



# Recurrent neural network and long short-term memory models for audio copy-move forgery detection: a comprehensive study

Fulya Akdeniz<sup>1,2</sup> · Yaşar Becerikli<sup>1</sup>

Accepted: 3 February 2024  
© The Author(s) 2024

## Abstract

One of the most pressing challenges in audio forgery detection—a major topic of signal analysis and digital forensics research—is detecting copy-move forgery in audio data. Because audio data are used in numerous sectors, including security, but increasingly tampered with and manipulated, studies dedicated to detecting forgery and verifying voice data have intensified in recent years. In our study, 2189 fake audio files were produced from 2189 audio recordings on the TIMIT corpus, for a total of 4378 audio files. After the 4378 files were preprocessed to detect silent and unsilent regions in the signals, a Mel-frequency-based hybrid feature data set was obtained from the 4378 files. Next, RNN and LSTM deep learning models were applied to detect audio forgery in the data set in four experimental setups—two with RNN and two with LSTM—using the AdaGrad and AdaDelta optimizer algorithms to identify the optimum solution in the unlinear systems and minimize the loss rate. When the experimental results were compared, the accuracy rate of detecting forgery in the hybrid feature data was 76.03%, and the hybrid model, in which the features are used together, demonstrated high accuracy even with small batch sizes. This article thus reports the first-ever use of RNN and LSTM deep learning models to detect audio copy-move forgery. Moreover, because the proposed method does not require adjusting threshold values, the resulting system is more robust than other systems described in the literature.

**Keywords** Digital multimedia forensics · Audio forensics · Audio tampering detection · Audio copy-move forgery detection · RNN · LSTM · Digital multimedia security

---

✉ Fulya Akdeniz  
fulyaakdeniz@gmail.com; fulya.akdeniz@kocaeli.edu.tr  
Yaşar Becerikli  
ybecerikli@kocaeli.edu.tr

<sup>1</sup> Department of Computer Science, Kocaeli University, Kocaeli, Türkiye

<sup>2</sup> Department of Computer Science, Recep Tayyip Erdoğan University, Rize, Türkiye

## 1 Introduction

Biometric data, obtained using various sensors and devices to capture people's physical and behavioral characteristics, are personal but essential data as they identify specific individuals [1]. Indicating the personal identity, age, gender, and even emotional state of individuals, biometric data most often derive from data representing people's faces, fingerprints, voices, hand geometry, irises, and signatures [2]. Voice and speech data, some of the most accessible and thus widely used biometric data, can be obtained without requiring contact with sensors and offer high mobility. Not only is speech the easiest, most natural means of communication between people [3], but voice signals are also more suitable than other biometric data because they can be authenticated conveniently, quickly, and reliably [4].

Face-based and voice data can be easily accessed due to the widespread use of the Internet and digital multimedia resources. Because tools for manipulating audio and video are also readily accessible online as well as easy to use, manipulating such data has become rather uncomplicated [5, 6], and the volume of manipulated records, whether generated with good or bad intentions, has ballooned. In response, recent studies on the detection of manipulated images have generated models for detecting such manipulation using various methods. Nevertheless, as researchers have that despite momentum in research on audio manipulation, such research remains in its infancy [7, 8]. Studies on detecting voice manipulation and voice copy-move forgery, for example, have been very few, and the forgery detection systems developed for those purposes have performed detection only under highly limited conditions. However, such studies are pivotal, for authenticating and verifying audio manipulation is not only far more difficult than authenticating and verifying digital images [5–9] but also demands the accurate, reliable analysis of audio signals.

The development of multimedia technology has driven a rapid increase in speech data, which have therefore become important parts of multimedia data [10]. Because those data are used directly in military commands, legal proceedings, banking systems, Internet of Things devices, smart speakers, automatic speaker verification systems, and copyright in the music industry, the protection and verification of the integrity of digital conversation records is critical for information security in those sectors [5]. To support such efforts, the emerging science of audio forensics involves analyzing audio recordings used in courts and other institutions [11]. The first audio record was forensically examined in the 1950s, and, in the 1990s, studies in audio forensics began to emerge [12]. Although analog audio data were used in the first forensics authentication systems, digital audio data are used today. The process of detecting the manipulations/forgery of digital audio data, however, remains extremely difficult, largely because such manipulation leaves no traces that can be detected visually or audibly [13].

In the literature, audio forensics is divided into two categories depending on whether active or passive techniques are used. Whereas active techniques rely on

private information such as watermarks, hash values, and digital signatures, passive techniques can process audio data that do not contain any additional information and passive techniques is quite difficult to detect than active techniques, since they do not contain extra information in the passive technique. Furthermore, when audio data are examined in real conditions, locating digital watermarks and signatures is not always possible, which gives passive techniques a wider area of application. For those reasons, passive forensics techniques are pivotal in audio forensics [14–18]. In our study, using the passive technique of audio copy-move forgery detection, we sought to detect manipulated audio files. The audio copy-move technique is a form of audio manipulation that involves copying a region from an audio recording and moving it to a different location in the same recording. Because copying and moving processes are performed on the same audio file [19], detecting that manipulation technique ranks among the most difficult problems in the literature, as various researchers have emphasized, and studies on and knowledge about the problem are greatly needed [7, 8, 14, 19–22].

Against that backdrop, in our study we aimed to develop a method of audio copy-move forgery detection. To that end, we developed recurrent neural network (RNN) and long short-term memory (LSTM) models that we tested with different hyperparameters to obtain the optimum result. The results of our tests are compared in this article, supported by tables and figures detailing accuracy and loss rates.

As described in this article, our study was the first to use RNN and LSTM sequence models to detect audio copy-move forgery. It was also the first to use Mel-frequency cepstral coefficients (MFCCs), delta ( $\Delta$ ) MFCCs, and  $\Delta\Delta$ MFCCs together as a fusion model. Moreover, because most of the few studies on audio copy-move forgery detection have involved using a threshold value, the decision-making process in those studies has depended on various factors such as age, gender, and speech environment. In our study, however, audio copy-move forgery detection was performed without using any threshold value in decision-making, which reduced dependence on the person, gender, audio recording, and environmental conditions. Beyond that, our study was the first comprehensive study to examine the effect of hyperparameter variation, optimization algorithm, batch size, and epoch on sequence-based deep learning models. Last, as the results of our experiment show, the proposed methods allow the detection of original and forged voices in audio copy-move forgery without using any threshold value. Figure 1 presents a graphical abstract of the proposed system.

In what follows, Sect. 2 reports relevant literature on copy move forgery detection, Sect. 3 presents the materials and methods that we employed in our



Fig. 1 Graphical Abstract

study, and Sect. 4 describes the database that we developed, our experimental setups, and the experimental results. Section 5 discusses our proposed model, after which Sect. 6 articulates our conclusions.

## 2 Related literature

The dearth of literature on audio copy-move forgery detection underscores the need for additional studies on the topic [7, 8, 14, 19, 20, 22], as does the lack of any robust, integrated system for detecting the copy-move forgery of audio data. Indeed, the measured authenticity and accuracy of audio data frequently is used, highly sought properties in verification systems. For that reason, researchers have developed various models for detecting audio copy-move forgery in such systems, often using various feature extraction models. The most preferred model among them involves using the audio feature of pitch.

