FORENSIC ANALYSIS AND LOCALIZATION OF MULTIPLY COMPRESSED MP3 AUDIO USING TRANSFORMERS

Ziyue Xiang[i], *Paolo Bestagini*[p], *Stefano Tubaro*[p], *Edward J. Delp*[i]

[i]Video and Image Processing Lab (VIPER), School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, USA

[p]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

ABSTRACT

Audio signals are often stored and transmitted in compressed formats. Among the many available audio compression schemes, MPEG-1 Audio Layer III (MP3) is very popular and widely used. Since MP3 is lossy it leaves characteristic traces in the compressed audio which can be used forensically to expose the past history of an audio file. In this paper, we consider the scenario of audio signal manipulation done by temporal splicing of compressed and uncompressed audio signals. We propose a method to find the temporal location of the splices based on transformer networks. Our method identifies which temporal portions of a audio signal have undergone single or multiple compression at the temporal frame level, which is the smallest temporal unit of MP3 compression. We tested our method on a dataset of 486,743 MP3 audio clips. Our method achieved higher performance and demonstrated robustness with respect to different MP3 data when compared with existing methods.

Index Terms— MP3 compression, audio forensics, convolutional neural networks, transformer networks

1. INTRODUCTION

The advance in machine learning techniques makes generating high-quality speech or music possible [1]–[3]. Almost everyone has the ability to tamper with audio signals due to the availability of audio editing techniques. It is easy to synthesize/manipulate "fake" speech signals that resembles the style and voice of a given person, and to concatenate real and synthetic signals to create new forged speech signals. This poses significant threats to individuals, organizations, society and national security.

The detection of synthesize/manipulated speech is usually challenging because of the flexibility of human voice, the presence of acoustic noise or reverberation, and the complexity of audio synthesis models [4]. However, audio signals are mostly saved and shared using lossy compression techniques. Lossy compression leaves distinct artifacts in the compressed audio which can be used for forensic analysis [5], [6]. In this paper we examine the forensic problem of detecting if an audio signal has been spliced into another audio signal by detecting locations in the signal that have been multiply compressed. The spliced audio signal may be real or synthetic.

Introduced in 1993, MPEG-1 Audio Layer III (MP3) [7], [8] has been one of the most popular digital audio compression methods. It changed our ways of listening to music, podcasts, and many other types of audio content. Despite having inferior compression efficiency compared to Advanced Audio Codec (AAC) [9], MP3 is still widely used due to compatibility with many existing applications and lower computational complexity [10].

Most of the existing MP3 audio forensics methods focus on double MP3 compression detection. These methods predict whether an entire MP3 file is compressed more than once. In [5], [6], [11], [12], the authors used different statistical and feature design techniques on the Modified Discrete Cosine Transform (MDCT) coefficients for double compression detection. Ma *et al.* [13] used the statistics of scalefactors for detecting doubly compressed MP3 with the same data rate. In [14], the authors used the Huffman table indices for double compression detection. Yan *et al.* [15] addressed the problem of double and triple compression detection using features extracted from scalefactors and Huffman table indices. In [16], the authors addressed the MP3 multiple compression localization problem for the first time. Essentially, their proposed method detects double compression using the histogram of MDCT coefficients. Localization was achieved by using small sliding detection windows. This technique allows one to extend any detection method to a localization method. Finally, more modern methods make use of Deep Neural Networks (DNNs). This is the case of Luo *et al.* [17] that proposed to use stacked autoencoder networks to detect multiply compressed audio signals.

In this paper, we present an MP3 multiple compression localization technique based on deep transformer neural networks [18]. Given an MP3 signal, our proposed method can distinguish between single compressed temporal segments and multiple compressed temporal segments thereby allowing us to temporally localize where the audio signal may have been spliced. Our proposed method can also be used for synthesized audio detection.

2. BACKGROUND

We first introduce a basic overview of MP3 compression. More details of MP3 compression can be obtained in [7], [8], [10]. Then we provide an overview of transformer neural networks.

