INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X

Impact factor: 4.295

(Volume 4, Issue 2)

Available online at: www.ijariit.com

Analysis of Different Classifier for the Detection of Double Compressed AMR Audio

Fathima Najiya P
najinasi1102@gmail.com
Cochin College of Engineering and Technology, Valanchery, Kerala

Vipin Kishnan. C. V
vipinkrishnanm@gmail.com
Cochin College of Engineering and Technology, Valanchery, Kerala

ABSTRACT

A digital audio can be easily recorded by handheld devices such as digital voice recorders and Smartphone. A nd these audio are using as evidence in courts in many cases. One of the most important problems is that many audios often contains content forgery. Here we analyze the authenticity of AMR audio. An AMR is an audio codec for speech compression, this format is widely used in today's handheld devices such as a digital audio recorder or in Smartphone etc. Designing hand-crafted features is a challenging and time-consuming problem. In this paper, instead of manually extracting the features, we investigate how to use deep learning techniques in this audio forensics problem.. For an audio clip with many frames, the features of all the frames are aggregated and classified by the classifier. Here we use three classifier and compare them. Instead of hand-crafted features, we used the SAE to learn the optimal features automatically from the audio waveforms. Audio frames is the input to feature extractor and the last hidden layer's output constitutes the features of a single frame. At last the features of all the frames are aggregated and classified by either UBM-GMM or SVM or Bayesian classifier . When comparing and analyzing these three classifier the SVM and Bayesian classifier shows high degree of accuracy for detecting the authenticity of an audio .

Keywords: *Adaptive Multi-Rate, Double Compressed Audio, Stacked Autoencoder, UBM-GMM, SVM Classifier, Bayesian Classifier.*

1. INTRODUCTION

Digital speech can be easily recorded by handheld devices such as digital voice recorders and smart phones. As a result, more and more digital speech recordings appear as evidence in courts, creating the need for relevant forensic techniques. One of the most important issues is the detection of double compression. Since it often follows content forgery. The AMR is an audio compression codec is optimized for speech coding. A typical forgery of AMR audio involves decompressing the AMR audio into a waveform, modifying the signal by audio editing software, and re-compressing it into the AMR format. Thus, the tampered audio is compressed twice and is commonly referred to as double compressed AMR audio. The authenticity of double compressed audio is questionable because it may have been previously altered. We used a deep network to detect double compressed AMR audio [1]. In the previous work, we exploited a neural network with universal background model - Gaussian mixture model (UBM-GMM) as a classifier to detect audio frames, followed by majority voting which boosts the performance. This approach achieved a classification accuracy of 94%. This paper, as an extension of, investigates how to use a deep network for automatic feature extraction. We propose a new framework based on stacked auto encoder (SAE) network for feature extraction and an SVM and Bayesian for classification. Audio frames are individually used as the network input, and the last hidden layer's output constitutes their features. The features of all the frames in the audio clip are then aggregated and used as the input of SVM.

In this paper, we address the problem of detecting double compressed AMR audio .In recent years, researchers have addressed many audio forensic issues such as tampering detection [4],[5] , recorder identification , speaker identification [3] and verification , and audio compression history analysis . Compression history analysis is helpful in digital audio forensics and has drawn the attention of researchers in the past decade. The main idea behind identifying an audio compression history is to analyze the introduced quantization artifacts proposed a method to expose fake-quality MP3 audio clips re-compressed with higher bit-rates (up- transcoding)[12] . They studied the MDCT (modified discrete cosine transform) coefficients in MP3 audio and found out that the number of MDCT coefficients with small values in fake- quality MP3 is lower than normal MP3. Based on double quantization artifacts, some researchers used statistical features of MDCT coefficients to detect MP3 re-compressed clips for both up-transcoding and down-transcoding.

2. DOUBLE COMPRESSED AMR AUDIO

In this section, we briefly introduce the Adaptive Multi-Rate (AMR) standard, originally developed by 3GPP (the 3rd Generation Partnership Project). The codec [2] is optimized for speech signals and encodes narrowband (200–3,400 Hz) signals, with sampling frequency of 8,000 Hz. There are eight compression bit-rates including 4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2 and 12.2 kbps (kbit/s). The codec is based on the code excited linear prediction (CELP) model. Audio frames of 20 ms are analyzed by extracting the parameters of the CELP model. When a recording is compressed into AMR for the first time, it is single compressed when unless someone has recompressed it for second time, it is double compressed. When a recording is compressed into AMR audio for the first time, it is a single compressed audio, which is also an original one. Normally, AMR audio is not double compressed, unless someone has re-compressed it for a second time, e.g., to conceal the traces of a forgery deliberately. Double com- pressed AMR audio implies that the recording is essentially not original, which makes the detection problem important in forensic applications.