In that vein, Li et al. developed a method of detecting copy-move forgery in audio signals in four stages: preprocessing, pitch sequence extraction, difference calculation, and comparison. First, in preprocessing, the audio signals were processed nonlinearly, after which pitch sequences were extracted by using the pitch frequency feature of the audio signals to obtain the signals' features. In the third and fourth stages, the differences between the syllables were calculated using a threshold value and compared to determine whether copy-move forgery was used in the audio signals. During analysis, the method was tested by applying six types of attacks to 500 audio-recording data. The recall values calculated all exceeded 90%, with no attack at 93.55%, compressed amplitude at 91.22%, 15 dB of added noise at 92.35%, 5 dB of added noise at 92.56%, 48 kHz up-sampling at 91.89%, 16 kHz down-sampling at 91.89%, and mp3 coding at 92.05% [20].

By contrast, Yan et al. divided audio signals into voiced and unvoiced segments using the yet another algorithm for pitch tracking (YAAPT) method. After the pitch sequence and the first two formant sequences were obtained in feature extraction from the voiced segment, the similarities of each feature set were calculated with the dynamic time warping algorithm. Last, using a threshold value, whether copy-move forgery was used in the audio recordings was determined. They tested their method with data from *The Wall Street Journal* and TIMIT speech databases using 20,000 duplicated segments of data tested with 16 types of attacks. Without any attack, the accuracy rate was calculated to be 99.25% and the recall value to be 100%. However, after the various attacks, the accuracy rates and recall values varied from 76.34 to 98.89% [7].

By further contrast, Yan et al. developed a two-stage method involving pitch extraction and the comparison of pitch sequences. First, the voiced and unvoiced segments were extracted from audio signals using the YAAPT method. After pitch sequences obtained from each audio segment of the audio signals were used for feature extraction, Pearson correlation coefficients (PCCs) were used to calculate the similarities between the pitch features. A threshold value for gauging the similarity between the audio segments was calculated to use in determining whether the audio signals have been tampered with. Their method was tested with three types

of attacks and 1,000 tampered audio recordings generated by the authors. Next, 3,000 post-processed tampered audio recordings were obtained by applying post-processing to those recordings, and the true positive rate (TPR) was calculated to assess the performance of the method. The TPR rate was 99.962% without any attack, 98.514% after noise was added, 99.398% after filtering, and 99.784% after recompression [22].

In other work, Xie et al. developed a multi-feature decision-making system for detecting copy-move forgery in audio signals using the signals' pitch features in a three-stage method: preprocessing, feature extraction, and decision-making. In preprocessing, variance-based, double-threshold voice activity detection was applied. Next, in feature extraction, four features were obtained: the pitch feature of the audio, gammatone features, MFCCs, and discrete Fourier transform (DFT) coefficients. The similarities of features were calculated by determining a definite threshold value with PCCs and average difference methods. Last, in decision-making, the C4.5 algorithm was used to feature set. The system was tested on 1000 original and 1000 forged audio data with 10 dB Gaussian noise added to the tampered data. The best accuracy rate was calculated to be 99.0% without any attack and 94.1% after noise was added [8].

By comparison, Huang et al. divided their method of detecting audio copy-move forgery into three steps. First, the audio signals were split into segments, after which DFT coefficients were applied to each segment for feature extraction. Last, each segment was ranked according to its characteristics, and audio copy-move forgery was detected by comparing the segments in a sorted list [10].

In a similar method, Liu et al. first split audio signals into syllables to segment them. After DFT transformation was applied to each segment for feature extraction, each segment was ranked according to attributes. Afterward, a threshold value was determined, and audio copy-move forgery was detected by examining the similarity of the characteristics of the sequenced segments, calculated using PCCs. Next, 1000 forged voice data were generated, and the results of the system were tested by adding Gaussian noise to those data. The accuracy rate, recall rate, and F1 score of copy-move forgery detection without any attack applied were 100, 98.9, and 99.4%, respectively, and after noise was added were 100, 97.6, and 98.8%, also, respectively [19].

Meanwhile, Wang et al. segmented audio signals and applied a voice activity detection algorithm to the signals. Discrete cosine transform (DCT) coefficients were obtained by applying the DCT method to the segments for feature extraction, after which the coefficients of each segment were converted into a square matrix. To adjust a singular eigenvector, a singular value decomposition (SVD) transformation was applied to the matrix. Last, the forged audio signals were identified by calculating the distance between any two singular vectors [14].

Xiao et al. developed a method of detecting audio copy-move forgery based on calculating the similarity between two segments in audio signals. Once a threshold value was determined, it was compared with the similarity values in order to calculate the similarity between segments, namely using fast convolution algorithm [12].

Last, Imran et al. used a voice activity detection algorithm to detect the presence of speech. After histograms of words were obtained with the one-dimensional local binary pattern algorithm for feature extraction, forged parts in the audio recordings with similar histograms were identified. Ultimately, the forged audio recordings were obtained using the Arabic Speech Corpus, and the accuracy rate was determined to be 96.59% [21].

### 3 Materials and methods

In our study, we designed and developed a first-ever end-to-end system based on RNN and LSTM models for detecting audio copy-move forgery. The proposed system consists of three stages: (1) Analysis and fragmentation of silent and unsilent segments of audio signals, (2) The extraction of Mel-frequency-based features from each segment, and (3) Audio copy-move forgery detection using RNN and LSTM models. Figure 2 presents a block diagram of the proposed system.

In the first stage of the proposed system, an audio signal is divided into voiced and unvoiced segments using the YAAPT for pitch tracking. In the second stage, after 13 MFCCs,  $\Delta$ MFCCs, and  $\Delta\Delta$ MFCCs are calculated from each segment, those coefficients are associated and used in a fusion feature vector. Last, in the third stage, the fusion feature vector provides input for the RNN and LSTM network models. In our study, we used different optimization algorithms, batch sizes, and epoch numbers to examine the optimum model.

#### 3.1 YAAPT

The robust algorithm for pitch tracking (RAPT) processing on the time domain performs the pitch-tracking process. Although the RAPT has achieved good results in many studies, one of its greatest disadvantages is frequent pitch doubling, which causes numerous pitch errors. In response, Zahorian et al. developed the YAAPT from RAPT in 2008 as a more powerful pitch-tracking method for performing frequency domain analysis [23, 24]. The YAAPT, a noise- and interference-resistant frequency pitch-tracking algorithm, works accurately and reliably for high-quality voice data and telephone conversations. Using both the time and frequency domain of audio signals, it estimates pitch frequencies by subtracting the local maxima of the normalized cross-correlation function of audio signals [25, 26].

One of the most important features of the YAAPT is its use of spectral information for the F0 tracking algorithm. Because F0 tracks obtained from the spectrograms of audio signals can better identify F0 candidates, spectral F0 tracks are produced with the peaks occurring at the fundamental frequency and



Fig. 2 Block diagram of the proposed system

the harmonics of those frequencies [27]. The YAAPT consists of four major stages: preprocessing, calculating the F0 track, predicting the F0 candidate, and determining the F0 [24]. First, in nonlinear preprocessing, multiple versions of a signal are developed using nonlinear processing. Second, in F0 track calculation from the spectrogram, F0 tracks are estimated from the nonlinear signal using spectral harmonic correlation and dynamic programming. Third, in F0 candidate estimation based on NFCC, using the spectral F0 traces estimated in the previous step, candidate F0s are extracted by optimizing both the signal and nonlinear processed signal. Fourth and finally, in determining F0s, an F0 trace is calculated using the dynamic programming technique.

### 3.2 MFCCs

MFCCs, representing the short-term power spectra of audio files on a nonlinear Mel scale, simulate the characteristics of the audience. MFCCs are a popular method frequently used in audio-processing studies involving speaker recognition, voice recognition, and emotion analysis [28]. MFCCs represent the cepstral parameters extracted in the Mel-scaled frequency domain, while the Mel scale itself reveals nonlinear characteristics of the frequency of the human ear. The greatest advantage of the Mel-frequency scale is its high accuracy rates even as the noise ratio increases [29–31]. Equation (1) can be used to convert the frequency value of a signal to the Mel-frequency scale [32, 33]:

$$f_{mel} = 1125 \ln \left( 1 + \frac{f}{700} \right) \tag{1}$$

in which  $f_{mel}$  is the Mel-scale frequency and  $f$  is the Hertz scale frequency.