2.1. MP3 Compression

The block diagram of a typical MP3 encoder is shown in Figure 1. To compress a digital audio signal using MP3, the signal is first split into fixed time length sample windows known as frames, where each frame contains 1152 temporal samples [7], [8]. MP3 files are made up of a series of such frames.

The input is first processed through a perceptual model that drives the selection of coding parameters (lower branch of Figure 1). During this step, the audio samples are transformed into frequency domain using Fast Fourier Transform (FFT). The FFT magnitude is passed to the psychoacoustic model, which exploits the characteristics of Human Hearing System (HHS) to balance between the sound quality and the data rate of the compressed signal [19]. After this per-

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under agreement number FA8750-20-2-1004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL or the U.S. Government. Address all correspondence to Edward J. Delp, `ace@ecn.purdue.edu`.

ceptual audio analysis, the lossy coding step is next (upper branch of Figure 1). The temporal samples are filtered into 32 equally spaced frequency sub-bands using a polyphase filterbank. Each sub-band is windowed according to the psychoacoustic model to reduce artifacts and then transformed through MDCT, which leads to 18 coefficients. In total, there are $32 \times 18 = 576$ MDCT coefficients.

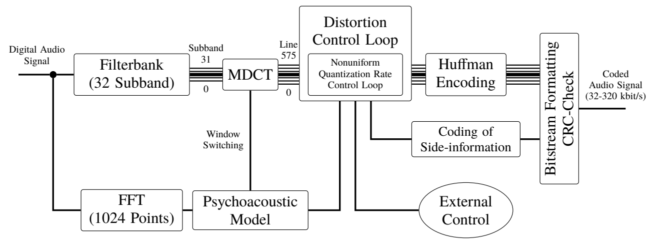

Fig. 1: The block diagram of an MP3 encoder.

After the MDCT, the resulting coefficients are quantized in the distortion control loop exploiting the psychoacoustic analysis. The HHS has approximately 24 critical bands based on a model of the HHS [19]. During quantization, the 32 sub-bands are grouped into scalefactor bands, which approximates the critical bands of HHS. The MDCT coefficients in each scalefactor band is multiplied by a scalefactor before quantization. The quantization step size increases as the frequencies become greater, because the HHS is less sensitive to higher frequency. The scalefactors and quantization step sizes work together to satisfy both audio quality and data rate constraints. After quantization, the MDCT coefficients are binary encoded using one of the 32 predefined Huffman tables. The binary coded coefficients, the encoding parameters (side-information) such as scalefactors, Huffman table indices, quantization step sizes are inserted in the data stream to form the compressed audio signal.

2.2. Transformer Neural Networks

Transformer networks have shown excellent performance in a variety of tasks such as language modeling [20], image classification [21], object detection [22], protein structure prediction [23].

Let the input to the transformer network be $\boldsymbol{Z} \in \mathbb{R}^{N \times d_{\text{model}}}$, which contains N elements, and each element is a vector of d_{model} dimensions. Transformer networks use the Self Attention (SA) mechanism [18], [21] to exploit the relationship between different elements in the input. Linear projection $\boldsymbol{U}_{qkv} \in \mathbb{R}^{d_{\text{model}} \times 3d_h}$ is first used to generate three different projected versions of the input, i.e., $\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$:

$$[\boldsymbol{q} \mid \boldsymbol{k} \mid \boldsymbol{v}] = \boldsymbol{Z}\boldsymbol{U}_{qkv}. \tag{1}$$

Then, the vectors $\boldsymbol{q}$ and $\boldsymbol{k}$ are used to form the attention map $\boldsymbol{A} \in \mathbb{R}^{N \times N}$:

$$\boldsymbol{A} = \text{softmax}\left(\boldsymbol{q}\boldsymbol{k}^T / \sqrt{d_h}\right). \tag{2}$$