3. PROPOSED DETECTION FRAMEWORK BASED ON SAE AND CLASSIFIER

The detection of double compressed AMR audio [7] can be regarded as a pattern recognition problem. The traditional framework involves a feature extraction stage and a classification stage. Manual feature extraction relies on known quantization artifacts that discriminate between single and double compressed audio. Generally speaking, compression disturbs various statistics such as band energies and spectrum coefficients, which can be detected by well-designed hand- crafted features. However, this process is challenging and time- consuming. In recent years, much effort has been made to allow for automatic learning of features from the data. Here, we adopt a deep learning architecture: the stacked auto encoder. Our framework consists of two stages, as shown in Fig. 3.1. The first stage is feature extraction using an SAE network. The network's input is the normalized waveform samples of an audio frame, and the output of the last hidden layer in the network will be regarded as the feature vector extracted from a given frame. Single and double compressed AMR audio frames are both used to train the SAE network. In the second stage of the framework, UBM-GMM is adopted for classification in the previous stage. The input of classifier are feature vectors extracted from successive frames from an audio clip. Both single and double compressed AMR clips are used to train UBM-GMM.

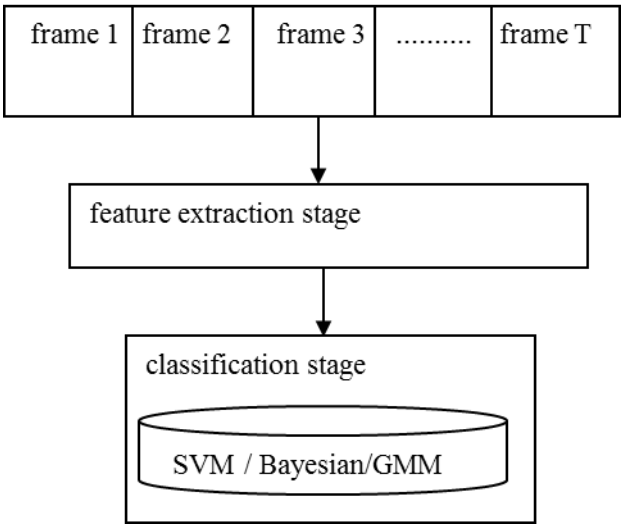


Fig.3.1 Framework of
Feature Extraction using Stacked Autoencoder

We employ the stacked auto encoder (SAE) to extract features from the data.. A typical autoencoder is a neural network containing three fully connected layers: an input layer, a hidden layer, and an output layer shown in fig 3.2 . The input of the autoencoder, vector ***xin*** = [*x*1*; x*2.....*xt*]′, will be mapped into hidden layer vector ***xhidden***, and the resulting vector is then mapped to a "reconstructed" vector, ***xout*** = [*x*^1*; x*^2*; :::; x*^*t*] by the following equations:

$$Xhidden = f(Xin) = A(W1Xin + b1)$$
$$Xout- g(Xhidden) = A(W2X\ hidden + b2).....3.1$$

Input Hidden Output

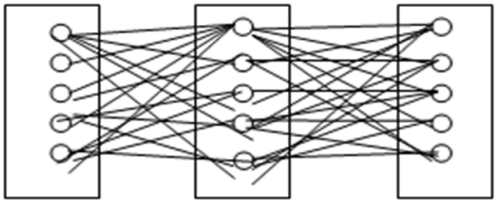


Fig 3.2 Autoencoder

Where *W*1 and *W*2 are weight matrices; ***b***1 and ***b***2 are bias vectors. We used the optimal *tanh* activation function:

$$A(t) = 1.7159 \times \tanh(2/3\, t) \quad \text{.....3.2}$$

Back-propagation is used for training, with the target values *x^1; x^2; :::; x^t* set to be the same as the inputs *x1; x2; :::; xt*. The autoencoder is forced to learn two mappings (*f* and *g*) and the combination of these two mappings aims to learn the data itself. In this procedure, the output of the hidden layer is another representation of the data in another domain, and thus they can be viewed as its features. The stacked autoencoder, referred to as the SAE network in this paper, is a combination of several autoencoders. The output layer is discarded after training, and the output of the hidden layer becomes the input to another autoencoder. We expect that the SAE network is able to learn features from the data layer by layer.

The training procedure of an SAE network includes two steps. In the first step, SAE pre-training creates (layer by layer) the aforementioned network. The following fine-tuning step uses an additional classification layer connected to the output of the last SAE layer. The classification layer corresponds to a classification result as either single or double compressed audio (two decision nodes). Such fine-tuning significantly improves the detection results. In our framework, this is also performed by back-propagation. After pre-training and fine-tuning, we discard the classification layer and use the outputs of the last hidden layer as our features. We use the normalized waveform of the audio signal as the input of the SAE network. Both single and double compressed AMR audio clips are first decompressed to waveforms, then normalized, and split into frames. When normalization is performed, all the single compressed AMR audio clips are regarded as a whole set, and each sample subtracts the mean and then divides by the standard deviation of the whole set. The same procedure is performed for double compressed AMR audio clips.

