# Speech Steganalysis using Evolutionary Restricted Boltzmann Machines

Catherine Paulin
Université de Moncton
Campus de Shippagan
LARIHS Laboratory
New Brunswick, Canada
Email: ecp8266@umoncton.ca

Sid-Ahmed Selouani
Université de Moncton
Campus de Shippagan
LARIHS Laboratory
New Brunswick, Canada
Email: sid-ahmed.selouani@umoncton.ca

Éric Hervet
Université de Moncton
Campus de Moncton
Département d'informatique
New Brunswick, Canada
Email: eric.hervet@umoncton.ca

***Abstract*—This paper presents a new method to train Restricted Boltzmann Machines (RBM) using Evolutionary Algorithms (EA), where RBM are used in the first step of a steganalysis tool for audio files. The following EA have been tested: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Artificial Bees Colony (ABC), and Cat Swarm Optimization (CSO). Our method has been tested with three steganographic techniques: StegHide, Hide4PGP, and FreqSteg. A fourth technique combining the three steganographic methods has also been tested. The results are compared to the contrastive divergence learning algorithm. All EA outperform the contrastive divergence algorithm.**

***Index Terms*—DBN, RBN, EA, PSO, ABC, CAT, GA, Steganography, Steganalysis.**

## I. Introduction

Steganalysis is the art of detecting secret message hidden through steganography in data such as images, audio or video file, etc.

Steganography is use mainly for secret communications and water marking. In a two-way communication where steganography is used, a secret message is undetectable to a third party. A file containing a hidden message is called the carrier. In this litterature, a cover with an hidden message is tag as marked, and as unmarked if there is no message hidden within.

In [1], a combination of audio quality metrics and SVM is used to build the steganalyzer. It has been tested on four watermarking and two steganographic techniques. First, each method is tested individually with results ranging from 87% to 100%. Then the detection for the ensemble of the watermarking techniques is tested with a detection rate of 69%, and the ensemble of the steganographic techniques with a detection rate of 73%.

In [2], Mel-Frequency Cepstral Coefficients (MFCC) are introduced to extract significant features from the speech signals. Those coefficients are used as input to a SVM classifier with Radial Basis Function (RBF) kernels. This technique has been tested on a number of steganographic methods including StegHide [3].

Following the work of [2], Sung et al designed second-order derivative-based MFCC combined with SVM using RBF kernels for classification [4]. Steganographic techniques tested with this method include Hide4PGP, LSB, and StegHide. The results of this modified MFCC approach outperform those from [2].

In [5], parameters of Line Spectral Frequencies (LSF) code the audio signals and use them in an autoregressive time delay neural network for classification. These methods have been tested on three steganographic techniques including stegHide and Hide4PGP on the Noizeus database [6]. They have been tested with 50% and 100% of hiding capacity, and gave results varying from 52.45% to 82.73%, with the best result for 100% of hiding capacity.

In [7], a method using MFCC and Gaussian Mixture Models (GMM) to classify audio signals for transcoding steganography (transteg) is implemented. It first uses MFCC parameters for acoustic analysis, then uses GMM to create two models: one for unmarked signals, and one for marked signals. Then the probability of the MFCC parameters of each signal to belong in each model is calculated. The two resulting probabilities are compared in order to classify the signal, so that the probability of belonging to a model is the highest. This method has been tested on five speech corpus including Timit [8]. Various configurations of transteg were tested, with the highest detection probability at 94.6%, and the lowest detectability at 63.3%.

In [9], a combination of Reversed-Mel Cepstrum Based Audio Steganalysis and SVM is performed to propose a new steganalysis method. This method has been tested on two steganographic techniques, Hide4PGP and StegHide, and two watermarking methods. The classification rates vary between 91% and 99%.

The aim of this research is to improve steganalysis techniques using Deep Belief Network (DBN) introduced in [10], where an acoustic analysis is performed on speech steganography to input those signals in a classifier. The acoustic analysis used is Mel-frequency Cepstrum Coefficients (MFCC). This DBN-based steganalyzer has been compared to two other existing robust steganalysis methods based on Support Vector Machines (SVM) and Gaussian Mixture Models (GMM). It outperforms both existing methods in two of the four steganographic methods tested.