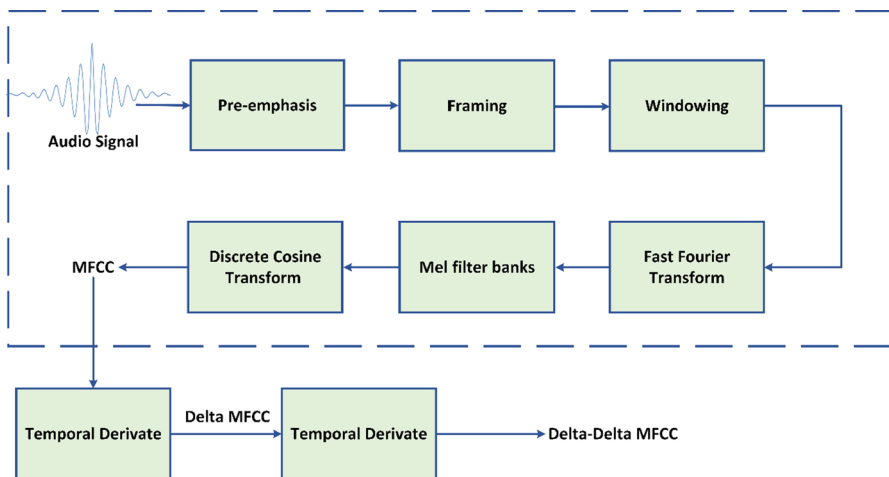


Fig. 3 Process of calculating MFCCs

Calculating MFCCs involves six steps—pre-emphasis, framing, windowing, fast Fourier transform (FFT), obtaining Mel filter banks, and DCT—as shown in Fig. 3 and detailed in the following subsections.

### 3.2.1 Pre-emphasis

The dual purpose of pre-emphasis, or preprocessing an audio signal using a high-pass filter, is to improve the signal's high-frequency components by making the signal more dominant and to increase the signal-to-noise ratio [28]. In our study, pre-emphasis was performed using the following equation:

$$y(t) = x(t) - \alpha x(t - 1) \quad (2)$$

in which  $x(t)$  is the audio signal,  $\alpha$  is a fixed value between 0, and  $y(t)$  is the new signal values.

### 3.2.2 Framing

Framing is the process of splitting an audio signal into brief parts. Because signal frequencies sustain changes, framing processes are needed to analyze the signal. The continuous variation of a signal's time domain causes the loss of spectral information during the FFT transformation of the entire signal. Therefore, it is advantageous to take each frame's FFT. When a short frame length is selected, it may not contain a sufficient number of samples in the frame length; however, when a large frame length is selected, it contains too many samples, which only complicates analysis. Thus, selecting a correct frame length is pivotal. Overlapping is also used in the framing process so that important information at the beginning and end of an audio signal is not lost [28, 34].

### 3.2.3 Windowing

After an audio recording's frames are divided, windowing is performed to calculate each frame's power spectrum [28]. Windowing serves to reduce the discontinuous regions that occur in the first and final parts of a signal after framing. With windowing process in the audio signals decided and applied correctly, the signal oscillations can be minimized, which boosts the signal's stability. Considering those processes, more efficient MFCC coefficients are obtained [35]. The windowing function that we used is shown in Eq. (3):

$$w_n = 0.54 \times 0.46 \cos\left(1 + \frac{2\pi n}{N - 1}\right) 1 \leq n \leq N \quad (3)$$

in which  $w_n$  is the windowing function and  $N$  is the window length.



### 3.2.4 FFT

Each frame of the separated audio signal is transformed from the time domain to the frequency domain by applying the FFT method [36], shown in Eq. (4):

$$s_n = \sum_{k=0}^{N-1} s_k e^{-\frac{j2\pi kn}{N}} \tag{4}$$

in which  $s_k$  is the input signal and  $N$  represents the frame length.

### 3.2.5 Mel-frequency scale

The Mel-frequency scale, a scaling system that simulates the hearing perception of the human ear, is used to obtain sounds similar to the sounds that people hear in an audio signal [37]. Mel filters consist of a triangular filter bank with a linear attribute up to 1000 Hz and a logarithmic attribute after 1000 Hz. Each filter is represented by a coefficient that indicates the energy of the signal in the band covered by the filter [36]. The relationship between linear frequency (in Hz) and the Mel scale is shown in Eq. (1).

### 3.2.6 DCT

After audio signals are passed through the Mel filter bank, the logarithm of signal is calculated. DCT is applied to the signal so that the logarithm Mel spectrum transforms the frequency domain to the time domain. At the end of DCT process, the Mel power spectrum coefficients are calculated using Eq. (5):

$$c_i = \sum_{j=1}^M m_j \times \cos\left(\frac{\pi \times i}{M}(j - 0.5)\right) \tag{5}$$

in which  $j$  is triangle filter,  $m_j$  is the logarithm of the energy obtained triangle filter, and  $M$  is the number of filter banks [35, 36].

### 3.3 ΔMFCCs

MFCCs represent the spectral characteristics of each frame analyzed in an audio signal. Along with static features, audio signals contain dynamic features obtained by subtracting the ΔMFCC and ΔΔMFCC features from the signals. The ΔMFCCs are determined using the first-order derivative of the Mel-frequency coefficients, while the ΔΔMFCCs are obtained using the second-degree derivative [32, 33, 38]. In turn, the ΔMFCCs represent the velocity information between consecutive frames, while the ΔΔMFCCs represent the acceleration properties between those frames, as shown in Eq. (6):

$$\text{delta}_t = \frac{\sum_{\tau=1}^N \tau(C_{t+\tau} - C_{t-\tau})}{2 * \sum_{\tau=1}^N \tau^2} \quad (6)$$

in which  $\text{delta}_t$  is the  $\Delta\text{MFCC}$  and  $C_t$  is the MFCC.

When MFCCs, efficiently representing the spectral features of each frame, are used for dynamic features extraction, the features in a speech signal are concentrated in the first coefficients. Because most features contained in the entire signal can be expressed by the coefficients, the features reduce complexity by representing a large amount of data with less data, as in speech recognition using MFCCs. In that case, using MFCC features has many advantages, including reduced complexity, time savings, and high performance. Thus, in our study, we used MFCC features in features extraction [39, 40].

### 3.4 RNN

An RNN, a neural network structure able to model time-sequential data such as audio signals [41], is a model capable of learning features and long-term dependencies from sequential and time-series data. Unlike the RNN's feedforward neural network, there is a recurrent connection between its nodes that the network structure uses to create a memory. That process allows mapping not only the input data in the network to the output but also all previous inputs to each output. Because of those structures, the state information produced by the feedforward neural network is stored and reapplied to the network with the input information. Those structures generate RNN models that can effectively learn sequential data [42–44].

The simple RNN model, shown in Fig. 4, consists of an input layer, recurrent hidden layers, and output layers. The RNN model uses the hidden state information from step  $t-1$  while processing the data in step  $t$ , as represented in Eq. (7):

$$h(t) = f_c(h(t-1), x(t)) \quad (7)$$

in which  $x(t)$  is the current input data,  $h(t-1)$  is the previous hidden state, and  $h(t)$  is the new state [41].

