Time Scaling Detection and Estimation in Audio Recordings

Michele Pilia, Sara Mandelli, Paolo Bestagini, Stefano Tubaro
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

2021 IEEE International Workshop on Information Forensics and Security (WIFS) | 978-1-6654-1717-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/WIFS53200.2021.9648389

***Abstract*—The widespread diffusion of user friendly editing software for audio signals has made audio tampering extremely accessible to anyone. Therefore, it is increasingly necessary to develop forensic methodologies aiming at verifying if a given audio content has been digitally manipulated or not. Among the multiple available audio editing techniques, a very common one is time scaling, i.e., altering the temporal evolution of an audio signal without affecting any pitch component. For instance, this can be used to slow-down or speed-up speech recordings, thus enabling the creation of natural sounding fake speech compositions. In this work, we propose to blindly detect and estimate the time scaling applied to an audio signal. To expose time scaling, we leverage a Convolutional Neural Network that analyzes the Log-Mel Spectrogram and the phase of the Short Time Fourier Transform of the input audio signal. The proposed technique is tested on different audio datasets, considering various time scaling implementations and challenging cross test scenarios.**

I. Introduction

The diffusion of electronic devices and multimedia platforms to share user-generated contents has been growing exponentially in the last decades. This, together with the drastic technological evolution, has resulted in high ease of production and editing of multimedia data with professional results. These phenomena have led to the research of multimedia forensic methods to identify whether a given image, video, or audio content is original or digitally-tampered. Concerning audio forensics, several techniques have been developed for different purposes, from outdoor-vs-indoor recording detection [1], to detection of post-processing operations [2], [3], [4], [5].

A very common modification applied to audio signals is that of time scaling, that is, to speed up or slow down an audio track without affecting its spectral content [6]. This comes handy in various applications. For instance, we can cite post-synchronization in cinema and music industry, where songs or dialogues need to be synchronized with images or video frames, thus there might be the need to adapt the speed of some audio signals. Another example is the "reading by listening", in which time scaling can increase the listening rate, providing blind people a valid alternative to reading.

Despite the undeniable advantages, time scaling could be exploited for malicious purposes as well. Indeed, time scaling enables the creation of natural sounding fake content and may lead to serious threats, holding a wide appeal with the general public. Audio tracks could be doctored to make conversations more brilliant (by speeding up the time evolution) or weaker (by slowing down the time evolution). The consequences of an uncontrolled use of time scaling might be catastrophic, helping the spreading of fake news and misinformation. For example, we may imagine what could happen to public opinion if politicians' interviews were time scaling modified in order to favour one political party with respect to another one [7].

WIFS'2021, December, 7-10, 2021, Montpellier, France.

In this paper, we propose to blindly identify traces of time scaling on audio signals and to estimate the precise time scaling factor applied. To this purpose, we rely on potential traces left by time scaling algorithms on the modified audio tracks. For instance, whenever audio signals are speeded up, we aim at tracing possible downsampling footprints; whenever audio signals are slowed down, we aim at spotting interpolation artifacts left by unavoidable resampling techniques needed to generate new audio samples. It is important to notice that time scaling is a far more complex operation than resampling. Resampling model is straightforward, whereas time scaling leaves artifacts that are more challenging to model and depend on the specific implementation, therefore resampling detectors as [8], [9], [10], [11] could not be straightforwardly applied.

Given a query audio file, we identify and estimate time scaling by leveraging a Convolutional Neural Network (CNN)-based solution. We propose to pre-process each audio file using three different methodologies which explore the audio Log-Mel Spectrogram (LMS) and the phase of the Short Time Fourier Transform (STFT). Then, we feed these data to the proposed network by exploiting four diverse configurations.

We evaluate the proposed methodology over almost 4000 original audio files selected from two well known datasets available in literature [12], [13]. Each file is modified by means of three different time scaling implementations. We work in challenging scenarios, considering cross tests among diverse time scaling implementations (i.e., when training and testing data come from different algorithms), and testing audio files modified with unknown time scaling factors never seen at training stage. Our method demonstrates to be a valid strategy for time scaling detection and estimation. However, it also highlights the need for further investigations in case time scaling is applied to shorten the signal duration.