In this article, we are using Evolutionary Algorithms (EA) to train the Restricted Boltzmann Machines used to construct the DBN. Those results are compared to the RBM trained with the Contrastive Divergence algorithm.

The rest of this paper is organized as follows: Section II presents the RBM theory. Section III describes the EA used in this article with a brief state of the art for each. Section IV describes the steganographic techniques used to test our new steganalysis tool, while Section V proceeds with a description of the experiment methodology, and a discussion about the obtained results. Section VI concludes the paper.

## II. Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBM) is a neural network with two layers: One visible and one hidden, as showed in Figure 1.

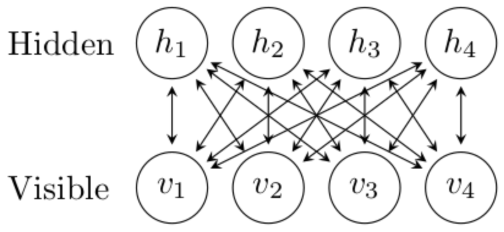


Fig. 1. Restricted Boltzmann Machines.

The energy function of this system is given by:

$$E(\mathbf{x},\mathbf{h}) = -\mathbf{b}^t\mathbf{x} - \mathbf{c}^t\mathbf{h} - \mathbf{h}^t\mathbf{W}\mathbf{x} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{h}$ represent respectively the units vector of the visible and hidden layers, $\mathbf{b}$ and $\mathbf{c}$ are the bias for those respective layers, and $\mathbf{b}^t$ is the transpose of $\mathbf{b}$. The weights between the two layers are represented by matrix $\mathbf{W}$. The probability distribution of this system is given by:

$$P(\mathbf{x},\mathbf{h}) = \frac{e^{-E(\mathbf{x},\mathbf{h})}}{Z} \tag{2}$$

where $Z$ is the partition function of the system. It gives the energies' sum for all possible configurations of the system. It is defined by:

$$Z(\mathbf{x},\mathbf{h}) = \sum_{\mathbf{x},\mathbf{h}} e^{-E(\mathbf{x},\mathbf{h})} \tag{3}$$

The conditional probability of the hidden layer knowing the visible layer is:

$$\begin{aligned} P(\mathbf{h}|\mathbf{x}) &= \frac{e^{-E(\mathbf{x},\mathbf{h})}}{\sum\limits_{h} e^{-E(\mathbf{x},\mathbf{h})}} \\ &= \frac{e^{\mathbf{b}^t\mathbf{x}+\mathbf{c}^t\mathbf{h}+\mathbf{h}^t\mathbf{W}\mathbf{x}}}{\sum\limits_{\mathbf{h}} e^{\mathbf{b}^t\mathbf{x}+\mathbf{c}^t\mathbf{h}+\mathbf{h}^t\mathbf{W}\mathbf{x}}} \\ &= \frac{e^{\mathbf{c}^t\mathbf{h}+\mathbf{h}^t\mathbf{W}\mathbf{x}}}{\sum\limits_{\mathbf{h}} e^{\mathbf{c}^t\mathbf{h}+\mathbf{h}^t\mathbf{W}\mathbf{x}}} \\ &= \frac{\prod\limits_{j} e^{\mathbf{c}_j\mathbf{h}_j+\mathbf{h}_j\mathbf{W}_j\mathbf{x}}}{\prod\limits_{j}\sum\limits_{\mathbf{H}} e^{\mathbf{c}_j\mathbf{h}_j+\mathbf{h}_j\mathbf{W}_j\mathbf{x}}} \\ &= \prod_{j} P(\mathbf{h}_j|\mathbf{V}) \end{aligned} \tag{4}$$