Finally, the SA of $\boldsymbol{Z}$ is the matrix multiplication of $\boldsymbol{A}$ and $\boldsymbol{v}$:

$$\text{SA}(\boldsymbol{Z}) = \boldsymbol{A}\boldsymbol{v}. \tag{3}$$

The transformer networks we use are based on Multihead Self Attention (MSA) [18], which is an extension of SA where h different SA values (called "heads") are computed in parallel. The h different SA values are combined to the result of MSA using the matrix $\boldsymbol{U}_{msa} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$:

$$\text{MSA}(\boldsymbol{Z}) = \left[\text{SA}_1(\boldsymbol{Z}) \mid \cdots \mid \text{SA}_h(\boldsymbol{Z})\right] \boldsymbol{U}_{msa}. \tag{4}$$

One must guarantee h divides d_{model} and set $d_h = d_{\text{model}}/h$ so that $\text{MSA}(\boldsymbol{Z})$ and $\boldsymbol{Z}$ have the same dimensionality. More details about transformers can be found in [18], [21].

3. MULTIPLE MP3 COMPRESSION LOCALIZATION

In this section we first introduce the temporal splicing detection and localization problem. We then present our proposed solution.

3.1. Problem Formulation

In MP3 compression, each audio signal is composed by a series of nonoverlapping fixed temporal length segments known as frames. Let $\boldsymbol{x}$ be the audio signal $\boldsymbol{x} = \{x_1, x_2, \ldots, x_L\}$, where x_l is the l-th frame of the signal, and L is the total number of frames, which depends on the signal length. We can associate to the audio file a sequence of labels $\boldsymbol{y}$ defined as $\boldsymbol{y} = \{y_1, y_2, \ldots, y_L\}$, where y_l is the l-th binary valued label indicating whether the l-th frame (x_l) has been compressed once (i.e., $y_l = 0$) or more than one time (i.e., $y_l = 1$).

During the creation of a spliced MP3 audio file, it is likely that frames from different audio signals are concatenated in time and compressed using MP3. Some of the audio signals can be uncompressed (e.g., pristine or generated from a synthetic speech), while others may have been compressed and then decompressed. The final spliced signal will likely contain both single and multiple compressed frames.

Our goal is to find $\hat{\boldsymbol{y}}$, which is an estimate of the sequence of labels $\boldsymbol{y}$ associated with the audio file $\boldsymbol{x}$ by examining the MP3 compressed data for $\boldsymbol{x}$ on a frame by frame basis. In doing so, we are able to detect if an audio file has undergone splicing, and localize which frames have been compressed more than one time.

3.2. Proposed Method

Our proposed method is shown in Figure 2. We examine the MP3 data corresponding to the individual frames of the input signal $\boldsymbol{x} = \{x_1, x_2, \ldots, x_L\}$. We examine L frames at a time to decide if the audio signal has been multiply compressed and localize the splicing. A MP3 compressed frame consists of two groups of samples known as granules [8], [9]. Each granule contains 576 temporal samples from the respective two stereo channels. Our proposed method uses the data produced by the MP3 codec for a frame shown in Table 1 from the first channel of the first granule. This data consists of MDCT coefficients, quatization step sizes, scalefactors, Huffman table indices, and sub-band window selection information. The MP3 codec data we choose contain a complete set of parameters required to decode the channel. We will use this information to decide if an audio signal has been multiple compressed and localize the muliple compressed frames.

The `mdct_coef` and `scalefactor` fields are used as inputs to two separate Convolutional Neural Networks (CNNs) (i.e., CNN-1 and CNN-2) to find more features (see Figure 2). We use CNN architectures that are similar to the well known VGG CNN [24]. The depth and number of filters are changed based on the size of the `mdct_coef` and `scalefactor` fields. By denoting the convolutional layer as Conv<receptive field size>–<number of filters> and the fully connected layers as FC–<number of neurons>, the architecture and parameters of each CNN are as follows:

- CNN-1: Conv3-32, Conv3-32, Maxpool, Conv3-64, Conv3-64, Maxpool, Conv3-128, Conv3-128, Maxpool, FC-233, Dropout, FC-233, Dropout.
- CNN-2: Conv3-16, Conv3-16, Maxpool, Conv3-32, Conv3-32, Maxpool, Conv3-64, Conv3-64, Maxpool, FC-49, Dropout, FC-49, Dropout.