II. Background and Problem Formulation

A. Mel scale and Log-Mel Spectrogram

The Mel scale is a frequency representation that aims to mimic the human non-linear perception of sound, by being

more discriminative at lower frequencies and less discriminative at higher frequencies [14]. We can simulate the human ear's behaviour by means of the Mel filterbank, a set of K triangular filters, which can be modeled as a 2D matrix $\mathbf{H}$ with size $K \times F$, where rows contain coefficients associated with different filters (related to K distinct pitches in Mel scale) and columns are related to frequencies in Hertz [14].

By applying the Mel filterbank $\mathbf{H}$ to the spectrogram of an audio signal, we can compute the Log-Mel Spectrogram (LMS), which is an important tool widely used for speech and audio processing [1], [5], [15], [16], [17], [18]. Given a signal with T temporal samples and F frequency bins, LMS can be represented as a 2D matrix $\mathbf{L}$ with size $K \times T$, computed as:

$$\mathbf{L} = \ln(\mathbf{H} \cdot \mathbf{S} + \epsilon). \tag{1}$$

$\mathbf{S}$ is a 2D matrix with size $F \times T$ containing the spectrogram of the audio signal (frequency information along rows and time information along columns), $\cdot$ computes the matrix multiplication, $\ln(\cdot)$ computes the natural logarithm and ϵ is a small constant used to avoid feeding zeros to the logarithm. The resulting LMS brings information about the spectral content of the audio signal (in Mel scale) as a function of the temporal evolution: along rows, we find pitches in Mel scale; along columns, the temporal evolution.

B. Time Scaling

Time scaling is used to control the temporal duration of a signal independently from its frequency behavior. The aim is to speed up or slow down the signal without changing its spectral content [6]. To provide a formal definition, we can employ the quasi-stationary model introduced by [19], [20]. A generic signal $s(t)$ is represented as the sum of $I(t)$ sinusoids with instantaneous frequency $\omega_i(t)$, phase $\phi_i(t)$, amplitude $A_i(t)$:

$$s(t) = \sum_{i=1}^{I(t)} A_i(t) e^{j\phi_i(t)}, \text{with } \phi_i(t) = \int_{-\infty}^{t} \omega_i(\tau) d\tau. \tag{2}$$

Time scaling can be seen as a mapping between time in the original signal and time in the modified one:

$$t \rightarrow t' = T_s(t) = \int_0^{\tau} \frac{\tau}{\alpha} d\tau, \tag{3}$$

where $T_s(\cdot)$ refers to the temporal mapping function and $\alpha > 0$ is the so-called time scaling factor. More specifically, α can be seen as the ratio between the duration of the original signal and the duration of the time scaled one. When $0 < \alpha < 1$, the signal is slowed-down, whereas, for $\alpha > 1$, the signal is speeded-up by means of time scale compression. If we refer to the sinusoidal model described in (2), the time scaled signal $s'(t')$ can be described as:

$$s'(t') = \sum_{i=1}^{I(T_s^{-1}(t'))} A_i(T_s^{-1}(t')) e^{j\phi_i'(t')}, \text{with } \phi_i'(t') = \int_{-\infty}^{t'} \omega_i(T_s^{-1}(\tau)) d\tau. \tag{4}$$

From (4), notice that the amplitude of the i-th sinusoid of the time scaled signal at the instant t' is exactly equal to the amplitude of the same sinusoid of the original signal at time $t = T_s^{-1}(t')$. In the same way, the instantaneous frequency at time t' in the original signal is the same of the original one at time $t = T_s^{-1}(t')$. This means the time evolution has changed, while the frequency content has not. This operation cannot be obtained through simple resampling which scales the signal in time but affects its frequency behaviour, introducing undesired artifacts and leaving traces that can be easily spotted. Different time scaling algorithms have been proposed, from frequency domain methods (e.g., phase vocoder [21] and sinusoidal spectral modeling [22]) to time-domain methods (e.g., those based on overlap and add [23] and synthesis frames [24]).