In the same manner, we find that:

$$P(\mathbf{x}|\mathbf{h}) = \prod_{i} P(\mathbf{x}_i|\mathbf{H}) \tag{5}$$

Therefore, the activation probability of a hidden neuron is given by:

$$\begin{aligned} P(\mathbf{h}_j = 1|\mathbf{x}) &= \frac{e^{\mathbf{c}_j\cdot 1+1\cdot\mathbf{W}_j\mathbf{x}}}{e^{\mathbf{c}_j\cdot 0+0\cdot\mathbf{W}_j\mathbf{x}} + e^{\mathbf{c}_j\cdot 1+1\cdot\mathbf{W}_j\mathbf{x}}} \\ &= sigm(\mathbf{c}_j + \mathbf{W}_j\mathbf{x}) \end{aligned} \tag{6}$$

where $sigm$ is the sigmoid function. The activation probability for a visible neuron is:

$$P(\mathbf{x}_j = 1|\mathbf{h}) = sigm(\mathbf{b}_i + \mathbf{W}_i\mathbf{h}) \tag{7}$$

As we train this model, we want it to generate data that resembles the training data. Therefore, we want to minimize the negative log probability of the training data:

$$\frac{\partial}{\partial\theta}(-logP(\mathbf{x})) = \frac{\partial}{\partial\theta}\left(-log\left(\sum_{\mathbf{h}} P(\mathbf{x},\mathbf{h})\right)\right) \tag{8}$$

By developing the second part of Eq. 8, we get :

$$\begin{aligned} \frac{\partial}{\partial\theta}(-logP(\mathbf{x})) &= \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x})\frac{\partial}{\partial\theta}E(\mathbf{x},\mathbf{h}) \\ &\quad - \sum_{\mathbf{x},\mathbf{h}} P(\mathbf{x},\mathbf{h})\frac{\partial}{\partial\theta}E(\mathbf{x},\mathbf{h}) \end{aligned} \tag{9}$$

Therefore, with our parameters, we get:

$$\begin{aligned} \frac{\partial}{\partial\mathbf{W}}(-logP(\mathbf{x})) &= \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x})(-\mathbf{h}^t\mathbf{x}) - \sum_{\mathbf{x},\mathbf{h}} P(\mathbf{x},\mathbf{h})(-\mathbf{h}^t\mathbf{x}) \\ \frac{\partial}{\partial\mathbf{b}}(-logP(\mathbf{x})) &= \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}(-\mathbf{x}) - \sum_{\mathbf{x},\mathbf{h}} P(\mathbf{x},\mathbf{h})(-\mathbf{x}) \\ \frac{\partial}{\partial\mathbf{c}}(-logP(\mathbf{x})) &= \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x})(-\mathbf{h}) - \sum_{\mathbf{x},\mathbf{h}} P(\mathbf{x},\mathbf{h})(-\mathbf{h}) \end{aligned} \tag{10}$$

The first part of those equations is easily calculated because it is the sum on the hidden layer only. The second part is not as trivial to compute because we need the sum on all possible states. In traditional RBM, the Contrastive Divergence algorithm (CD) is used to compute it [11]. It consists of using the training data as the first iteration of the visible layer$^{(1)}$, and then calculate the first iteration of the hidden layer$^{(1)}$ with the visible layer$^{(1)}$. We then compute the second iteration of the visible layer$^{(2)}$ with the hidden layer$^{(1)}$, and finally the second iteration of the hidden layer$^{(2)}$ with the visible layer$^{(2)}$. We use the visible layer$^{(1)}$ and hidden layer$^{(1)}$ for the first part of those equations, and the visible layer$^{(2)}$ and hidden layer$^{(2)}$ for the second part [12].

In this paper, EA are used to train RBM, where the Reconstruction Error (RE) is the parameter to optimize. This parameter is given by the following expression:

$$RE = \frac{1}{N} \sum_{N} (V_1 - V_2)^2 , \tag{11}$$

where $N$ is the number of training sample, $V_1$ is the visible layer$^{(1)}$ and $V_2$ is the visible layer$^{(2)}$ computed using the CD. The EA used are described in the next section.

## III. Evolutionary Algorithms

Evolutionary Algorithms are based on events that occur in nature. They are mainly used to solve optimization problems. In this paper, four algorithms are used: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Artificial Bees Colony (ABC) and Cat Swarm Optimization (CSO). They are used to optimize the reconstruction error of RBM by changing the weights between the two layers, that will be called the RBM Matrix. These algorithms are roughly explained in the next subsections.