For the l-th frame, we concatenate the outputs of CNN-1 and CNN-2 as well as the values of the remaining fields into a feature vector $z_l \in \mathbb{R}^{d_{\text{model}}}$, which is used as the input to the transformers. The sizes of the CNN outputs are selected so that the dimensionality of z_l's satisfies $d_{\text{model}} = 300$. The self attention mechanism does not use the order information of the elements in the input sequence. To allow the transformer to make use of the temporal order of frames, the frames' corresponding feature vectors are added with positional encoding [18]. The positional encoding is a series of L distinct fixed-valued vectors $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_L\}$, where $\boldsymbol{p}_l \in \mathbb{R}^{d_{\text{model}}}$. After adding

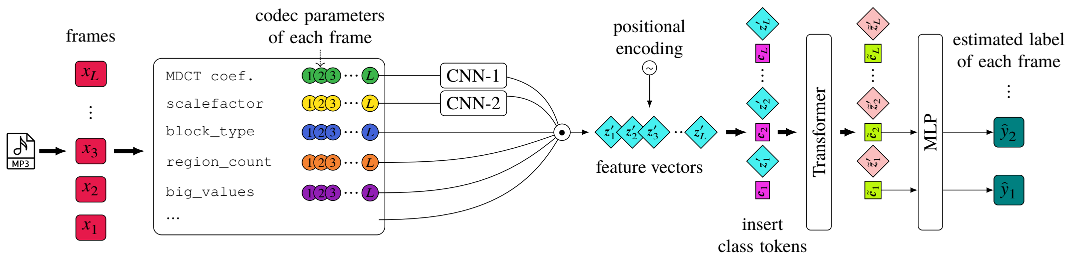

Fig. 2: The block diagram of the proposed method, which analyzes L frames at a time. Each circular node (ⓛ) represents a MP3 codec parameter vector whose corresponding frame is shown by the number inside. Each diamond node (◈) represents a vector associated with the feature vector z'_l. Each rectangular node ($\boxed{c_l}$) represents a vector associated with the class token $\boldsymbol{c}_l$. Different groups of vectors are shown in different colors.

Table 1: MP3 codec information fields used in our method. Details about each field can be obtained from [25], [26].

Fields	Description
`part_23_length`	Size of coded binary data
`scalefactor`, `scalefac_compress`, `scalefac_scale`, `preflag`	Scalefactor value info
`global_gain`, `subblock_gain`, `big_values`, `region_count`	Quantization step sizes
`table_select`, `count1_table`	Huffman table selection info
`block_type`, `mixed_block_flag`	Sub-band window selection info
`mdct_coef`	Decoded MDCT coefficients

positional encoding, the l-th feature vector is $z'_l = z_l + \boldsymbol{p}_l$. The transformer will likely be able to find the position of z'_l being l based on the $\boldsymbol{p}_l$ component of z'_l.

We use a similar approach as described in [21] to binary classify the feature vectors corresponding for each MP3 frame as being "single compressed" or "multiple compressed". The network has L special vectors, known as "class tokens", which are denoted by $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_L\}$, where $\boldsymbol{c}_l \in \mathbb{R}^{d_{\text{model}}}$. The input to the transformer $\boldsymbol{Z}$ is the feature vector z_l's interleaved with the class tokens, which can be written as

$$\boldsymbol{Z} = [\boldsymbol{c}_1 \mid z'_1 \mid \boldsymbol{c}_2 \mid z'_2 \mid \ldots \mid \boldsymbol{c}_L \mid z'_L] \in \mathbb{R}^{2L \times d_{\text{model}}}. \quad (5)$$