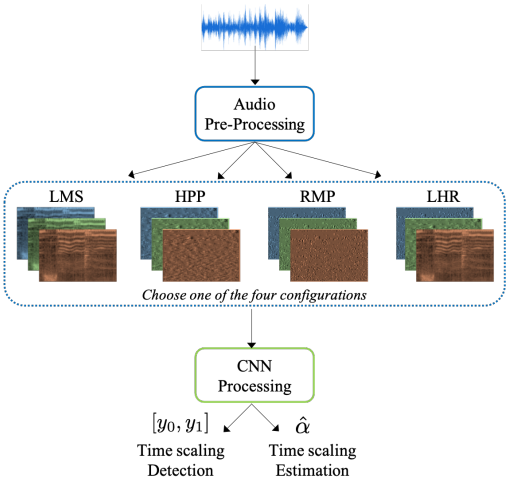

Fig. 1. A summary representation of the proposed time scaling detection and estimation system.

C. Problem Formulation

Our goal is to blindly identify the application of time scaling on an audio track. In particular, we have two main tasks:

1) Time scaling detection, where we aim at detecting whether the query audio signal is the original version or a time scaled one. In particular, we produce a discrete output $y \in [0, 1]$, in which 0 means the signal is original, and 1 means the signal is affected by time scaling.
2) Time scaling estimation, where the goal is to estimate the time scaling factor applied to the signal, i.e., to find an approximation $\hat{\alpha} \in \mathbb{R}^+$ of the real α. If time scaling has not been applied, the returned $\hat{\alpha}$ should be 1.

III. Proposed Methodology

We propose to solve our problems by feeding one CNN with pre-processed input audio data. A summary representation of the proposed system is shown in Fig. 1. Our methodology includes two main steps:

1) audio content pre-processing, in order to manipulate data prior to feed them to a CNN;
2) data processing through one CNN, purposely modifying the final network layer in order to face the specific tackled problem, i.e., time scaling detection or estimation.

A. Audio Pre-Processing

As CNNs work with input 2D data, we must convert the information contained in one dimensional audio tracks into 2D matrices. In particular, we propose to extract the time-frequency information of audio signals by investigating both the magnitude and phase of their STFT. Given an audio signal, we extract three different 2D representations:

- the LMS, defined in (1), which carries the information on the STFT magnitude (i.e., the spectrogram);
- a high-pass version of the unwrapped phase of the STFT obtained by filtering the unwrapped phase through a 2^{nd}-order Butterworth filter [25]. We define this as High-Pass unwrapped Phase (HPP);
- a residual median-filtered version of the STFT unwrapped phase, defined as Residual Median unwrapped Phase (RMP).

The reason to adopt the two aforementioned filtering-based approaches is to filter-out low frequency information and to enhance the high-frequency variations. Indeed, we aim to investigate the possibility that time scaling leaves peculiar but inaudible high-frequency phase variations.

All the three representations are derived from the signal STFT. Specifically, the LMS size depends on the amount of K distinct filters and T temporal samples used to compute it as shown in (1). The size of HPP and RMP depends only on the parameters used to compute the STFT. Referring to Section II for further details, given an input signal analyzed in F different frequencies and T time instants, both HPP and RMP have size $F \times T$. We propose to exploit the same size for all the applied pre-processings. Therefore, we select a number of frequencies F equal to the number of K Mel bins used to compute the audio LMS, by keeping only the frequency bins associated with the center frequencies of the Mel filterbank.

Once converted the audio data into 2D content, a further step is needed prior to feed the CNN. Indeed, the proposed CNN (described in detail in Section III-B) requires 3-channel images (i.e., color images) as input. We propose to arrange the three different data representations in two possible ways: (i) replicating one single representation three times; (ii) combining the three representations together, by concatenating them along the channel dimension. We define this configuration as LHR, namely the concatenation of LMS, HPP and RMP.