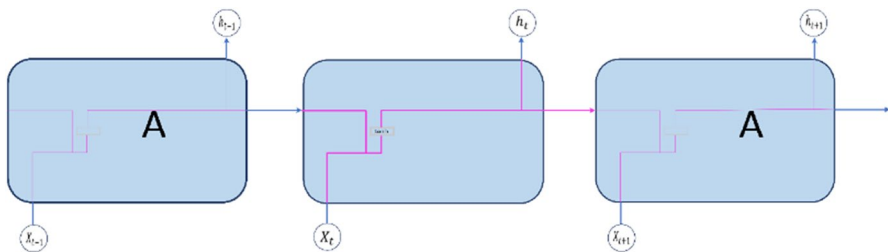


Fig. 4 The SimpleRNN model

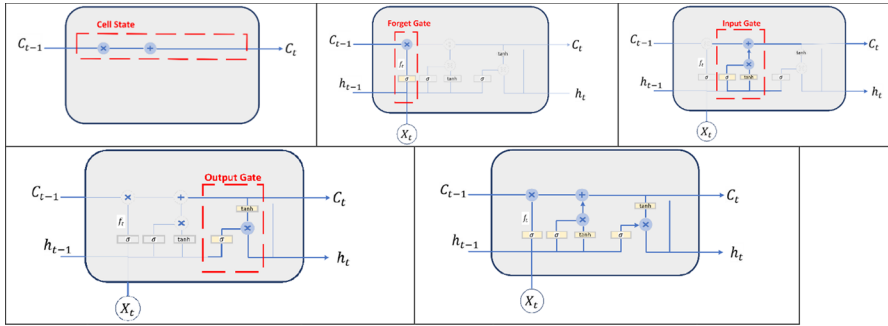


Fig. 5 LSTM network structures

### 3.5 LSTM

LSTM, an iterative recursive neural network model able to learn long-term dependencies, is one of the most popular deep learning models used for time-series predictions made by networks such as RNN. The essential distinction between LSTM and RNN models is that the former stores time-dependent information in the long term by mapping nonlinear relationships between inputs and outputs.

LSTM consists of three primary structures—the forget gate, the input gate, and the output gate—as shown in Fig. 5. Between those three structures, the input activation function vector is denoted as  $i_t$ , the output activation vector as  $o_t$ , and the forgetting activation vector as  $f_t$  [46]. Once the input gate decides to update cell states, the output gate decides which information to use at the output. Subsequently, the forget gate decides whether the information will be transferred and, if so, then how much is needed, namely by analyzing the input and previous state information. Equations (8)–(13) give LSTM gates formula:

$$f_t = \sigma_{LR}(W_f x_t + U_f h_{t-1} + b_f) \tag{8}$$

$$i_t = \sigma_{LR}(W_i x_t + U_i h_{t-1} + b_i) \tag{9}$$

$$O_t = \sigma_L(W_o x_t + U_o h_{t-1} + b_o) \tag{10}$$

$$c_t^* = W_c x_t + U_c h_{t-1} + b_c \tag{11}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \sigma_h(c_t^*) \tag{12}$$

$$h_t = O_t \otimes \sigma_h(c_t) \tag{13}$$

in which  $x_t$  represents the input,  $h_t$  represents the output,  $\sigma_{LR}$  (i.e., LeakyReLU) represents the activation functions, and  $W$  and  $U$  represent the weight matrix from Eq. (8) [45].

Cell state is a state vector information and is represented  $C_t$  and old cell vector state is represented  $C_{t-1}$ . Data information added and deleted from this state vector using the forget and input gates. Forget gate computes a 0 and 1 value using activation output function from the input is represented  $X_t$ , and  $h_t$  is represented the current hidden state. Information multiplicatively merged with cell state and “forget information” where the gate outputs update to 0. Input gate determines input node’s value counts added to the current memory cell internal state. Output gate computes a activation function of the input and current hidden state to determine which information of the cell state to “output.” Cell state is updated by using vector multiply to “forget” and vector addition to “input” new information.

Although successful results have been obtained for audio signals using RNNs, training RNN models remains difficult. When an RNN is trained with a gradient algorithm, its gradient is lost at an exponential rate as it performs iterative backpropagation to the same layer. In the literature, the problem is the “vanishing gradient problem” or the “exploding gradient problem.” The LSTM is a deep learning model developed to minimize that problem and can indeed mostly solve the problem [43]. LSTM and RNN models, when combined, are also known to be more effective than deep neural networks for acoustic modeling [42, 47].

### 3.6 Optimization algorithms

Optimization algorithms adjust the hyperparameters of models to minimize the loss function in the training phase of deep learning and machine learning models. The loss function is used to calculate the difference between the parameters’ actual and predicted values by the model. Because minimizing such loss is crucial, obtaining a smaller loss value is better for model predictions. The variation of the values of the hyperparameters of models and the optimum hyperparameter selection are calculated in the training phase. For that reason, the selection of hyperparameters is pivotal for minimizing the loss [48].

In optimization computing, weights and bias values are iteratively updated in models, which allows learning the information in the data of a model. The performance of deep learning models increases by means of optimization algorithms, while the accuracy of the models improves. Because a deep learning model may consist of hundreds of thousands or even millions of parameters, determining the best weights for the model is extremely difficult. To determine the optimum weights, the most appropriate optimization algorithm should be selected.

#### 3.6.1 Adaptive gradient descent (AdaGrad)

The adaptive gradient descent (AdaGrad) algorithm is a gradient-based learning algorithm that scales the learning rate of each parameter according to the previous

gradients calculated for that parameter. Unlike SGD and momentum methods, AdaGrad updates the learning rate for each iteration [49]. To determine the learning rate during an update in the training phase, the parameters of the model are examined. The algorithm gives a smaller learning rate to parameters with large gradient values and a larger learning rate to parameters with small gradient values. In other words, the more that the parameters change, the less that the learning rate changes. Depending on the problem, that dynamic can make the model learn better. Other important advantages of AdaGrad are that it helps to prevent the learning rate from decreasing too quickly depending on the parameter and weight values, allows the training process to converge faster, and allows the learning rate to be set automatically by the algorithm, not manually [50].

### 3.6.2 Adaptive delta (AdaDelta)

The adaptive delta (AdaDelta) algorithm was developed to solve the decreasing learning rate problem of AdaGrad. Unlike AdaGrad, AdaDelta does not take all of the squares of the previous gradients but the sum of the gradient squares in a certain proportion [49].

### 3.6.3 Batch size

Batch size is a hyperparameter that defines the number of samples to be used before the model parameters are updated. At the end of the batching process, the predicted values are compared with the expected output values, and an error is calculated [51]. In our study, aside from the different models used, the effects of the batch size and number of epochs on the sequence models were also investigated. Experimental results revealed a relationship between batch size and the number of epochs; when a high batch size was used, large numbers of epochs afforded more accurate results than small numbers of epochs.

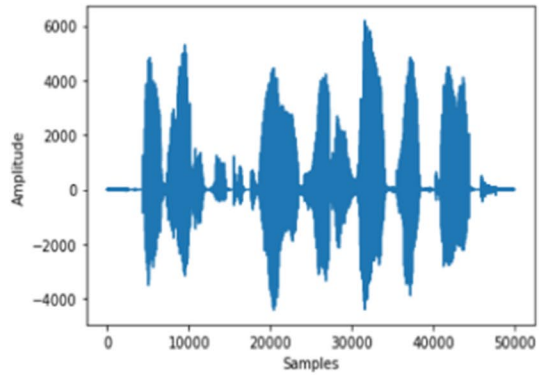
One of the chief hyperparameters for tuning is the batch size, specifically the number of signals used at each step to train the model. If the batch size chosen is too large, then it can take training too long to achieve convergence. By contrast, if the batch size is too small, then the training process ends without achieving good performance [52].