Let $\text{tf}(\cdot)$ be the function corresponding to the transformer network. After the transformer operations, the output can be written as

$$\text{tf}(\boldsymbol{Z}) = [\tilde{\boldsymbol{c}}_1 \mid \tilde{z}'_1 \mid \tilde{\boldsymbol{c}}_2 \mid \tilde{z}'_2 \mid \ldots \mid \tilde{\boldsymbol{c}}_L \mid \tilde{z}'_L] \in \mathbb{R}^{2L \times d_{\text{model}}}. \quad (6)$$

To find the estimated labels for the l-th frame, we use a Multilayer Perceptron (MLP) network to classify $\tilde{\boldsymbol{c}}_l$. During training, the gradients with respect to $\boldsymbol{c}_l$'s are computed and used to update them using gradient descent [27]. That is, we train each class token to collect information necessary to determine the label for its corresponding time step. Finally, the label of each frame can be determined by only examining the class tokens.

After preliminary experiments, we used 8 transformer layers with the number of heads $h = 15$ in the Multihead Self Attention (see Section 2.2), which yielded the best performance.

The MLP network we use is made up of one layer of 800 neurons and a final output layer with 2 neurons using softmax activation. We use the GELU [28] nonlinear activation function in the CNNs and the transformer network.

After training, for a given MP3 audio frame sequence $\{x_1, x_2, \ldots, x_L\}$, our method generates a binary decision sequence $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_L\}$ describing whether each frame is multiply compressed or not. This enables one to localize the questionable frames in an MP3 file and determine which part of the MP3 frames may have been spliced.

4. EXPERIMENTS AND RESULTS

In this section we describe our experiments and present results comparing our method to other methods for splicing localization using traces of multiple MP3 compression.

4.1. Dataset

The data for our experiments contain only uncompressed WAV audio files. We used three publicly available datasets consisting of speech and different music genres: LJSpeech [29], GTZAN [30], and MAESTRO [31]. We then compressed the WAV files using MP3. The MP3 sampling rate we selected is 44.1 kHz, and the length of each frame is $1152/44100 \approx 26.12$ms. In our experiments, our method examined $L = 20$ frames at a time.

The MP3 compression data rates and data rate types used in our experiments are shown in Table 2. We used Constant Bit Rate (CBR) and Variable Bit Rate (VBR) compression. The audio signals were compressed using FFmpeg[1] v3.4.8 and LAME[2] v3.100. We excluded low-quality MP3 compression configurations that make multiple-compression detection unreliable [15]. We also excluded ultra-high-quality MP3 compression that is not recommended by FFmpeg [32].

Table 2: MP3 compression types used in our experiments. The data rate of VBR compression decreases as quality index increases. More details can be obtained from [32].

Compression type	Bit rate/Quality
Constant Bit Rate (CBR)	64kbps, 96kbps, 128kbps, 160kbps, 192kbps, 256kbps
Variable Bit Rate (VBR)	1, 2, 3, 4, 5, 6

Our dataset generation scheme is described as follows. We split each uncompressed WAV audio file into smaller segments of 80–320 frames randomly. Each segment is further temporally subdivided into "slices" of $L/2 = 10$ frames where L is the number of frames our method examines at a time. Now, each segment will contain 8–32 slices. In each segment, the odd slices will be multiple compressed for 2–3 times. The even slices will be single compressed. The compression parameters are chosen from Table 2 at random. In Figure 3 we illustrate the dataset generation method on a segment containing 4 slices. If a slice needs to be compressed three times, then compression 1–3 will be used. If a slice needs to be compressed two times, then compression 2–3 will be used. Otherwise, only compression 3 will be used. For a given slice, after compression 1 the MP3 file is decompressed back to the time domain, which is used as the input to compression 2. At compression 3, we gather the decompressed or pristine time domain samples of all slices in a segment and then compress them as one MP3 file. Therefore, the parameters for compression 3 will be the same for all slices in a given segment. This completes the construction of our ground-truthed experimental data set.