3.1 Classification Based on UBM-GMM

In order to obtain the final decision for the full au- dio clip, we need to analyze all of its frames. We adopt UBM-GMM (universal background model - Gaussian mix- ture model), which is a conventional framework for speaker recognition. A typical speaker recognition system using UBM-GMM includes two components: a front-end and a back-end. The front-end transforms audio waveforms into another representation, called acoustic features (such as Cepstral coefficients). In the back-end, speakers are modeled (enrolled) after a universal background model (UBM) is created. The UBM in the form of a Gaussian mixture model (GMM) represents the speaker-independent distribution of features. The speaker- specific models are then adapted from the UBM by maximum a posteriori (MAP) estimation. During speaker identification, the test audio is scored against all enrolled speaker models to identify the speaker.

To address our forensic issue, features of successive frames are first extracted from the audio clip via the SAE network and then are used as the input of UBM-GMM. For a speech segment (i.e., an audio clip), we obtain a set of features from successive frames. Let $\boldsymbol{X} = \boldsymbol{x}_1, \ldots \boldsymbol{x}_T$ *denote a* set of features from a speech segment, where $\boldsymbol{x}_t$ is a feature vector (column vector) of a frame at time t [1, 2, ..., T]. For a D- dimensional feature vector $\boldsymbol{x}_t$, the Gaussian mixture density used for the likelihood function to the model λ is defined as:

$$p(\mathrm{xt}/\lambda) = \Sigma\, \mathrm{wipi}(\mathrm{xt}) \quad \text{.....3.1.1}$$

Where M is the number of Gaussian models, and w_i is the weight for the ith model. The log-likelihood of the model *l for* feature vectors $\boldsymbol{X} = \{x1, x2 \ldots xt\}$ is defined as:

$$\log p(X/\lambda) = \Sigma \log p(Xt/\lambda) \quad \text{.....3.1.2.}$$

As shown in Fig. 3.1.1, the features of single and double compressed AMR audio serve as input for UBM training. We used the *adaptation* technique to obtain two separate models for single and double compressed audio. The final decision relies on the log-likelihood ratio between the two models. In essence, UBM is a large GMM trained to represent the overall distribution. We denote this model as λ_{ubm} and use expectation maximization (EM) for training. In the next step, features from single and double AMR audio are used to adapt the model λ_{ubm} by maximum a posteriori (MAP) estimation to create individual models λ_{sgl} and λ_{dbl}.The training of UBM- GM is described in detail in [3].

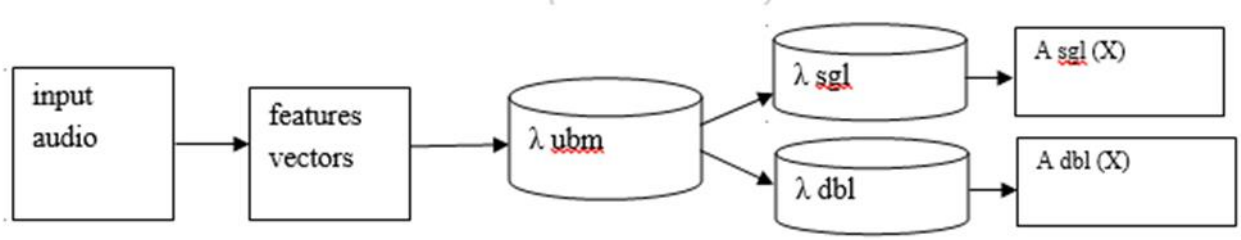
For a test speech $\boldsymbol{X}$, the log-likelihood ratios for both models λ_{sgl} and λ_{dbl}. are given by the following equations. When A $sgl(\boldsymbol{X})$ > A $dbl(\boldsymbol{X})$, the speech is identified as single AMR audio, otherwise as double AMR audio.

$$\text{A sgl (X)} = \log \frac{p(X/\boldsymbol{\lambda\ sgl})}{p(X/\boldsymbol{\lambda\ ubm})} = p(X/\boldsymbol{\lambda\ sgl}) - \boldsymbol{log}\ p(X/\boldsymbol{\lambda\ ubm})$$

……..3.1.3

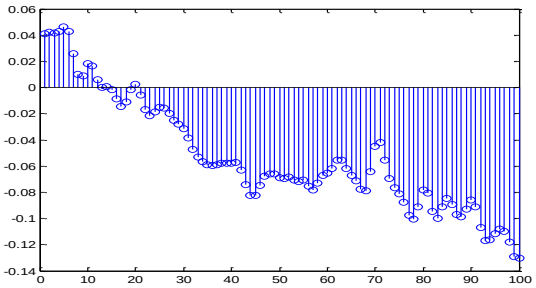
3.2 Visualization of Features Extracted by SAE