B. CNN Data Processing

Our network architecture is based on EfficientNet family of models [26], whose main peculiarity is to uniformly scale its dimensions by means of a compound scaling factor. This scaling process starts from the baseline architecture, namely EfficientNet-B0, which has achieved very good results, comparable with networks requiring much more parameters, both in computer vision and multimedia forensics [27].

In the light of this, we propose EfficientNet-B0 as backbone for our network architecture. Given a 2D input with three channels (as explained in Section III-A), EfficientNetB0 extracts 1280 scalar features. Finally, we apply a linear transformation to the feature vector, such that the output of the CNN counts n_{out} elements. The parameter n_{out} differentiates the CNN used for time scaling detection from the CNN used for time scaling estimation.

Time Scaling Detection. In time scaling detection, $n_{\text{out}} = 2$, since we aim at identifying original audio tracks from time scaled ones. We define the CNN output as $\mathbf{y}$, which is a vector of two numbers $[y_0, y_1]$ associated with the probability of detecting the absence or presence of time scaling, respectively (probabilities are obtained by passing $\mathbf{y}$ through the Softmax function). The final decision taken on the audio track is related to the highest probability among y_0 and y_1.

Time Scaling Estimation. In case of time scaling estimation, instead, $n_{\text{out}} = 1$. The output $\hat{\alpha}$ is a scalar number, and it is the CNN prediction of the time scaling factor α.

IV. Experiments and Results

A. Datasets

We select almost 4000 original audio tracks from two well known and widely used datasets in state-of-the-art:

- the GTZAN genre recognition dataset, created for the research described in [12]. This dataset is composed by 1000 mono tracks, each of which 30 seconds long, and is equally divided into 10 music genres.
- the IRMAS Testing Dataset, designed for instrument recognition in musical audio signals [13]. It is composed by 2874 excerpts with a length between 5 and 20 seconds.

To compare diverse time scaling implementations and test the generalization of the proposed method, we generate time scaling modified tracks using three different algorithms for phase vocoder time scaling modification: (i) the time scaling algorithm implemented by the Audiotsm library [28]; (ii) the TimeStretch algorithm proposed by Torchaudio [29]; (iii) the time scaling implementation provided in [30]. We always train and/or test the proposed methodology by keeping separated the audio files modified with these three distinct implementations, generating three different datasets.

We group the available original audio files into separate subsets for training and evaluation stages, randomly selecting the 85% of the files for the training phase (further divided into 80% for the training set and 20% for the validation one) and leaving the remaining ones to evaluation. In training and validation phases, we can apply time scaling modifications by selecting time scaling factors from a minimum $\alpha = 0.1$ to a maximum $\alpha = 1.9$, with step size equal to 0.2. In evaluation phases, audio tracks can be modified by exploiting also time scaling factors unknown at training stage. Testing time scaling factors can be selected from a minimum $\alpha = 0.1$ to a maximum $\alpha = 1.9$, with step size equal to 0.1.

The final datasets are built in different ways according to the tackled task, i.e., time scaling detection or time scaling estimation. For time scaling detection, each audio track is modified by randomly picking (following a uniform distribution) only one time scaling factor among the available ones. In this vein, the dataset size consists of twice the original

dataset dimensions, both for training and testing stages (i.e., the dataset includes the original data and one modified version per audio track). For time scaling estimation, each audio track is modified with all time scaling factors, thus resulting in a much larger dataset with respect to the detection goal. Contrary to time scaling detection, the dataset dimensions vary according to the specific experimental phase: at training stage, the dataset is 11 times larger than the original one (i.e., together with the original files, audios are modified with 10 time scaling factors); at evaluation stage, the dataset is 19 times larger, and the case $\alpha = 1$ corresponds to absence of modifications on the original files.