### *A. Artificial Bees Colony*

An artificial bee colony is based on how a bee colony works in nature. It was first developped by Dervis Karaboga in [13]. The bees are divided in three groups: employees, onlookersa, and scouts. The number of employees, onlookers and scouts is each equal to the number of solutions to be optimized in the algorithm. Therefore, each employee represents a potential solution. One iteration of the algorithm is done by passing through three phases, one for each group. The pseudocode is described in Algorithm 1.

**Algorithm 1** ABC Algorithm

```
1: Randomly generate the first generation of employees
2: Calculate the fitness of each employee
3: while Number of iterations desired is not reached do
4:     Employees phase
5:     Onlookers phase
6:     Scouts phase
7:     Best solution saved
8: end while
```

In the employees phase, each employee explores a new neighbor, meaning changing the weights in the RBM Matrix and calculating the new fitness for this solution. If the new fitness is higher than the old one, this employee is changed to the new neighbor.

Afterwards, each onlooker chooses an employee with the standard roulette wheel selection. They then produce a new solution and evaluate it. If the new fitness is better than the old once, they replace the employee with the new solution.

In the scouts phase, if an employee bee hasn't been changed in a predetermined number of iterations, the employee bee is replaced by a new employee bee.

The specific details of the algorithm used in this paper can be found in [14]. Here are some articles where ABC are used to trained Neural Networks [15] [16].

### *B. Cat Swarm Optimization*

The Cat Swarm Optimization is based on the behavior of cats and was first proposed by Shu-Chuan Chu and al. in [17]. The two main behaviors of cats are used to make the optimization algorithm: The seeking mode where they observe the environment, and the tracking mode.

The population is formed of cats and every individual is a possible solution to the optimization problem. Every individual has a velocity vector and a flag indicating the phase of the cat at each iteration. The pseudocode is implemented in Algorithm 2.

**Algorithm 2** CSO Algorithm

```
1: Randomly create the cats population and their velocity.
2: Calculate the fitness of every cat.
3: Randomly update the flags of every cat.
4: while For each cat do
5:     Apply selected phase
6:     Randomly update the flags
7: end while
```

The seeking mode can be described as follows:

1) Make a copy of the position of the cat.
2) For each copy, a number of parameters are changed by adding or removing a certain value.
3) Calculate the fitness of each new copy.
4) Calculate the selecting probability.
5) Randomly pick a new position from the modified copy and change the cat's position for that copy.

The tracking mode can be described as follows:

1) Update the velocities for every dimension of the cat.
2) Verify if the velocities are in range of the maximum velocity, if not they are set to equal the limit.
3) Update the position of the cat with the velocities and calculate the new fitness.

The specific details of the algorithm used in this paper can be found in [17]. Subsequent work on CSO can be found in [18].

### *C. Genetic Algorithms*

Genetic algorithms have been around since the 60s. They were introduced by John Holland [19]. It consists in creating a population of individuals and moving through generations until the problem is optimized. A population is composed of individuals, which in turn are composed of chromosomes, which in turn are formed of genes. At each generation, a function mimics the behavior of natural selection.

One common application of Genetic Algorithms is function optimization. For the purpose of this article, a gene is represented as a weight vector, a chromosome as an RBM Matrix, and an individual is composed of a single chromosome. Every chromosome is a potential solution. It can be implemented in a few simple steps shown in Algorithm 3.

**Algorithm 3** Genetic Algorithm

```
1: Randomly generate the first generation
2: Calculate the fitness of every individual
3: while Number of generations desired is not reached do
4:     Mortality
5:     Selection
6:     Fitness
7:     Mortality
8:     Mutation
9: end while
```

The Mortality function randomly eliminates individuals amongst the ones with the lowest fitness, according to standard Roulette wheel selection. The Mortality function has a constant elimination rate.

The Selection function divides the rest of the population into groups of two, called parents, to create new individuals. The first child is created by taking the first $x$ parameters of the first parent, and the rest from the second parent. The second child is inversely created. Then, both children and parents are added to a new population.