## 4 Experimental results

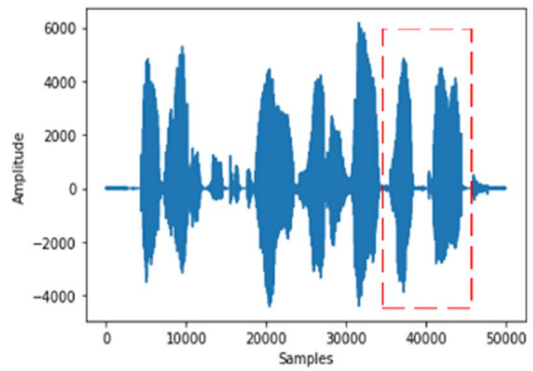
### 4.1 Data set

In our study, we used the TIMIT corpus—a widely used corpus in audio processing that represents 630 speakers of American English of different genders and dialects [53]—to produce forged audio files. We tested 2189 audio recordings, varying in length from 4 to 6 s, and produced 2,189 forged files of voice data by tampering

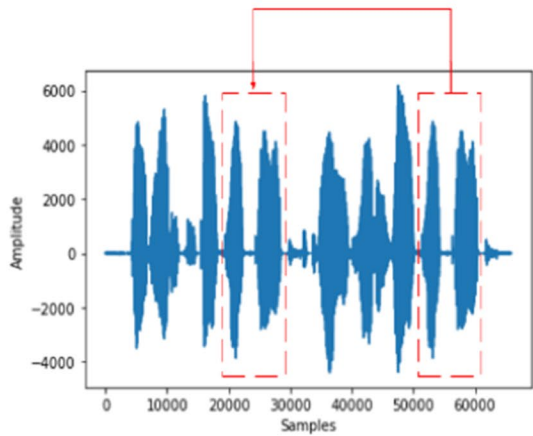
**Fig. 6** **a** Original audio signal length, **b** copy-move forgery of the audio signal, and **c** length of the audio signal after copy-move forgery



(a)



(b)



(c)

**Table 1** Data samples

Name	Original data	Forged data
Sample 1		
Sample 2		
Sample 3		
Sample 4		
Sample 5		

with those recordings. Thus, 4378 audio files were obtained: 2189 forged ones and 2189 original ones. During the forgery of the audio recordings, the third second of each recording was added after the first second. If the audio recording lasted 4 s, then an audio recording lasting 5 s was obtained by adding 1 s. Figure 6a shows the original audio signal length, Fig. 6b shows the situation of copy-move forgery to the audio signal, and Fig. 6c shows the length of the audio signal after copy-move forgery.

Table 1 provides examples of original and forged data in the data set used in our study.

### 4.2 Experimental setups

In our previous study [2], MFCCs and MFCCs were obtained from audio signals, and the correlations between those coefficients were examined. In another of our studies [54], LPC coefficients were obtained from audio signals. In both studies, by examining the correlations between the coefficients, we could detect copy-move forgery by comparing audio files according to a set threshold value.

In our other study [55], Mel frequency-based features were extracted and classified with ANN model.

In the study presented here, copy-move forgery detection in audio signals was based on RNN and LSTM deep learning models, which are called “sequential models.”

In the study, 2189 fake audio recordings were generated via copy-move forgery from 2189 original audio recordings. In the first stage of the study, the YAAPT method was used to separate audio signals into silent and unsilent segments. In the second stage, Mel-frequency-based features were extracted from the audio signals to represent the nonlinear relationships between the frequency of the human voice and hearing. In general, signals give stable, more reliable results because they are less affected by the recorded environment, the recording device, and features that vary from individual to individual depending on Mel features. To take temporal variation into account,  $\Delta$ MFCCs and  $\Delta\Delta$ MFCCs were also used. The pre-emphasis value to make the high-frequency components of the signal more dominant was determined to be 0.97, while to better detect the acoustic information and characteristics of the audio signals, the frame size was set at 25 ms and the overlap size to 10 ms, respectively. Hamming windowing was used to increase the continuity of the audio signals after framing and to reduce the distortions caused by FFT. DFT was applied to the signals to calculate their spectral values after each Hamming windowing process. The FFT size was determined to be 512, and 26 filter banks were used for each sliding window size. Ultimately, 13 Mel-frequency coefficients were produced for each segment. Along with the static information contained in the audio signals, a stronger feature set was obtained by using the dynamic properties of the signals. As a result, a fusion feature was settled that contained 13 MFCCs, 13  $\Delta$ MFCCs, and 13  $\Delta\Delta$ MFCC features from each segment. Last, the data were classified in RNN and LSTM networks. Altogether, a model for audio copy-move forgery detection was developed. In our study, four experimental setups—two with RNN and two with LSTM—were tested, and the results of the experiments were compared. Figure 7 illustrates the structure of the model.

In the first experimental setup, AdaGrad was determined by changing the optimization parameter in the RNN model, which was used in designing the audio copy-move forgery detection system. In the classification phase of the experiment, three models were designed. Model M11 had three layers, with 8, 4, and 1 neurons, respectively; model M12 also had three layers, with 4, 8, and 1 neurons, respectively; and model M13 had three layers as well, with 16, 8, and 1 neurons, respectively. The