In other fields such as face recognition, some meaningful features can be learnt by the deep network, e.g., edges or face-like shapes. In audio applications, it is not trivial to describe explicit meaning to the obtained features. However, we observed that they exhibit certain interesting statistics and can be efficiently discriminative between the two considered audio classes. In Fig. 3.3, we visualize a 2-dimensional projection of the obtained features (output of the second hidden layer) generated using t-SNE. The input of t- SNE are the features from both single and double compressed frames, and their dimensionality is reduced by the t-SNE tool in an unsupervised manner. For reference, we also show an analogous projection of the input layer (top row).

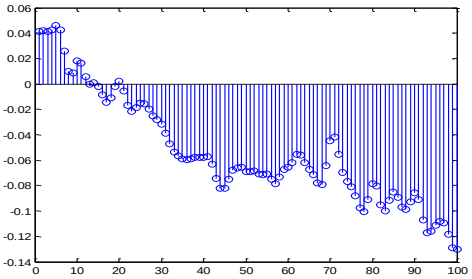


Bright (dark) colors indicate high (low) density. We can see that the distribution of the input layer's values for both single/double compressed AMR audio is similar. However, the distribution of the second hidden layer's output values moves toward a different direction.

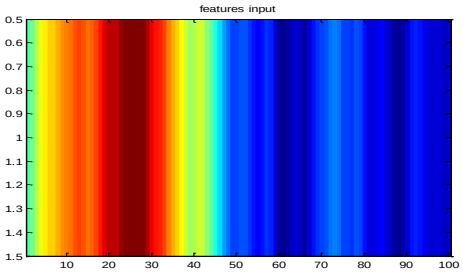
Note that t-SNE uses an unsupervised learning algorithm. Therefore, we have reasons to believe that the network learns useful information for their discrimination.



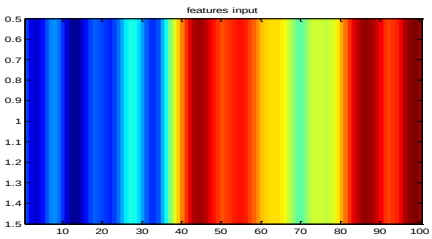
The input layer of single AMR



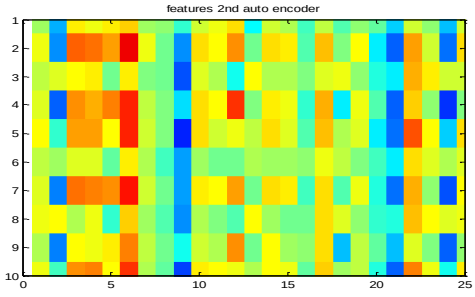
The input layer of double AMR



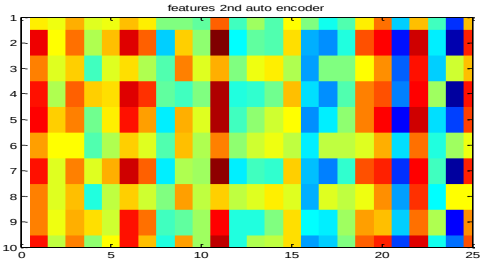
Features the first autoencoder of single AMR



Features the first autoencoder of double AMR



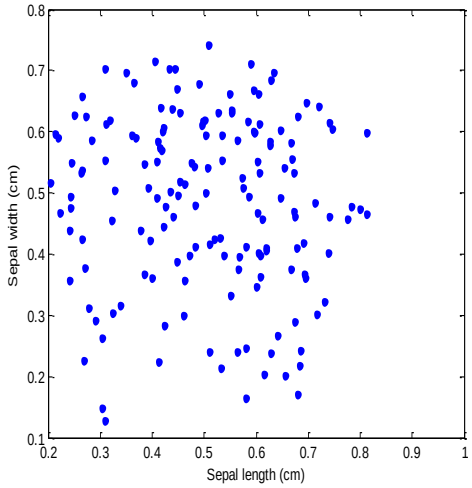
The hidden layer output of single AMR



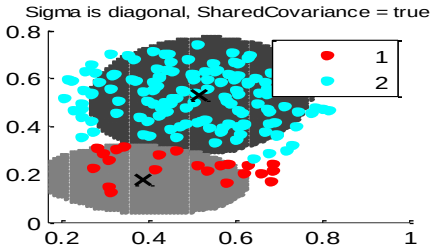
The hidden layer output of double AMR

Fig. 3.2.1. Distribution of the Value of Input and Hidden Layer's Output for 4,000 Frames