The Mutation function is used to generate new genes on chromosomes according to a small mutation rate. It changes a random gene on a chromosome to a new value.

An overview of GA for training Neural Networks can be found in [20]. A list of papers where GA are used to trained Neural Networks can be found in [14].

### *D. Particle Swarm Optimization*

Particle Swarm Optimization is based on the behavior and interaction of bird flocking and fish schooling. Using these concepts for problem optimization was introduced by James Kennedy and Russel Eberhart in [21], [22]. A population is formed of birds, where each bird is a possible solution. In the algorithm, the birds move to different places by changing their position, which is reflected in the RBM Matrix, depending on their velocity. The algorithm is described in Algorithm 4.

The specific details of the algorithm used in this paper can be found in [23]. A list of papers where PSO is used to train Neural Networks can be found in [14].

**Algorithm 4** PSO Algorithm

```
1: Randomly generate the first generation of birds with
   position and velocity
2: Calculate the fitness
3: while Number of generations desired is not reached do
4:     Best solution saved
5:     Update the velocity of every bird
6:     Update the position of every bird
7:     Calculate the fitness
8: end while
```

## IV. Steganographic techniques

Three techniques of message embedding are tested in this paper, then a forth is made by combining them together. These techniques are presented in the next subsections.

### *A. StegHide*

StegHide has been widely used in steganalysis and is available as a free software [3]. StegHide hides data in a cover signal by altering its Least Significant Bit (LSB) [24]. It can be used on JPEG, BMP, WAV and AU files. In this paper, it is used to hide a signal in WAV files at its full capacity.

### *B. Hide4PGP*

Hide4PGP is another free software that has been widely used in steganalysis [25]. It can embed large messages in WAV and BMP by spreading the data evenly through the file. In WAV files, it can hide 4 bits per sample. In this paper, it is also used to hide signal in WAV files at its full capacity.

### *C. FreqSteg*

The third steganographic technique hides secret information in the high frequencies of the carrier signal. We will refer to it as FreqSteg in the rest of the paper. FreqSteg relies on the fact that a human ear cannot hear sounds beyond a frequency of 18 KHz. For signals with a sample rate of 44.1 KHz, the highest frequency recorded is 22.05 KHz, therefore the secret message is hidden between frequencies 18 KHz and 22.05 KHz.

Because the sampling rate of the databases we use for tests is different from 22.05 Khz, the hiding process percentage has been modified following these steps:

1) Pass the carrier signal through a Low-Pass Filter with a cut-off frequency of 70% of the original frequency.
2) Pass the secret signal through a Band-Pass Filter that keeps the frequency over 0.02% of the original frequency.
3) Generate a cosine signal at a frequency of 93.75% of the carrier signal frequency.
4) Multiply the secret signal and modulate the signals together.
5) Sum the resulting signal from the previous step with the resulting carrier signal.

This technique is detailed in [26] and is illustrated in Figure 2.

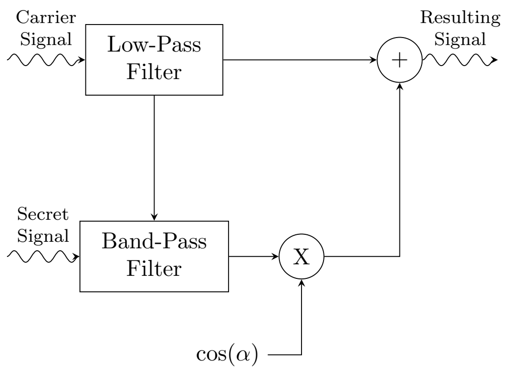


Fig. 2. Overview of the FreqSteg steganographic technique.

### D. Combined Steganographic Method

A fourth technique is created by combining the three previous steganographic techniques together to get a four-way classification. The classifiers are trained to detect whether or not some data was embedded, and if so, which steganographic technique the data was embedded with. This classification scheme is illustrated in Figure 3.