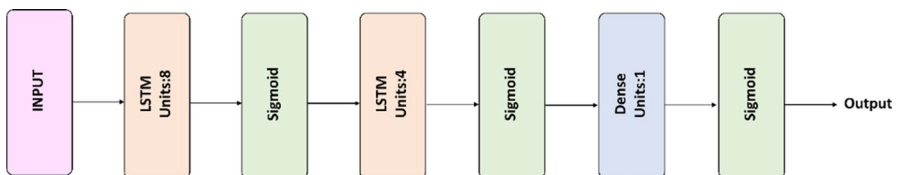


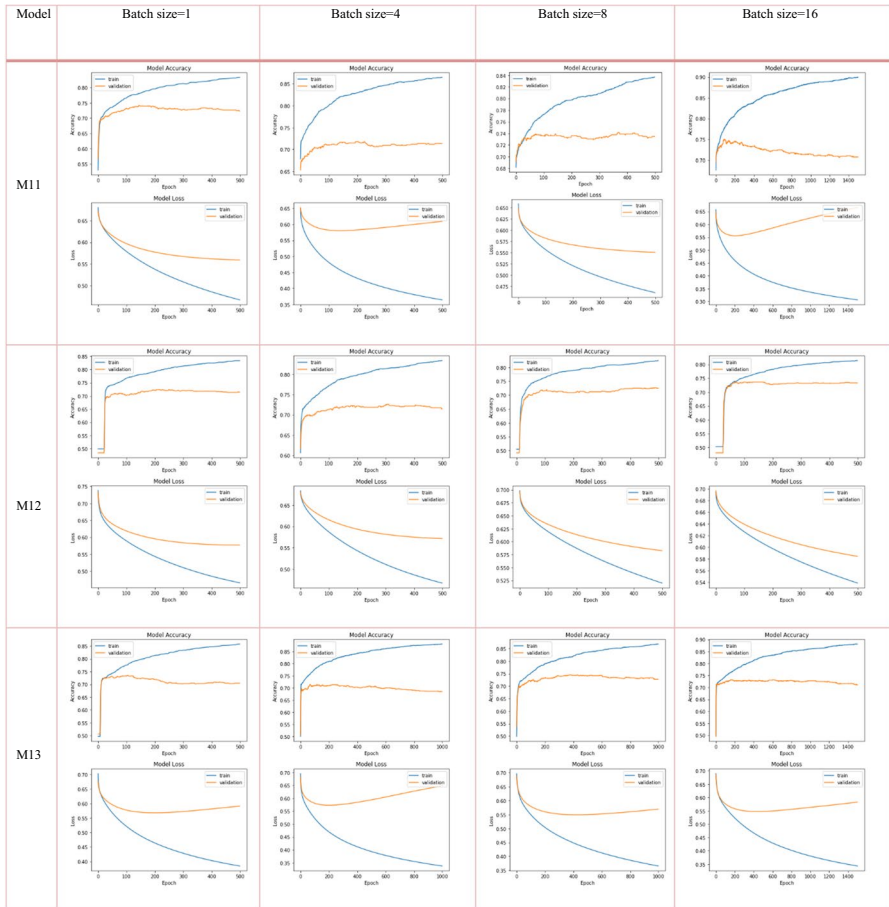
Fig. 7 Model overview



**Table 2** RNN models results with AdaGrad

Model	Epoch	Batch size = 1			Batch size = 4			Batch size = 8			Batch size = 16		
		Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)
M11	500	<b>83.40</b>	<b>72.33</b>	<b>75.23</b>	<b>86.50</b>	<b>71.33</b>	<b>72.83</b>	<b>83.79</b>	<b>73.47</b>	<b>75.91</b>	<b>84.43</b>	<b>77.18</b>	<b>72.37</b>
	1000	84.18	71.90	73.63	89.58	70.90	69.86	88.79	72.75	72.49	88.11	70.76	71.46
	1500	88.72	70.33	72.60	90.68	70.33	70.66	90.00	72.75	73.86	89.97	70.76	71.69
	2000	88.11	70.90	72.37	90.72	68.33	71.12	90.25	72.33	71.23	90.93	69.19	71.69
M12	500	<b>83.33</b>	<b>71.33</b>	<b>76.03</b>	<b>83.47</b>	<b>71.47</b>	<b>76.48</b>	<b>82.43</b>	<b>72.47</b>	<b>73.97</b>	<b>81.40</b>	<b>73.32</b>	<b>73.29</b>
	1000	86.22	72.90	70.32	86.43	72.90	71.23	86.11	70.47	72.95	85.54	71.18	71.92
	1500	87.25	72.33	74.89	87.00	72.18	71.92	86.83	72.90	71.58	86.36	73.18	72.03
	2000	87.61	71.33	73.06	87.08	72.90	75.11	88.22	72.18	72.26	88.83	69.33	71.23
M13	500	<b>85.79</b>	<b>70.47</b>	<b>73.17</b>	84.22	74.18	72.60	83.36	75.46	74.43	<b>81.65</b>	<b>72.47</b>	<b>73.74</b>
	1000	88.50	74.18	71.12	<b>88.00</b>	<b>68.47</b>	<b>73.29</b>	<b>86.83</b>	<b>72.75</b>	<b>74.54</b>	87.79	72.18	70.89
	1500	90.25	74.47	68.95	89.83	70.76	71.23	89.11	72.18	71.80	88.11	71.04	73.06
	2000	91.22	70.47	70.21	90.82	70.90	69.98	89.36	72.75	70.43	90.79	70.76	70.89

**Table 3** Accuracy and loss graphics for fusion MFCCs using RNN with AdaGrad



sigmoid activation function was determined for use in all three sets of layers, and AdaGrad and binary\_crossentropy were used in all three models.

The results for the M11, M12, and M13 RNN models results appear in Table 2. The hybrid features data were trained on different hyperparameters. For M11, the best accuracy rate was obtained with 500 epochs and a batch size of 1, for rates of 83.40, 72.33, and 75.23% for training, validation, and test accuracy, respectively. For M12, the best accuracy rate was obtained with 500 epochs and a batch size of 4, for rates of 83.47, 71.47, and 76.48% for training, validation, and test accuracy, also, respectively. Last, for M13, the best accuracy rate was obtained with between 500 and 1000 epochs and a batch size of 8, for rates of 86.83, 72.75, and 74.54% for training, validation, and test accuracy, respectively. The RNN model results with AdaGrad appear in Table 2, whereas the accuracy and loss graphics for MFCCs using RNN with AdaGrad appear in Table 3.

In the second experimental setup, an audio copy-move forgery detection system was designed by developing the RNN model with AdaDelta. In the classification phase of the experiment, three models were designed. Model M1 had three layers, with 8, 4, and 1 neurons, respectively; model M2 also had three layers, with 4, 8, and 1 neurons, respectively; and model M3 had three layers as well, with 16, 8, and 1 neurons, respectively. The sigmoid activation function was determined for use in all three sets of layers, and AdaDelta and binary\_crossentropy were used in all three models.

In the second experimental setup, the hybrid features data were trained on different hyperparameters. For M1, the best accuracy rate was obtained with at least 1500 epochs and a batch size of 4, for rates of 79.51, 74.04, and 74.54% for training, validation, and test accuracy, respectively. For M2, the best accuracy rate was obtained with 2000 epochs and a batch size of 4, for rates of 76.69, 72.04, and 75.51% for training, validation, and test accuracy, respectively. For M3, the best accuracy rate was obtained with between 1000 and 1500 epochs and a batch size of 16, for rates of 73.40, 73.61, and 76.94% for training, validation, and test accuracy, respectively. The RNN model results with AdaDelta appear in Table 4, whereas the accuracy and loss graphics for MFCCs using RNN with AdaDelta appear in Table 5.

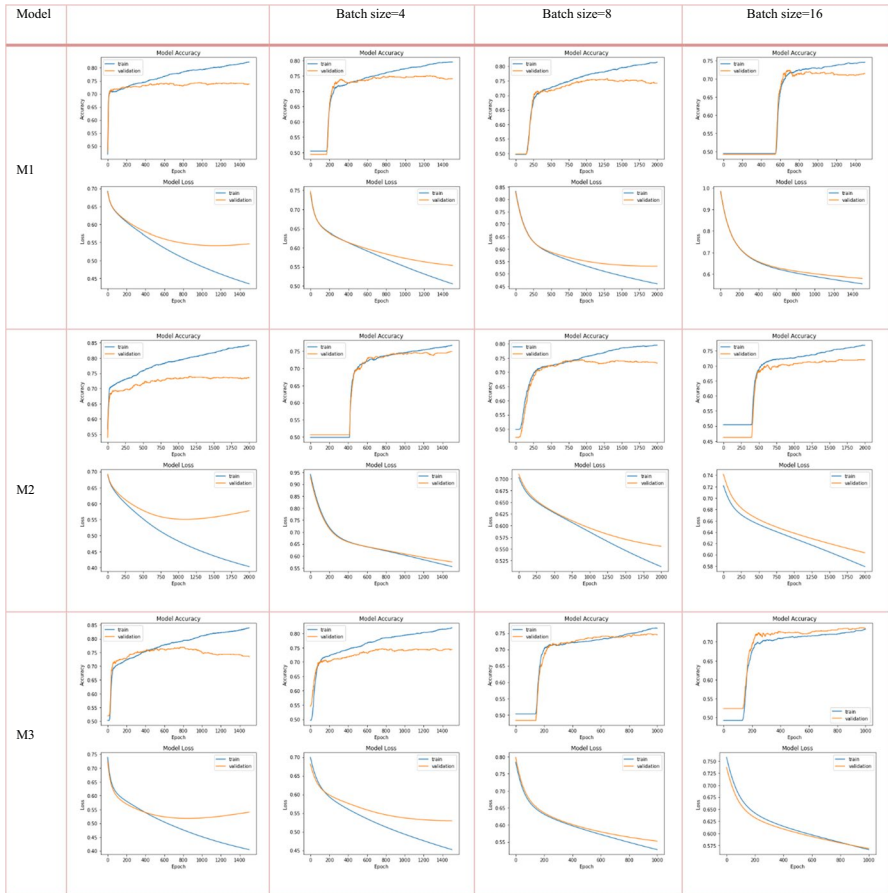
In the third experimental setup, AdaGrad was determined by changing the optimization parameters in the LSTM model, which was used to design the audio copy-move forgery system. Results for M11, M12, and M13 appear in Table 6. The hybrid features data were trained on different hyperparameters. For M11, the best accuracy rate was obtained with 500 epochs and a batch size of 8 as 81.76, 69.19, and 76.37% for training, validation, and test accuracy, respectively. For M12, the best accuracy rate was obtained with between 1000 and 1500 epochs and a batch size of 1, for rates of 87.04, 73.18, and 75.11% for training, validation, and test accuracy, respectively. For M13, the best accuracy rate was obtained with 500 epochs and a batch size of 4, for rates of 83.40, 72.75, and 76.14% for training, validation, and test accuracy, respectively. Accuracy and loss graphics for MFCCs using LSTM with AdaGrad appear in Table 7.