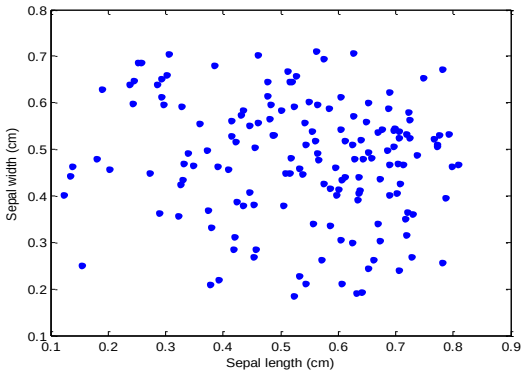
4. EXPERIMENTAL SETUP



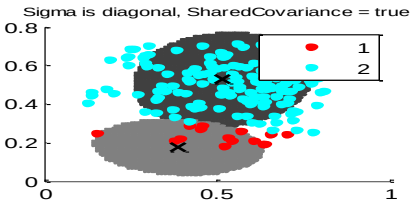
Features of SAE having learning rate of 0.02



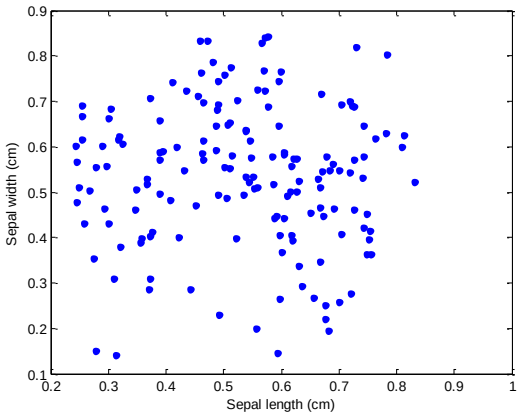
Classifier output 1 is double AMR samples and 2 is single AMR samples of learning rate 0.02



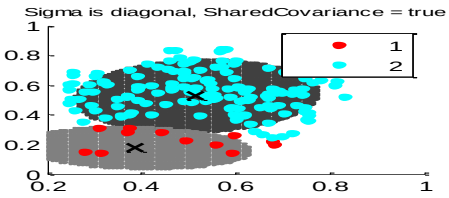
Features of SAE of 0.005 having learning rate



Classifier output 1 is double AMR samples and 2 is single AMR samples of learning rate 0.005



Features of SAE having learning rate of 0.001



Classifier output 1 is double AMR samples and 2 is single AMR samples of learning rate 0.001

Fig 4.1. A Feature of SAE (2nd hidden layer) and Classifier Output of Different Learning rate of 0.2, 0.05 and 0.01

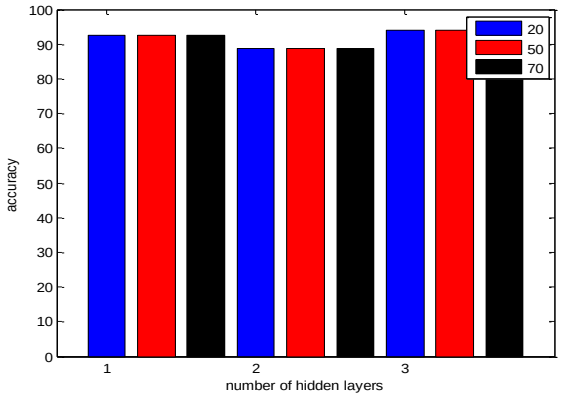


Fig .4.2 Accuracy of Proposed System DURING Different no. of Hidden Layer used and Having Different no. of Nodes in Hidden Layer

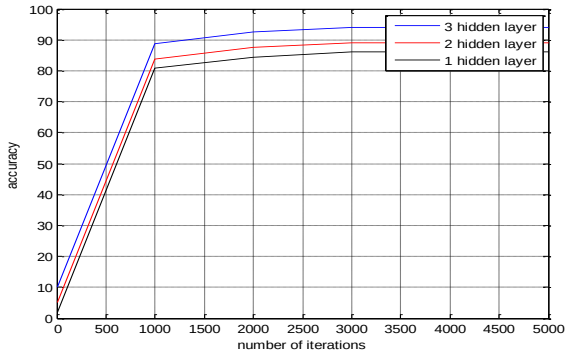


Fig. 4.3 Impact of the Number of Fine-Tuning Iterations with Learning Rate of 0.001 for Different Architectures

5. CLASSIFICATION BASED ON SUPPORT VECTOR MACHINE (SVM)

SVM uses linear models to implement nonlinear class boundaries. It transforms the input space using a nonlinear mapping into a new space (F feature space). Then a linear model constructed in the new space can represent a nonlinear decision boundary in the original space suppose a two-class dataset is linearly separable. The maximum margin hyperplane is the one that gives the greatest separation between the classes. Among all hyperplanes that separate the classes, the maximum margin hyperplane is the one that is as far away as possible from the two convex hulls, each of which is formed by connecting the instances of a class.The below figure shows hyperplane concept of SVM classifier.

