of images with hadoop cluster without any GPU card, whereas the GPU time represents the total time taken to process a given set of images with hadoop cluster having GPU cards attached to each node.

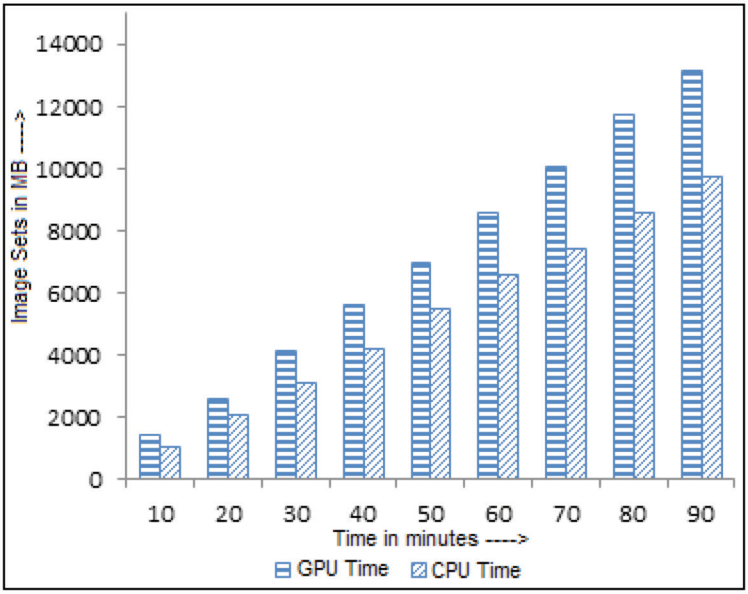


Figure 3. CPU-GPU Time Comparison Graph

4. Conclusion

In this paper, a Map-Reduce based implementation of steganalysis for JPEG images stored in big data environment, is introduced. The paper proposes algorithms that offload computationally intensive operations to GPU for faster operation. The proposed approach consists of 3 essential steps: image estimation, feature generation and classification. During the image estimation phase, five image estimation techniques have been introduced. Next, in the feature generation phase, a total of 438 features were extracted from spatial and DCT domains. Finally, in the classification phase, three classifiers were used for final classification. Introducing the usage of GPU in the Hadoop cluster shows a good improvement of the speedup over the CPU time, which is expected to increase with increase in data-size, that could go up to tera-bytes in big data; see Figure 3. The performance and efficiency of the proposed engine is validated by experimenting and comparing with ten steganographic techniques which include F5, Outguess, Steghide, MB1, MB2, LSB Matching, LSB Replacement, Jphide, Jsteg and Perturbed Quantization(PQ). In conclusion, the observations obtained from the proposed engine show a high degree of accuracy and offer an intelligent selection among the classifiers and its corresponding optimum set of features. The proposed steganalysis engine can also be applied to stego video data along with higher number of steganographic techniques.

References

[1] Lyu, S., Farid, H. (2002) Detecting Hidden Messages Using HigherOrder Statistics and Support Vector Machines, Proceedings of the 5th International Workshop on Information Hiding (pp.340-354). Noordwijkerhout: Springer.

[2] Harmsen, J.J., Pearlman, W. A. (2003) Steganalysis of Additive Noise Modelable Information Hiding, Proceedings of SPIE - The International Society for Optical Engineering, 5020, 131-142.

[3] Fridrich, J. (2004) Feature-based Steganalysis for JPEG Images and Its Implications for Future Design of Steganographic Schemes, Proceedings of the 6th International Information Hiding Workshop (pp.67-81). Toronto: Springer

[4] Shi, Y. Q., Chen, C. H., Chen, W. (2007) A Markov process based approach to effective attacking JPEG steganography, Proceedings of the 8th Information Hiding Workshop (pp.249-264). Alexandria: Springer

[5] Li, Z., Lu, K.J., Zeng, X.T., Pan, X.Z. (2009) Feature-Based Steganalysis for JPEG Images, Proceedings of the 1st International Conference on Digital Image Processing (pp.76-80).

[6] Zhang, Q., Liu, Y., Zhang, Sh., Chen, K. (2009) Classification Method of Jsteg Stego-Images and F5 Stego-Image, Proceedings of the Fourth International Conference on Innovative Computing, Information and Control (ICICIC) (pp.394-397). Kaohsiung, Taiwan.

[7] Westfeld, A. (2002) Detecting low embedding rates, Proceedings of the 5th International Workshop on Information Hiding (pp.324-339). Noordwijkerhout: Springer.

[8] Chen, M., Agaian, S.S., Chen, C.L.P., Rodriguez, B.M. (2009) Alpha-trimmed image estimation for JPEG steganography detection, Proceedings of IEEE International Conference on Systems, Man and Cybernetics (pp.4581-4585). San Antonio, USA.

[9] Yu, W, Li, Z., Ping, L. (2010) Blind detection for JPEG steganography, Proceedings

of the International Conference on Networking and Information Technology (pp.128-132). Manila, Philippines.

[10] Bansal, D., Chhikara, R. (2014) Performance Evaluation of Steganography Tools Using SVM and NPR Tool, Proceedings of the 4th International Conference on Advanced Computing and Communication Technologies (ACCT) (pp.483-487). Rohtak, India.

[11] Oplatkova, Z., Holoska, J., Zelinka, I., Senkerik, R. (2009) Detection of Steganography Inserted by OutGuess and Steghide by Means of Neural Networks, Proceedings of the 3rd Asia International Conference on Modelling and Simulation (pp.7-12).

[12] Ker, A.D. (2005) Steganalysis of LSB matching in grayscale images, IEEE Signal Processing Letters, 12(6), 441-444.

[13] Zhi-Min, H.; Ng, W.W.Y., Chan, P.P.K., Yeung, D.S. (2010) Steganography detection using localized generalization error model, Proceedings of the IEEE International Conference on Systems Man and Cybernetics (pp.1544-1549).

[14] Xu, M., Li, T., Ping, X. (2009) Estimation of MB steganography based on least square method, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, (pp.1509-1512).

[15] Li, X., Zeng, T., Yang, B. (2008) A further study on steganalysis of LSB matching by calibration, Proceedings of the 15th IEEE International Conference on Image Processing, (pp.2072 – 2075).

[16] Nvidia MPS. (2015, February 19). Retrieved from: https://docs.nvidia.com/deploy/pdf/CUDA Multi Process Service Overview.pdf

[17] Bas, P., Furon, T. (2015, February 19). Break our watermarking system, second edition. Retrieved from: http://bows2.gipsa-lab.inpg.fr/.

[18] Chang, C. C., Lin C. L. (2011) LIBSVM: A Library for Support Vector, ACM Transactions on Intelligent Systems and Technology, 2 (3), 1-27.

[19] Tiwary, M., Priyadarshini R., Misra, R. (2014) A faster and intelligent steganography detection using Graphics Processing Unit in cloud. Proceedings of the International Conference on High Performance Computing and Applications (pp.1-6). Bhubaneswar, India.

Submitted 05.02.2018
Accepted 08.05.2018