[1]https://www.ffmpeg.org/
[2]https://lame.sourceforge.io/

	Slice 1 $L/2 = 10$ frames	Slice 2 $L/2 = 10$ frames	Slice 3 $L/2 = 10$ frames	Slice 4 $L/2 = 10$ frames
Compression 1	CBR, 256kbps			
Compression 2	VBR, Q=4		CBR, 160kbps	
Compression 3	CBR, 64kbps	CBR, 64kbps	CBR, 64kbps	CBR, 64kbps
$\boldsymbol{y}$ = [	1,1,1,1,1,1,1,1,1,1,	0,0,0,0,0,0,0,0,0,0,	1,1,1,1,1,1,1,1,1,1,	0,0,0,0,0,0,0,0,0,0]

Fig. 3: An illustration of our dataset generation method on a segment of 40 frames (4 slices). The corresponding label for this segment $\boldsymbol{y}$ is shown at the bottom.

4.2. Preparing an MP3 file For Training

Recall our method operates in the "MP3 domain" by examining the MP3 codec fields shown in Table 1. To train our method, we used a sliding window of length L with offset step size 8 to extract the MP3 codec fields from the MP3 compressed frames as shown in Table 1.

We generated 486,743 MP3 frame sequences of length L from our experimental dataset. We used 54% of the frame sequences in the training set; 13% of the frame sequences in the validation set, and 33% of the frame sequences in the testing set. The partition was done in a way that all frame sequences generated from one segment can only be used in one of the three sets.

Selecting the MP3 frame sequence length L for analysis is a trade-off between the accuracy of the estimation of $\hat{\boldsymbol{y}}$ and the training difficulty of the network. Using a larger L may improve the performance, but it may also significantly increase training time and the need of hyperparameter tuning. In our experiments, we used $L = 20$ corresponding to an audio signal length of approximately 522.4ms. This may be small compared to the typical length of most audio signals. In training we used a sliding window stride of 8, which does not align with the slice boundaries. In each sliding window, the index of the first multiple compressed frame and the number of multiple compressed frame are different. This forces the trained network to predict the labels correctly given an arbitrary portion taken from an audio signal. Therefore, our method is able to examine longer audio files even if L is relatively small.

4.3. Hyperparameters and Training

The entire network including the CNNs and the transformers are trained end-to-end. We trained the network using the Adam optimizer [33] with an initial learning rate of 10^{-4} and a dropout rate of 0.2 until the validation accuracy no longer increased for 20 epochs.

4.4. Results

We compared the performance of our method to [6], [11], [15], which are introduced in Section 1. Since they are all detection methods, we used the predictions from these methods on short frame sequences of length $L' = 4$ to approximate localization. Choosing the length L' for localization approximation is a trade-off between classification accuracy and granularity of localization. Larger L' improves the label estimation of each frame sequence, while smaller L' improves the localization accuracy. Selecting $L' = 4$ is a reasonable balance between these two factors [16].

In Table 3, we compare the performance of our method to that of the three previous methods. We used three different metrics: Jaccard score [34], F_1-score [35] and balanced accuracy score [36]. Let $\mathcal{I}(\boldsymbol{y})$ be the function that returns the set of indices for the one-entries in $\boldsymbol{y}$. The Jaccard score can be computed by $|\mathcal{I}(\boldsymbol{y})\cap\mathcal{I}(\hat{\boldsymbol{y}})|/|\mathcal{I}(\boldsymbol{y})\cup\mathcal{I}(\hat{\boldsymbol{y}})|$, which compares the similarity between the predicted multiple compressed region and the ground truth multiple compressed region [34]. The F_1-score is the harmonic mean of the precision and recall, which considers both factors at the same time [35]. The balanced accuracy score is the traditional accuracy score with class-balanced sample weights [36]. Our approach achieved the highest score on all three metrics. We also tested our method on a separately generated dataset containing 53,541 MP3 frame sequences with variable slice length ranging from 10 to 80 frames. For each slice, the slice length, compression types and the number of compressions are chosen at random. Our method achieved 81.64 balanced accuracy on this dataset, which is close to the result in Table 3. This shows our method did not learn strong dataset bias.