The instances that are closest to the maximum hyperplane are called support vectors. There is at least one support vector for each class, and often there are more. A set of support vectors can uniquely define the maximum margin hyperplane for the learning problem. All other training instances are irrelevant; they can be removed without changing the position and orientation of the hyperplane.

5.1. The equation of the maximum margin hyperplane

In general, a hyperplane separating the two classes in the two-attribute case can be written as $F(x) = w + w\,a + w\,a$, where a1 and a2 are attribute values, and *w0*, *w1*, *w2* are weighting. The equation for the maximum margin hyperplane can also be written in

terms of support vectors.

$$f(x) = b + \Sigma\alpha i y i < xi \bullet x > \text{.....(5.1.1)}$$

Where

i is the support vector; *yi is* the class value of a training instance: either 1 or -1; *x i* is the *i*th support vector; x is a vector which represents a test instance;< *xi* • *x* > denotes the dot product of x and *i x* ; and b and α's are parameters that determine the hyperplane (just as the weights *w0*, *w1*, *w2*).

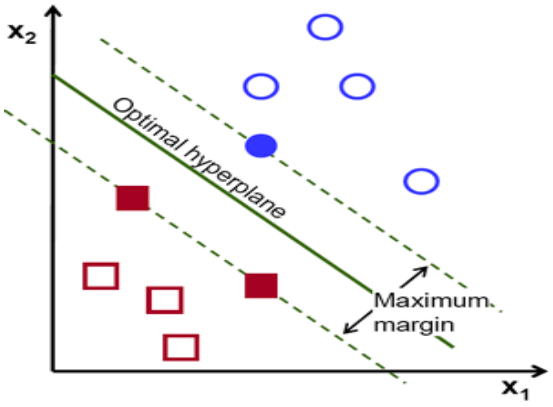


Fig 5.1.1 Hyper Plane of SVM Classifier

According to the equation of the maximum margin hyperplane above, every time an instance is classified, its dot product with all support vectors must be calculated. In a high-dimensional space resulting after a non-linear transformation is applied to the instances, the number of attributes in the new space could become huge and the computation of dot product becomes very expensive. The same problem occurs during training as well. Working in such a high dimensional space becomes intractable very quickly.

$$f(x) = b + \Sigma \alpha i yi < \phi(xi) \bullet \phi(x) > \text{.....(5.1.2)}$$

Where φ is a non-linear transformation function

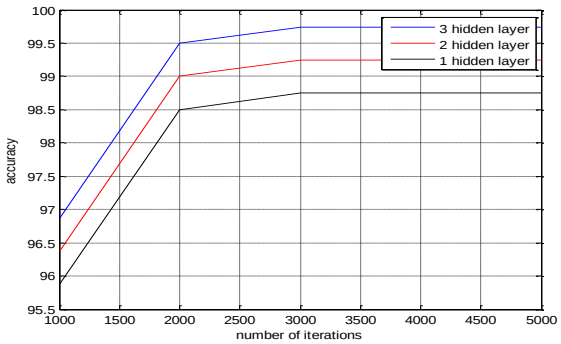
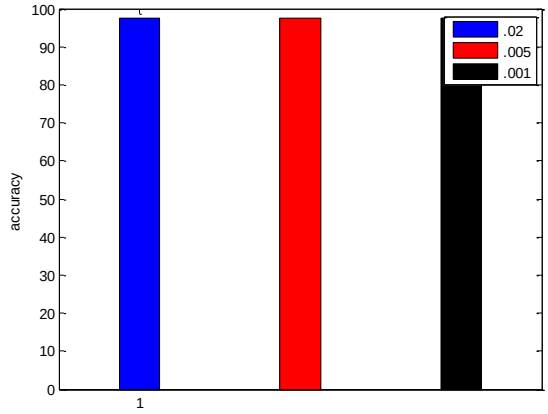


Fig 5.1.2 Accuracy Based on Hidden Layer

Fortunately, we don't need to explicitly construct a high dimensional features space. By using the kernel function, it is possible to calculate the dot product before the non-linear transformation is performed. In other words, a kernel function (5.3.2) can be applied to instances in the original input space which brings out the same effect as the linear transformation without expanding the feature space by non-linear transformation – *"kernel trick"*.



5.1.3 Accuracy Based on Learning Rate

6. CLASSIFICATION BASED ON BAYESIAN CLASSIFIER

A Bayesian classifier is a probabilistic model where the classification is a latent variable that is probabilistically related to the observed variables. Classification then becomes inference in the probabilistic model. A Bayesian classifier is based on the idea that the role of a (natural) class is to predict the values of features for members of that class. Examples are grouped in classes because they have common values for the features. Such classes are often called natural kinds. In this section, the target feature corresponds to a discrete class, which is not necessarily binary.