After editing the audio files, for each track we compute the LMS, the HPP and the RMP, i.e., the three data representations proposed. These are based on the computation of the signal STFT, which we implement as suggested in [15]. We consider Hanning windows with length $25\mathrm{ms}$ and hop size $10\mathrm{ms}$. The investigated frequency range spans from a minimum of $125\mathrm{Hz}$ to a maximum of $7.5\mathrm{KHz}$, with a sampling rate of $16\mathrm{kHz}$. We compute STFTs with 512 frequency samples and variable temporal dimensions according to the input audio length. As suggested in [15], we select only 64 Mel bins to compute the LMS, and keep only the 64 frequency bins associated to the center frequencies of the Mel filterbank for cropping HPP and RMP. Then, for each audio track, we select 4 non-overlapped time windows to increase the overall dataset dimensions. The selected windows start from the 4-th to the 7-th temporal bin of the STFT and count 96 temporal bins each.

To summarize, we feed our CNN with 64×96 patches, replicated three times over the channel dimension or concatenated among them (i.e., ending up with the previously defined LHR configuration) to obtain input "color" patches. In the evaluation phase, for each data descriptor (i.e., LMS, HPP, RMP, or LHR) and each time scaling algorithm implementation (i.e., choosing among those proposed in [28], [29], [30]), we have more than 4600 test patches for the detection task and more than 44000 test patches for the estimation task.

B. Experimental Setup

Following a common procedure in CNN training, we initialize the network weights using those trained on the ImageNet database [31]. In training phase, we update the weights according to two different loss functions. For time scaling detection, we adopt the binary cross entropy loss, while for time scaling estimation we use the mean square error. The optimization algorithm is the Adam optimizer with standard parameters.The initial learning rate is 0.001, reduced by a factor 10 whenever the validation loss does not improve for 10 epochs. We stop the training process if the validation loss does not decrease for 20 consecutive epochs; in any case, we consider a maximum number of epochs equal to 80. After the training, the model providing the best validation loss is selected.

Concerning the evaluation metrics, we use the average accuracy of correct predictions for time scaling detection. We evaluate time scaling estimation by means of the Pearson Correlation Coefficient (PCC) between the estimated and the original time scaling factors. In both the two cases, the higher the metrics (ideally approaching 1), the better the performance.

TABLE I
TIME SCALING DETECTION ACCURACY, CONSIDERING AUDIO FILES MODIFIED WITH [28], [29], [30]. IN BOLD, THE HIGHEST ACCURACY PER ALGORITHM.

CNN input	[28]	[29]	[30]
LMS	**0.8585**	**0.8631**	**0.8671**
HPP	0.6273	0.6755	0.6801
RMP	0.5998	0.6617	0.6522
LHR	0.7162	0.7259	0.7136

TABLE II
TIME SCALING DETECTION ACCURACY, CONSIDERING AUDIO FILES MODIFIED WITH [28], [29], [30], AND USING LMS AS INPUT TO THE CNN. IN BOLD, THE MINIMUM AND MAXIMUM ACCURACIES.

Train \ Test	[28]	[29]	[30]
[28]	0.8585	0.8569	0.8558
[29]	**0.8516**	0.8631	0.8526
[30]	0.8542	**0.8690**	0.8671

C. Time Scaling Detection

We report in Table I the achieved evaluation accuracies in case we train and test our method over audio files modified with the same time scaling algorithm, and evaluating all four data descriptors. It is worth noticing that the three different time scaling implementations achieve similar results, on average. This behaviour demonstrates the effectiveness of the proposed methodology in spotting traces of different time scaling implementations. Moreover, notice that LMS significantly outperforms the other data descriptors. Indeed, we might have expected this result, as it has been shown several times that LMS provides a valuable feature for audio processing and classification [5], [1], [15], [16], [17]. The LHR solution achieves worse results than LMS, even though being more accurate than phase-based strategies. Nonetheless, phase-based methods achieve accuracies up to 0.68, meaning for the presence of time scaling artifacts in the STFT phase as well.

We also experiment cross test scenarios, i.e., in which training and testing data do not come from the same time scaling algorithm. For brevity's sake, we only show results achieved with the representation providing the best results in non-cross test scenario, i.e., the LMS. Table II depicts the results, that are pretty similar to each other and are all located in the range of $0.85 - 0.86$. This means we can generalize well on different algorithms and we are not overfitting over the specific implementation seen at training stage.