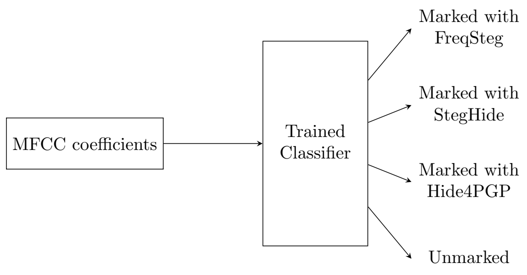


Fig. 3. Four-way classification.

## V. Experiments and Results

### A. Data

The experiments have been carried out on the Noizeus database. It is an English corpus of six speakers. Thirty sentences have been recorded originally sampled at 25 KHz and downsampled at 8 KHz [6]. The sentences last from 2 to 3.5 minutes. After embedding, an acoustic analysis is perfomed and each audio file is divided into frames. Each frame is then labeled marked or unmarked. All frames are then mixed randomly together, then 2/3 of them are used in the training process, and the rest as testing data.

### B. Experimental Methodology

In this experiment, the four EA are used to train the RBM. The four steganographic techniques are used to embed data in 50% of the Noizeus database in four different training sets. They are hidden at the full capacity of the cover files. Afterwards, the Mel-Frequency Cepstrum is used to extract features from the resulting speech signals. The signals are framed into 16 ms and the hamming window is also applied. Those frames are used as the training set. Based on the results found in [10], only MFCC coefficients ranging from 20 to 25 were tested.

For each EA, the training set is used as the visible layer for the RBM. The EA optimized the RE by changing the weights between the visible and the hidden layers.

### C. Experimental Results

Each EA was initialized with a population of 100, the hidden layer containing 10 neurons, and a batch size of 10 turned for 1000 iterations The mutation rate and the mortality rate for the GA have been set respectively to 0.075 and 0.2. The limit variable for ABC has been set to 10.

The results can be found in Figure 4. We observe clearly that the Reconstruction Error increases with the number of MFCC coefficients. We also notice that for every MFCC coefficient, the order of performance of the EA stays the same:

1) PSO
2) ABC
3) GA
4) CAT

Therefore, for each MFCC coefficient, the best results are obtained with the PSO algorithm. These results are summarized in Table I.

| MFCCs | FreqSteg | StegHide | Hide4PGP | Mixte |
|---|---|---|---|---|
| 20 | 3.91569 | 4.00847 | 3.88814 | 4.05334 |
| 21 | 4.22154 | 4.23103 | 4.0803 | 4.31356 |
| 22 | 4.49993 | 4.41834 | 4.39834 | 4.39629 |
| 23 | 4.77005 | 4.78249 | 4.64344 | 4.75001 |
| 24 | 4.94409 | 4.92138 | 5.00506 | 4.84554 |
| 25 | 5.03427 | 5.16952 | 5.06339 | 5.1478 |