The idea behind a Bayesian classifier is that, if an agent knows the class, it can predict the values of the other features. If it does not know the class, Bayes' rule can be used to predict the class given (some of) the feature values. In a Bayesian classifier, the learning agent builds a probabilistic model of the features and uses that model to predict the classification of a new example.

A latent variable is a probabilistic variable that is not observed. A Bayesian classifier is a probabilistic model where the classification is a latent variable that is probabilistically related to the observed variables. Classification then becomes inference in the probabilistic model.

The simplest case is the naive Bayesian classifier, which makes the independence assumption that the input features are conditionally independent of each other given the classification. The independence of the naive Bayesian classifier is embodied in a particular belief network where the features are the nodes, the target variable (the classification) has no parents, and the classification is the only parent of each input feature. This belief network requires the probability distributions $P(Y)$ for the target feature Y and $P(X_i|Y)$ for each input feature X_i. For each example, the prediction can be computed by conditioning on observed values for the input features and by querying the classification.

Given an example with inputs $X_1=v_1,...,X_k=v_k$, Bayes' rule is used to compute the posterior probability distribution of the example's classification, Y:

$$P(Y \mid X_1=v_1,...,X_k=v_k)$$

$$= (P(X_1=v_1,...,X_k=v_k \mid Y) \times P(Y))/(P(X_1=v_1,...,X_k=v_k))$$

$$= (P(X_1=v_1|Y)\times\cdots\times P(X_k=v_k \mid Y)\times P(Y))/(\sum_Y P(X_1=v_1|Y)\times\cdots\times P(X_k=v_k \mid Y) \times P(Y))$$

Where the denominator is a normalizing constant to ensure the probabilities sum to 1. The denominator does not depend on the class and, therefore, it is not needed to determine the most likely class.

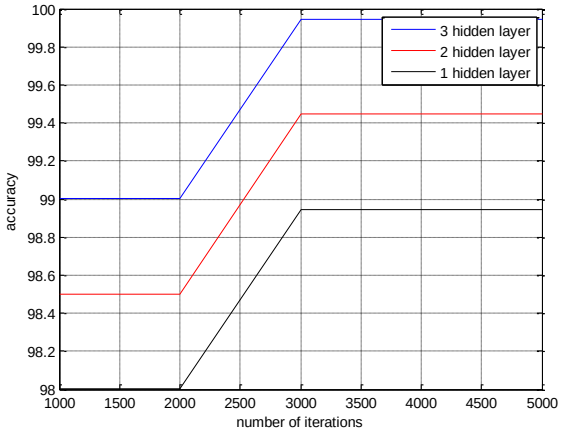


Fig 6.1 Accuracy Based on Hidden Layer

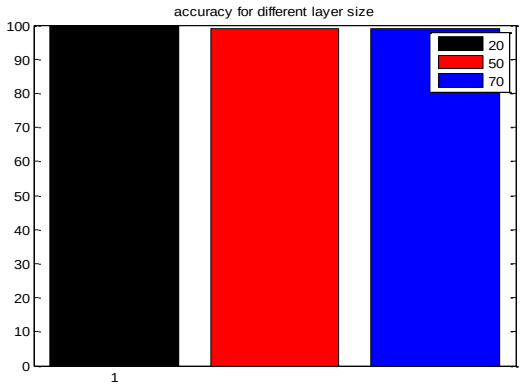


Fig 6.2 Accuracy Based on Hidden Layer

We test our proposed method on several databases. We elaborate the experimental setup for TIMIT because most of the analyses are based on TIMIT. For the other databases, the experimental setup is the same as TIMIT. The well-known database TIMIT contains 6,300 speech utterances from 630 people with 10 utterances for each person. We collected 6,000 one-second-long clips (cut from the database) and discarded the remaining shorter utterances**.** The audio clips are compressed with the AMR codec to obtain the single compressed AMR audio. For each audio clip, the compression bit-rate is randomly selected ranging from 4.75 12.2 kbps. Then, we decode the AMR audio and re-compress the clips to generate the double compressed AMR audio with random bit-rates still ranging from 4.75 12.2 kbps. The figure above shows the accuracy based on different hidden layer and learning rate. (6.1 and 6.2)