Last, in the fourth experimental setup, the audio copy-move forgery detection system was designed based on the LSTM model with AdaDelta. The results for the M1, M2, and M3 LSTM models appear in Table 8. The hybrid features data were trained on different hyperparameters. For M1, the best accuracy rate was obtained with 2000 epochs and a batch size of 8, for rates of 74.90, 73.32, and 75.80% for training, validation, and test accuracy, respectively. For M2, the best accuracy rate was obtained with 2000 epochs and a batch size of 1, for rates of 81.79, 72.33, and 74.66% for training, validation, and test accuracy, respectively. For M3, the best accuracy rate was obtained with between 1000 and 2000 epochs and a batch size of 1, for rates of 82.72, 75.75, and 76.03% for training, validation, and test accuracy, respectively. Accuracy and loss graphics for MFCCs using LSTM with AdaDelta appear in Table 9.

**Table 4** RNN model results with AdaDelta

Model	Epoch	Batch size = 1			Batch size = 4			Batch size = 8			Batch size = 16		
		Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)
M1	500	74.83	72.33	73.86	74.76	70.04	73.52	72.94	72.90	72.95	73.12	70.19	69.06
	1000	80.01	73.47	71.92	76.94	75.46	74.32	77.62	72.18	71.46	74.94	68.33	73.63
	1500	<b>82.22</b>	<b>73.75</b>	<b>73.52</b>	<b>79.51</b>	<b>74.04</b>	<b>74.54</b>	78.97	75.46	71.12	74.58	71.47	73.40
	2000	83.72	75.18	71.35	83.68	76.18	72.83	<b>81.44</b>	<b>74.18</b>	<b>73.06</b>	<b>80.01</b>	<b>71.61</b>	<b>73.29</b>
M2	500	75.44	77.46	72.95	73.51	71.18	72.26	72.05	70.61	69.98	49.09	51.50	51.71
	1000	79.54	72.33	73.17	77.12	71.90	75.23	74.26	72.75	70.32	73.22	73.18	70.89
	1500	81.90	70.61	75.80	76.76	74.89	75.57	77.94	72.90	74.66	75.44	71.90	71.00
	2000	<b>84.22</b>	<b>73.47</b>	<b>74.32</b>	<b>79.69</b>	<b>72.04</b>	<b>75.51</b>	<b>79.47</b>	<b>73.18</b>	<b>75.23</b>	<b>76.87</b>	<b>72.04</b>	<b>74.89</b>
M3	500	75.37	73.75	73.40	75.01	68.90	74.20	72.83	72.75	71.58	73.08	72.47	71.46
	1000	81.86	75.61	73.74	78.72	75.18	73.63	<b>76.51</b>	<b>74.47</b>	<b>74.20</b>	<b>73.40</b>	<b>73.61</b>	<b>76.94</b>
	1500	<b>84.01</b>	<b>73.32</b>	<b>75.23</b>	<b>81.94</b>	<b>74.32</b>	<b>74.77</b>	80.04	70.61	72.72	76.54	70.90	74.43
	2000	90.93	71.47	69.63	83.47	71.33	72.49	83.54	70.76	72.15	79.51	71.47	73.86

**Table 5** Accuracy and loss graphics for fusion MFCCs using RNN with AdaDelta



### 4.3 Results

In our study, 2801 training data, 701 validation data, and 876 test data were used, and accuracy was chosen as a metric to calculate performance. The total number of predictions made was taken into account when calculating accuracy, namely by dividing the number of correct predictions made as a result of training by the total number of predictions. The formula for calculating performance is given in Eq. (14).

$$\text{Accuracy(Acc)} = \frac{1}{N} \sum_i^N y_i == y_i^{\text{prediction}} \tag{14}$$

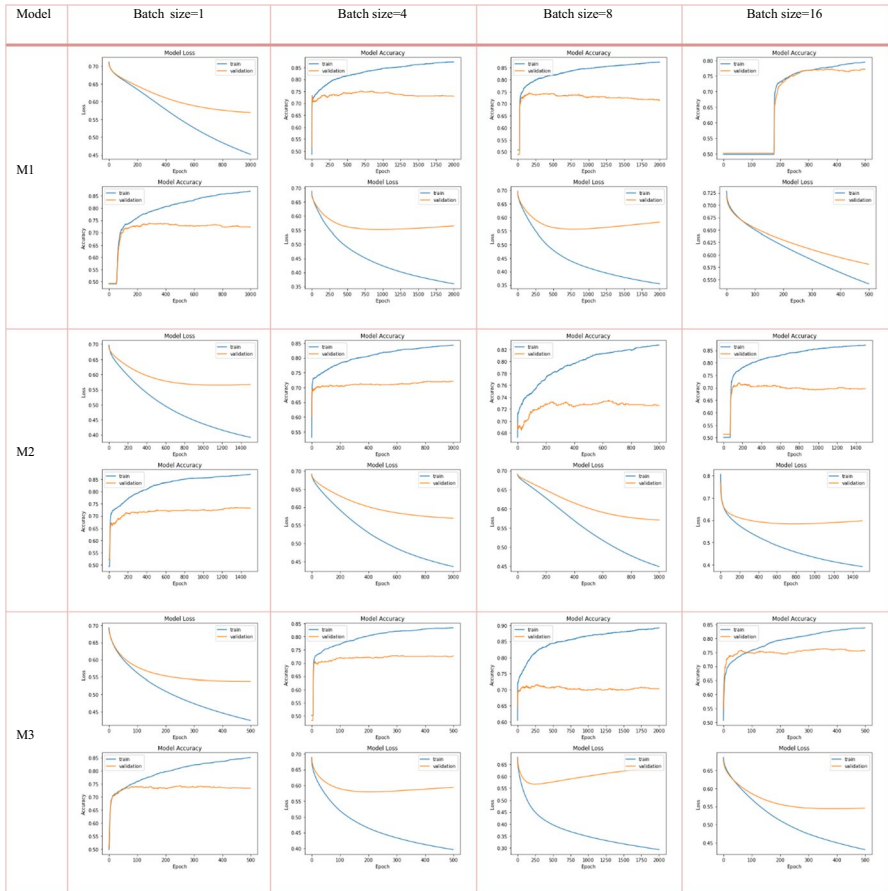
in which  $y_i$  represents actual values,  $y_i^{\text{prediction}}$  represents predicted values, and  $N$  represents the number of items.

Table 10 compares our method with alternative methods in the literature.