D. Time Scaling Estimation

Results of time scaling estimation are depicted in Table III. We are considering a restricted evaluation set: only audio files modified with time scaling factors previously seen at training stage are selected. This situation represents a straightforward scenario, thus it can provide a baseline for evaluating more

TABLE III
TIME SCALING ESTIMATION PCC, CONSIDERING AUDIO FILES MODIFIED WITH [28], [29], [30] AND TESTING KNOWN TIME SCALING FACTORS. IN BOLD, THE HIGHEST ACCURACY PER ALGORITHM.

CNN input	[28]	[29]	[30]
LMS	**0.9391**	**0.9391**	**0.9375**
HPP	0.8120	0.8460	0.8365
RMP	0.7455	0.8098	0.8248
LHR	0.9006	0.9167	0.9146

TABLE IV
TIME SCALING ESTIMATION PCC, CONSIDERING AUDIO FILES MODIFIED WITH [28], [29], [30] AND EXPLOITING LMS AS CNN INPUT. WE TEST KNOWN (K) TIME SCALING FACTORS AND UNKNOWN (U) ONES.

Train \ Test	[28]-K	[29]-K	[30]-K	[28]-U	[29]-U	[30]-U
[28]	0.9391	0.9015	0.8998	0.9164	0.8784	0.8785
[29]	0.7728	0.9391	0.9385	0.7965	0.9275	0.9275
[30]	0.7809	0.9376	0.9375	0.7846	0.9221	0.9222

TABLE V
MULTIPLE-WINDOWS RESULTS (PCC) FOR TIME SCALING ESTIMATION, CONSIDERING AUDIO FILES MODIFIED WITH [28], [29], [30] AND EXPLOITING LMS AS CNN INPUT. WE TEST KNOWN (K) TIME SCALING FACTORS AND UNKNOWN (U) ONES.

Train \ Test	[28]-K	[29]-K	[30]-K	[28]-U	[29]-U	[30]-U
[28]	0.9737	0.9506	0.9507	0.9623	0.9397	0.9397
[29]	0.8593	0.9722	0.9720	0.8831	0.9664	0.9664
[30]	0.8662	0.9738	0.9737	0.8682	0.9670	0.9670

challenging experimental setups. Results are in line with those reported for detection. Exploiting the LMS is the best strategy, followed by LHR and subsequently by phase-based solutions.

We further deepen our investigations by evaluating the performances in cross test scenarios (i.e., considering different time scaling algorithms for training and testing sets) and by testing audio files modified with unknown time scaling factors at training phase. Table IV depicts the results achieved by LMS as input to the CNN. Precisely, the suffix K refers to known time scaling factors at training phase, while the suffix U refers to unknown ones. It is worth noticing that the proposed time scaling estimation strategy is robust both to cross test scenarios and unknown time scaling factors. Focusing on the first three columns of Table IV, we can evaluate cross test results in case of time scaling factors known at training stage. All the results are similar, and the only values under 0.9 are represented by training on data modified with [29] and [30] and testing on data coming from [28]. This behaviour is followed by the last three columns of Table IV, in which we show cross test results for scaling factors unknown at training stage. On average, this challenging situation can only cause a performance loss limited to $0.02 - 0.03$, highlighting the effectiveness and the robustness of the proposed method.

Furthermore, for the time scaling estimation task, we propose a slightly different evaluation approach, which allows

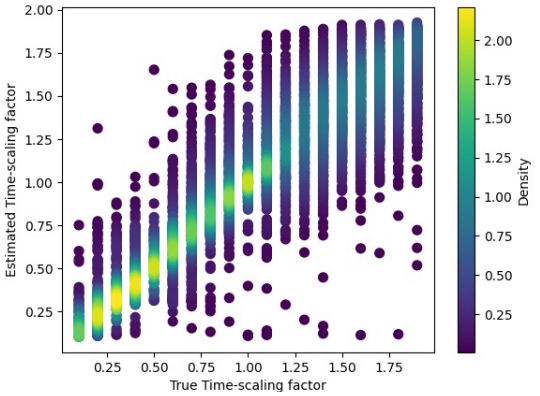

Fig. 2. Single-window time scaling estimation results by exploiting LMS as CNN input, considering audio files modified with [28].