In Table 4, we show the recall [35] of each method on multiply compressed MP3 frames against selected last MP3 compression types. It can be seen that the detection accuracy decreases as the MP3 compression quality declines. Different methods reacted to CBR and VBR compression in contrasting manners. For [6] and our method, the performance was similar. For [11], the score of VBR frames was significantly lower than those of CBR; the behavior of [15] was the opposite.

In table 5, we show the recall of each method compared to the number of MP3 compression applied to a frame. The recall of our method is more consistent across different repetition of MP3 compression. For all methods, the recall for double compression is close to that of triple compression.

Overall, our approach demonstrated high localization performance and robustness across many types of MP3 compression.

Table 3: Performance metrics comparison.

Method	Jaccard Score	F_1-score	Balanced Accuracy
Yan *et al.* [15]	13.53	18.71	53.35
Yang *et al.* [11]	30.73	40.95	55.28
Liu *et al.* [6]	48.18	58.91	68.72
Our Approach	80.50	84.43	84.49

Table 4: The recall of multiple compression localization of each method against selected last MP3 compression types. CBR compression is denoted by C<bit rate>; VBR compression is denoted by V<quality index>.

	Last MP3 Comression Type							
Method	C64	C128	C160	C192	V1	V2	V4	V6
Yan *et al.* [15]	17.30	6.86	6.80	4.87	22.80	23.23	22.59	17.55
Yang *et al.* [11]	19.27	70.75	71.71	66.21	45.58	39.05	27.12	22.06
Liu *et al.* [6]	58.45	56.48	54.47	55.01	55.15	56.41	57.47	67.93
Our Approach	73.09	88.06	89.65	90.84	93.31	91.21	83.52	65.51

Table 5: The recall of each method against the number of MP3 compression.

	Num. MP3 Compression			
Method	Single	Double	Triple	Overall
Yang *et al.* [11]	67.28	43.51	43.02	43.27
Yan *et al.* [15]	91.71	14.86	15.10	14.98
Liu *et al.* [6]	79.36	57.27	58.85	58.06
Our Approach	84.61	83.76	84.92	84.34

5. CONCLUSIONS

We proposed a multiple MP3 compression temporal localization method based on transformer neural networks that uses MP3 compressed data. Our proposed method localizes multiple compression at the frame level. The experiment results showed that our method had the best performance compared to other approaches and was robust against many MP3 compression settings. In the future, we will examine extending this approach to other compression methods such as AAC. We will also investigate the use of stereo channels as well as the second granule in MP3 compressed frames. We will generalize the concept of multiple compression detection to compression history detection. That is, to find the number of compressions and the types of compression used. Knowing the compression history can greatly enhance the interpretability for audio forensics.

6. REFERENCES

[1] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3617–3621, 2019, Brighton, UK.

[2] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Fastspeech 2: Fast and high-quality end-to-end text to speech,” *arXiv preprint arXiv:2006.04558*, 2020.

[3] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.

[4] C. Borrelli, P. Bestagini, F. Antonacci, A. Sarti, and S. Tubaro, “Synthetic speech detection through short-term and long-term prediction traces,” *EURASIP Journal on Information Security*, vol. 2021, no. 2, pp. 1–14, 2021.

[5] R. Yang, Y.-Q. Shi, and J. Huang, “Defeating fake-quality MP3,” *Proceedings of the 11th ACM Workshop on Multimedia and Security*, pp. 117–124, 2009, Princeton, NJ, USA.

[6] Q. Liu, A. H. Sung, and M. Qiao, “Detection of double MP3 compression,” *Cognitive Computation*, vol. 2, no. 4, pp. 291–296, 2010.