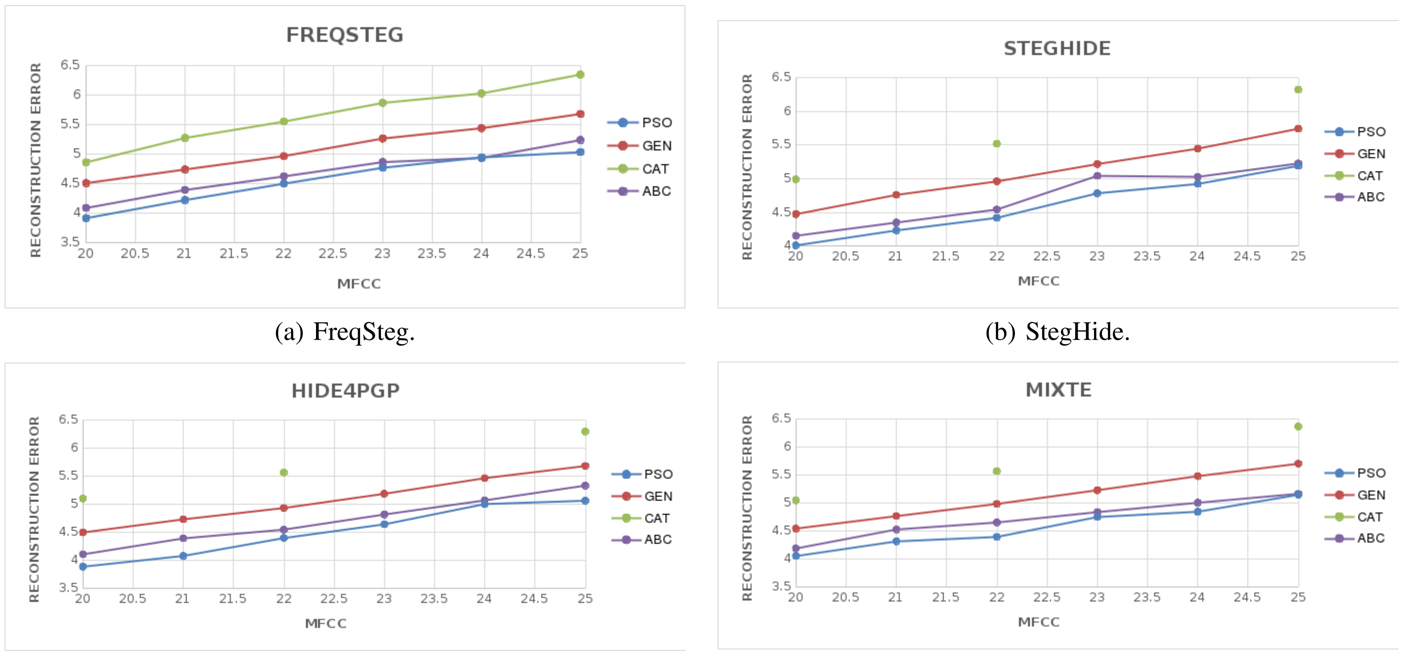
TABLE I
Best RBM results for each steganographic technique and MFCC coefficients tested trained with EA.

These results have then been compared with the same RBM trained with the Contrastive Divergence algorithm. The hidden layer is also composed of 10 neurons and the batch size set to 10. The momentum for the training process ranges from 0.1 to 1, the alpha varies in 0.00001, 0.0001, 0.001, 0.01 and 0.1. The number of epoch ranges from 100 to 1000. The best results are summarized in Table II.

| MFCCs | FreqSteg | StegHide | Hide4PGP | Mixte |
|---|---|---|---|---|
| 20 | 5.3524 | 5.3843 | 5.3843 | 5.3766 |
| 21 | 5.6379 | 5.9047 | 5.9244 | 5.6434 |
| 22 | 5.8988 | 5.9047 | 5.9244 | 5.9122 |
| 23 | 6.1688 | 6.1707 | 6.1916 | 6.4527 |
| 24 | 6.4401 | 6.4295 | 6.4527 | 6.4452 |
| 25 | 6.7017 | 6.7048 | 6.7159 | 6.7093 |

TABLE II
Best RBM results for each steganographic techniques and MFCC coefficients tested trained with CD.

These results show that every EA used in our experiments outperforms the Contrastive Divergence algorithm.



(a) FreqSteg. (b) StegHide.

(c) Hide4PGP. (d) Four-way classification.

Fig. 4. Classification results for each steganographic technique and each EA on the Noizeus database.

## VI. Conclusion

In this article, we used Evolutionary Algorithms to train unsupervised Restricted Boltzmann Machines. The EA used are: Particle Swarm Optimization, Genetic Algorithm, Artificial Bees Colony, and Cat Swarm Optimization.

Every algorithm has been used to train an RBM with MFCC coefficients ranging from 20 to 25 from steganographic audio files. The results obtained were compared to the Contrastive Divergence Algorithm.

It shows that EA outperform the CD where the PSO algorithm performs best. In the next steps of this research, the weights of the RBM trained with EA will be used as the initials weights of a Recurrent Neural Network.

## References

[1] H. Ozer, I. Avcibas, B. Sankur, and N. D. Memon, "Steganalysis of audio based on audio quality metrics," in *Electronic Imaging 2003*. International Society for Optics and Photonics, 2003, pp. 55–66.