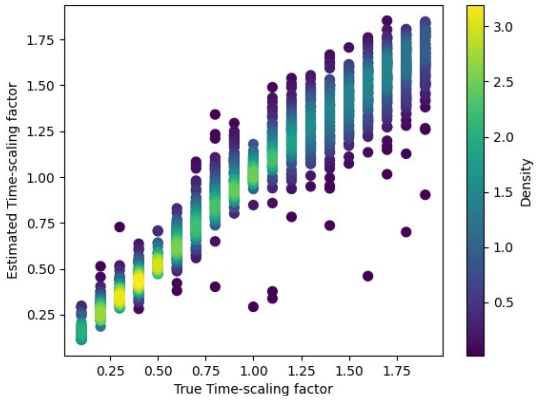

Fig. 3. Multiple-windows time scaling estimation results by exploiting LMS as CNN input, considering audio files modified with [28].

to achieve even more accurate results. Instead of testing the query audio file considering only one temporal window at a time (i.e., only 96 temporal samples selected from the signal STFT), we explore the possibility to gather information from the entire temporal evolution of the signal. Specifically, we apply the previously-trained CNN to all the subsequent non-overlapped temporal windows extracted from each audio track, ending up with M patches with size 64×96 per audio file. Then, we propose to reduce the predictions related to the M temporal windows to a single value which estimates the actual time scaling factor. In particular, we propose to compute the prediction $\hat{\alpha}$ related to the audio file as the arithmetic mean between the M diverse estimations associated with the single windows. To make a direct comparison with the single-window results shown in Table IV, we report in Table V the multiple-windows performance achieved by LMS. On average, PCC is 0.05 larger than PCC in the single-window modality.

We compare again the single-window and the multiple-windows strategies in Figs. 2 and 3, respectively. Both the figures depict the scatter plot computed between the estimated time scaling factors and the original ones. The CNN input is the LMS and data are generated with [28]. At first glance, we notice that the estimation density in the multiple-windows

scenario is much more focused around true scaling factors, meaning for a better estimation. Secondly, from these figures it emerges an interesting insight: on average, the predictions for $\alpha > 1$ (i.e., when the modified signal has shorter duration) exhibit a more prominent vertical dispersion than predictions for lower factors, which are more concentrated near the correct value. We suppose time scaling with $\alpha < 1$ (i.e., when the modified signal has longer duration) implies the application of some interpolation, with the introduction of artificial samples that can be relatively easy to detect. Time scaling with $\alpha > 1$ involves the rejection of some samples, making harder the recognition of artifacts in the modified audio signal.

V. Conclusions

This work proposes a CNN-based method for the blind detection and estimation of time scaling applied to audio signals. We propose to feed our CNN with different representations derived from the audio track: (i) the Log-Mel Spectrogram; (ii) a high-pass filtered version of the unwrapped phase of the signal STFT; (iii) a median filtered version of the STFT unwrapped phase; (iv) the concatenation of the three reported representations. We select audio files from two well-known datasets in state-of-the-art. Then, we modify the signals by means of three different time scaling implementations.

To evaluate the robustness of the proposed methodology, we consider cross test scenarios in which training and testing data come from diverse time scaling implementations. Moreover, we evaluate the method performance in estimating unknown time scaling factors, never seen during training phase. In both these challenging situations, the proposed methodology demonstrates its effectiveness and achieves valuable results.

We also investigate the method performance in aggregating multiple results related to various time windows extracted from the query audio file. In doing so, results for time scaling estimation are further improved, however highlighting the need for future investigations in case time scaling is applied to shorten the signal duration.