[7] *ISO/IEC 13818-3:1995 - Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio*, `https://www.iso.org/standard/22991.html`.

[8] H. Musmann, “Genesis of the mp3 audio coding standard,” *IEEE Transactions on Consumer Electronics*, vol. 52, no. 3, pp. 1043–1049, 2006.

[9] *ISO/IEC 13818-7:1997 - Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC)*, `https://www.iso.org/standard/25040.html`.

[10] K. Brandenburg, “MP3 and AAC explained,” *Proceedings of the AES 17th International Conference on High-Quality Audio Coding*, 1999, Signa, Italy.

[11] R. Yang, Y. Q. Shi, and J. Huang, “Detecting double compression of audio signal,” *Media Forensics and Security II*, vol. 7541, pp. 200 –209, 2010.

[12] M. Qiao, A. H. Sung, and Q. Liu, “Improved detection of MP3 double compression using content-independent features,” *Proceedings of 2013 IEEE International Conference on Signal Processing, Communication and Computing*, pp. 1–4, 2013, KunMing, China.

[13] P. Ma, R. Wang, D. Yan, and C. Jin, “Detecting double-compressed MP3 with the same bit-rate.,” *Journal of Software*, vol. 9, no. 10, pp. 2522–2527, 2014.

[14] ——, “A huffman table index based approach to detect double MP3 compression,” in *Digital-Forensics and Watermarking*, Berlin, Heidelberg, 2014, pp. 258–271.

[15] D. Yan, R. Wang, J. Zhou, C. Jin, and Z. Wang, “Compression history detection for MP3 audio,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 12, no. 2, pp. 662–675, 2018.

[16] T. Bianchi, A. De Rosa, M. Fontani, G. Rocciolo, and A. Piva, “Detection and localization of double compression in mp3 audio tracks,” *EURASIP Journal on information Security*, vol. 2014, no. 10, 2014.

[17] D. Luo, W. Cheng, H. Yuan, W. Luo, and Z. Liu, “Compression detection of audio waveforms based on stacked autoencoders,” in *Artificial Intelligence and Security*, Cham, 2020, pp. 393–404.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, vol. 30, 2017, pp. 5998–6008.

[19] N. Jayant, J. Johnston, and R. Safranek, “Signal compression based on models of human perception,” *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1385–1422, 1993.

[20] T. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” *Proceedings of 2020 European Conference on Computer Vision*, pp. 213–229, 2020.

[23] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[24] S. Liu and W. Deng, “Very deep convolutional neural network based image classification using small training sample size,” *Proceedings of 2015 3rd IAPR Asian Conference on Pattern Recognition*, pp. 730–734, 2015.

[25] R. Raissi, *The theory behind MP3*, 2002. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.6804`.

[26] P. Sripada, “MP3 decoder in theory and practice,” M.S. thesis, Blekinge Institute of Technology, Ronneby, Sweden, Mar. 2006. [Online]. Available: `https://www.diva-portal.org/smash/get/diva2:830195/FULLTEXT01.pdf`.

[27] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[28] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” *arXiv preprint arXiv:1606.08415*, 2016.

[29] K. Ito and L. Johnson, *The lj speech dataset*, 2017. [Online]. Available: `https://keithito.com/LJ-Speech-Dataset/`.

[30] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[31] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” *Proceedings of the International Conference on Learning Representations*, 2019.

[32] *FFmpeg MP3 encoding guide*. [Online]. Available: `https://trac.ffmpeg.org/wiki/Encode/MP3`.

[33] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[34] M. Levandowsky and D. Winter, “Distance between sets,” *Nature*, vol. 234, no. 5323, pp. 34–35, 1971.

[35] D. M. Powers, “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,” *arXiv preprint arXiv:2010.16061*, 2020.

[36] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, “The balanced accuracy and its posterior distribution,” *2010 20th International Conference on Pattern Recognition*, pp. 3121–3124, 2010, Istanbul, Turkey.