7. RESULT ANALYSIS

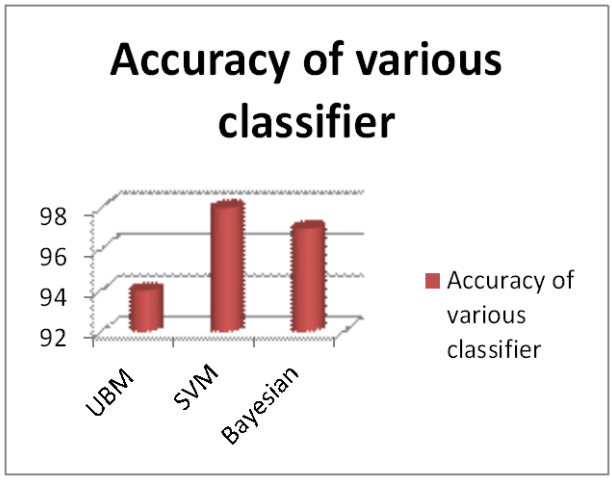


Fig 7.1 Accuracy of Various Classifier

Our goal is to distinguish the double compressed AMR audio clip from the single compressed one. We randomly choose 25% of the clips for training and the remaining ones for testing.

The input of the SAE network is the waveform samples of audio frames. A fixed frame size of 100 samples is used and the frame number is decided by the audio length. Each one-second audio clip (8,000 waveform samples) can be divided into 40 non-overlapping frames. Therefore, we obtain in total 480,000 (= 4000 clip X 40 frame classes) frames, including both classes of AMR compressed audio frames (single/double). The SAE network is trained and fine-tuned as described above. Then we use the output of the last hidden layer as the features (D-dimension) of the audio frame. For each audio clip, we obtain T = 40 feature vectors from all of its frames. Feature vectors are aggregated to form a 2-D array of size $D \times T$, which is used to train the UBM-GMM or SVM or Bayesian . For a given testing speech, we will first run a forward pass in the network to obtain the features. Then, we compute the log-likelihood ratios between UBM and both single/double AMR models. We make the final decision by comparing the ratios. In the experiments, double compressed AMR audio is regarded as positive samples while single compressed AMR audio as negative samples. We measure the rate of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The detection accuracy is defined as:

$$\text{Accuracy} = \frac{TP + TN}{(TP + FN) + (TN + FP)} \quad \text{.....7.1}$$

Different SAE training parameters will affect the detection results. The training procedure contains the SAE pre-training and fine-tuning steps. The key parameters for pre-training include SAE learning rate, number of nodes in the hidden layers, and the number of hidden layers. The key parameters of fine-tuning include learning rate and a number of epochs. In this section, we will investigate the effect of these parameters. Optimal parameters are selected by the grid search method. We first fix some of the parameters and search the optimal values for other parameters. In this part, the learning rate for NN fine-tuning uses a constant value of 0.1. In order to show the importance of fine-tuning the network, we evaluate two configurations: only SAE pre-training (with- out fine-tuning), and SAE pre-training with NN fine-tuning. We choose individual learning rates for both. The results are shown in Fig. 7.1 ie, acquire high percentage accuracy for SVM and Bayesian. Almost 97-98 percentage. But the UBM -GMM only shows the accuracy of about 94%.

8. CONCLUSION

Double compressed AMR audio usually implies a forgery; therefore, detection of double compressed AMR audio is an important issue in audio forensics. In this paper, we have presented a detection framework based on SAE network with different three classifiers and finally compare this three classifier. The main contributions are as follows:

- Most existing studies of audio forensics rely on hand- crafted features, which are challenging and time- consuming to design. We have used deep learning techniques to extract the meaningful features from a single audio frame automatically. Features from all the frames in an audio clip are aggregated and classified either by GMM or SVM or Bayesian.
- We have extracted features from the waveform of the audio signal and visualized the automatically learned statistical features to demonstrate their ability to distinguish between single and double compressed audio.

9. REFERENCES

[1] Detection of Double Compressed AMR Audio Using Stacked Autoencoder Da Luo, Member, IEEE, Rui Yang, Member, IEEE, Bin Li, Member, IEEE, Jiwu Huang, Fellow, IEEE.
[2] AMR Codec. [Online]. Available: http://www.3gpp.org/ftp/Specs/htmlinfo/.
[3] Speaker verification using adapted gaussian mixture models by douglas a. reynolds, thomas f. quatieri, and robert b. dunn.

[4] H. Su, R. Garg, A. Hajj-Ahmad, and M. Wu, "Enf analysis on recaptured audio recordings," in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13), 2013, pp. 3018–3022.
[5] H. Malik and H. Farid, "Audio forensics from acoustic reverberation," in Proc. of the International Conference on Acoustics Speech and Signal Processing, march 2010, pp. 1710–1713.
[6] R. Ynag,Y.Shi and j. Huang, "Defeating Fake quality MP3" in Proc of the ACM workshop on multimedia and security.
[7] Y. Shen, J. Jia, and L. Cai, "Detecting double compressed AMR-format audio recordings," in Proc. of the 10th Phonetics Conference of China (PCC), 2012.x.