References

[1] M. Mascia, A. Canclini, F. Antonacci, M. Tagliasacchi, A. Sarti, and S. Tubaro, "Forensic and anti-forensic analysis of indoor/outdoor classifiers based on acoustic clues," in *European Signal Processing Conference (EUSIPCO)*, 2015.

[2] Q. Yan, R. Yang, and J. Huang, "Copy-move detection of audio recording with pitch similarity," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[3] D. Luo, M. Sun, and J. Huang, "Audio postprocessing detection based on amplitude cooccurrence vector feature," *IEEE Signal Processing Letters*, vol. 23, pp. 688–692, 2016.

[4] Y. Zhan and X. Yuan, "Audio post-processing detection and identification based on audio features," in *International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, 2017.

[5] B. Liang, G. Fazekas, and M. Sandler, "Piano sustain-pedal detection using convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[6] M. Kahrs and K. Brandenburg, *Applications of Digital Signal Processing to Audio and Acoustics*. USA: Kluwer Academic Publishers, 1998.

[7] D. Harwell, "Faked pelosi videos, slowed to make her appear drunk, spread across social media," https://www.washingtonpost.com/technology/2019/05/23/faked-pelosi-videos-slowed-make-her-appear-drunk-spread-across-social-media/, 24 May 2019, accessed: 2021-07-26.

[8] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of resampling," *IEEE Trans. Signal Process.*, vol. 53, no. 2-2, pp. 758–767, 2005. [Online]. Available: https://doi.org/10.1109/TSP.2004.839932(410)53

[9] M. Kirchner, "Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue," in *Proceedings of the 10th ACM workshop on Multimedia and security*, 2008, pp. 11–20.

[10] D. Vázquez-Padín and F. Pérez-González, "Prefilter design for forensic resampling estimation," in *2011 IEEE International Workshop on Information Forensics and Security*. IEEE, 2011, pp. 1–6.

[11] Z. W. D. Y. R. Wang, L. Xiang, and T. Wu, "Speech resampling detection based on inconsistency of band energy," *Computers, Materials & Continua*, vol. 56, pp. 247–259, 2018.

[12] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 293–302, 2002.

[13] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[14] S. S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology*, vol. 53, pp. 329–353, 1940.

[15] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[16] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural TTS synthesis by conditioning Wavenet on mel spectrogram predictions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[17] H. Meng, T. Yan, F. Yuan, and H. Wei, "Speech emotion recognition from 3D log-mel spectrograms with deep learning network," *IEEE access*, vol. 7, pp. 125 868–125 881, 2019.

[18] L. Comanducci, P. Bestagini, M. Tagliasacchi, A. Sarti, and S. Tubaro, "Reconstructing speech from CNN embeddings," *IEEE Signal Processing Letters*, 2021.

[19] L. Almeida and F. Silva, "Variable-frequency synthesis: An improved harmonic coding scheme," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1984.

[20] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 744–754, 1986.

[21] J. L. Flanagan and R. M. Golden, "Phase vocoder," *The Bell System Technical Journal*, vol. 45, pp. 1493–1509, 1966.

[22] T. Quatieri and R. McAulay, "Speech transformations based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 1449–1464, 1986.

[23] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1993.

[24] D. Malah, "Time-domain algorithms for harmonic bandwidth reduction and time scaling of speech signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, pp. 121–133, 1979.

[25] "Scipy butterworth filter," https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html, accessed: 2021-02-27.

[26] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019.

[27] S. Mandelli, N. Bonettini, P. Bestagini, and S. Tubaro, "Training cnns in presence of JPEG compression: Multimedia forensics vs computer vision," in *IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2020.

[28] Muges, "Audiotsm phase vocoder," https://github.com/Muges/audiotsm, Oct. 2017, accessed: 2021-02-21.

[29] Pytorch, "Time stretch," https://pytorch.org/audio/stable/_modules/torchaudio/transforms.html#TimeStretch, accessed: 2021-02-21.

[30] "Time stretch master," https://github.com/gaganbahga/time_stretch, Dec. 2020, accessed: 2021-02-22.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.