[2] C. Kraetzer and J. Dittmann, "Mel-cepstrum-based steganalysis for voip steganography," in *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007.

[3] S. Hetzl, *StegHide Steganography*, 2003, http://steghide.sourceforge.net/.

[4] Q. Liu, A. H. Sung, and M. Qiao, "Temporal derivative-based spectrum and mel-cepstrum audio steganalysis," *Information Forensics and Security, IEEE Transactions on*, vol. 4, no. 3, pp. 359–368, 2009.

[5] S. Rekik, S.-A. Selouani, D. Guerchi, and H. Hamam, "An autoregressive time delay neural network for speech steganalysis," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*. IEEE, 2012, pp. 54–58.

[6] Y. Hu and P. C. Loizou, "Subjective comparison and evaluation of speech enhancement algorithms," *Speech communication*, vol. 49, no. 7, pp. 588–601, 2007.

[7] A. Janicki, W. Mazurczyk, and K. Szczypiorski, "Steganalysis of transcoding steganography," *annals of telecommunications-annales des télécommunications*, vol. 69, no. 7-8, pp. 449–460, 2014.

[8] J. S. Garofolo et al., "TIMIT: Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download," *Philadelphia: Linguistic Data Consortium*, 1993.

[9] H. Ghasemzadeh and M. K. Arjmandi, "Reversed-mel cepstrum based audio steganalysis," in *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*. IEEE, 2014, pp. 679–684.

[10] Catherine Paulin, Sid-Ahmed Selouani, and Éric Hervet, "Audio steganalysis using deep belief networks," *International Journal of Speech Technology, Under Review*.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[12] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Master's thesis, Technical University of Denmark*, 2012.

[13] Dervis Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep., Technical

report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[14] Dervis Karaboga, Bahriye Akay, and Celal Ozturk, “Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks,” in *Modeling decisions for artificial intelligence*, pp. 318–329. Springer, 2007.

[15] John A Bullinaria and Khulood AlYahya, “Artificial bee colony training of neural networks,” in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, pp. 191–201. Springer, 2014.

[16] John A Bullinaria and Khulood AlYahya, “Artificial bee colony training of neural networks: comparison with back-propagation,” *Memetic Computing*, vol. 6, no. 3, pp. 171–182, 2014.

[17] Shu-Chuan Chu, Pei-Wei Tsai, and Jeng-Shyang Pan, “Cat swarm optimization,” in *PRICAI 2006: Trends in Artificial Intelligence*, pp. 854–858. Springer, 2006.

[18] Shu-Chuan Chu and Pei-Wei Tsai, “Computational intelligence based on the behavior of cats,” *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 1, pp. 163–173, 2007.

[19] Melanie Mitchell, “Genetic algorithms: An overview,” *Complexity*, vol. 1, no. 1, pp. 31–39, 1995.

[20] Jürgen Branke, “Evolutionary algorithms for neural network design and training,” in *In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications*. Citeseer, 1995.

[21] James Kennedy and Russell Eberhart, “Particle swarm optimization,” in *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[22] Russ C Eberhart and James Kennedy, “A new optimizer using particle swarm theory,” in *Proceedings of the sixth international symposium on micro machine and human science*. New York, NY, 1995, vol. 1, pp. 39–43.

[23] Jun Sun, Wei Fang, Vasile Palade, Xiaojun Wu, and Wenbo Xu, “Quantum-behaved particle swarm optimization with gaussian distributed local attractor point,” *Applied Mathematics and Computation*, vol. 218, no. 7, pp. 3763–3775, 2011.

[24] D. Artz, “Digital steganography: hiding data within data,” *internet computing, IEEE*, vol. 5, no. 3, pp. 75–80, 2001.

[25] H. Repp, *Hide4PGP Steganography*, 1996, http://www.heinz-repp.onlinehome.de/Hide4PGP.htm.

[26] E. Swanson, CJ Ganier, R. Holman, and J. Rosser, *Frequency Domain Steganography*, https://www.clear.rice.edu/elec301/Projects01/smokey_steg/group.html.