**Table 6** LSTM models results with AdaGrad

Model	Epoch	Batch size = 1			Batch size = 4			Batch size = 8			Batch size = 16		
		Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)
M11	500	<b>82.58</b>	<b>72.47</b>	<b>74.32</b>	<b>80.76</b>	<b>70.61</b>	<b>74.77</b>	<b>81.76</b>	<b>69.19</b>	<b>76.37</b>	<b>79.47</b>	<b>77.03</b>	<b>72.60</b>
	1000	86.97	72.33	74.77	84.33	72.18	72.95	85.01	75.32	73.06	84.54	73.32	70.89
	1500	88.58	73.47	71.35	86.79	71.47	71.23	85.58	73.04	72.60	85.29	74.04	70.09
	2000	89.04	73.47	70.32	87.29	72.90	73.06	87.36	71.47	74.32	86.22	70.76	72.37
M12	500	81.61	73.32	74.43	80.86	74.61	72.72	81.29	71.04	71.69	80.76	73.04	73.97
	1000	85.04	70.47	74.54	<b>84.33</b>	<b>72.04</b>	<b>73.74</b>	<b>82.79</b>	<b>72.61</b>	<b>73.86</b>	82.68	75.04	74.09
	1500	<b>87.04</b>	<b>73.18</b>	<b>75.11</b>	86.36	72.18	70.09	86.04	71.18	72.72	<b>87.11</b>	<b>69.61</b>	<b>74.77</b>
	2000	87.65	75.18	69.41	86.00	70.19	73.17	85.79	73.18	72.60	85.83	72.47	74.54
M13	500	<b>85.18</b>	<b>73.32</b>	<b>74.43</b>	<b>83.40</b>	<b>72.75</b>	<b>76.14</b>	<b>83.83</b>	<b>75.18</b>	<b>72.37</b>	<b>83.79</b>	<b>75.46</b>	<b>73.86</b>
	1000	89.29	72.04	70.32	87.58	73.47	72.26	86.93	74.32	71.12	85.04	74.04	72.26
	1500	91.40	69.90	70.78	87.97	71.61	72.49	88.00	73.32	70.21	88.68	72.33	72.03
	2000	91.82	75.04	67.69	89.47	70.90	72.95	89.29	70.33	72.72	89.40	69.47	71.23

**Table 7** Accuracy and loss graphics for fusion MFCCs using LSTM with AdaGrad



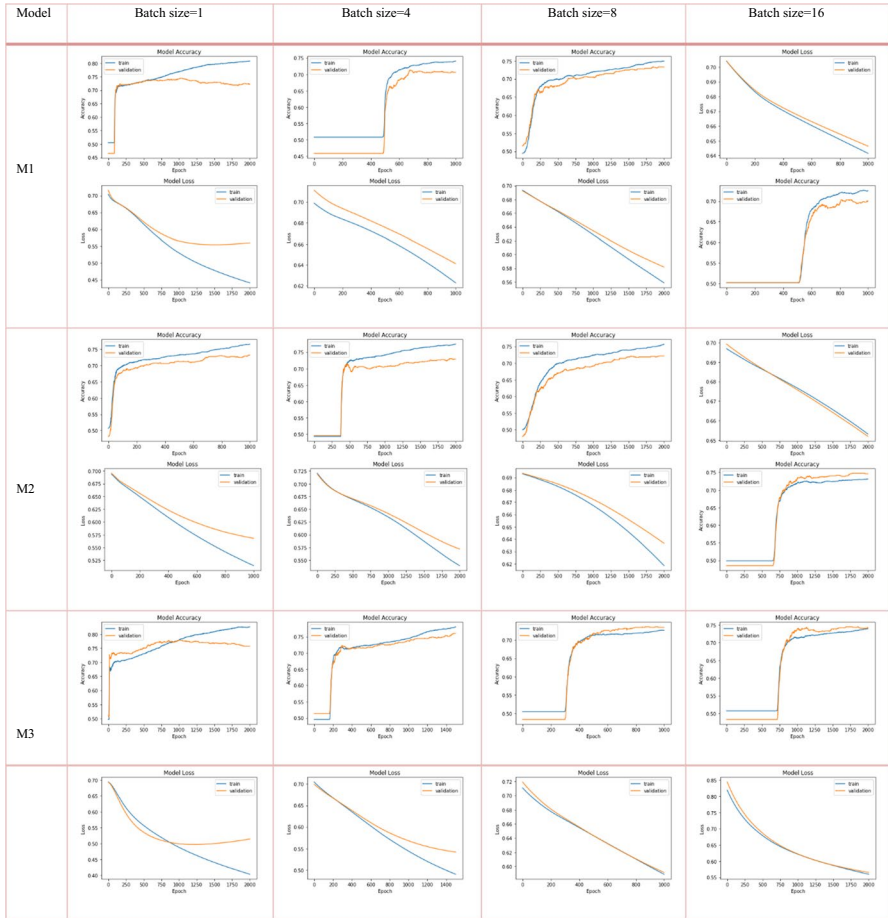
As many researchers have emphasized, studies on audio copy-move forgery detection are few, and no system exists for detecting such forgery despite the urgent need for one. As shown in Table 10, most such studies have involved extracting features from audio signals and detecting forgery by examining the degree of similarity and the relationships between those features. In those studies, a fixed decision criterion, or threshold value, has been used to decide how similar the data are. Because that value is fixed, it depends on the data used and their properties. As a consequence, the developed systems cannot be generally used for all data, because the threshold value needs to be constantly updated depending on changes to the device used to record the data, the file format, the environment, and the speaker. In our study, we developed a system that automatically detects audio copy-move forgery. To that purpose, the system extracts effective features from the signal and trains them using learning-based algorithms, after which decision-making is performed. Thus, the system allows detecting audio

**Table 8** Results of the LSTM models using AdaDelta

Model	Epoch	Batch size = 1			Batch size = 4			Batch size = 8			Batch size = 16		
		Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)
M1	500	73.72	72.47	71.58	71.55	70.04	68.72	49.98	48.50	51.26	49.30	50.21	52.05
	1000	77.40	72.18	73.40	74.04	70.61	73.40	72.37	71.75	72.49	72.55	70.04	73.74
	1500	79.65	75.18	72.95	75.12	74.61	69.52	74.65	74.47	72.03	67.73	63.05	65.98
	2000	<b>80.83</b>	<b>72.18</b>	<b>73.97</b>	<b>78.08</b>	<b>73.89</b>	<b>72.49</b>	<b>74.90</b>	<b>73.32</b>	<b>75.80</b>	<b>72.37</b>	<b>70.90</b>	<b>72.37</b>
M2	500	73.69	73.47	71.46	72.26	72.33	68.84	50.12	46.65	52.28	50.87	48.36	48.52
	1000	76.58	73.18	75.91	74.04	69.61	72.95	72.37	74.47	69.52	61.37	63.20	56.51
	1500	78.86	73.75	73.74	75.04	72.47	73.97	73.69	74.18	68.84	72.08	70.90	69.29
	2000	<b>81.79</b>	<b>72.33</b>	<b>74.66</b>	<b>77.51</b>	<b>72.90</b>	<b>73.86</b>	<b>75.76</b>	<b>72.18</b>	<b>72.60</b>	<b>73.12</b>	<b>74.61</b>	<b>69.86</b>
M3	500	74.01	73.04	74.09	71.80	70.61	72.83	67.76	66.33	68.72	63.44	67.62	64.61
	1000	78.65	74.18	75.57	<b>74.94</b>	<b>73.75</b>	<b>73.52</b>	<b>72.65</b>	<b>73.47</b>	<b>73.74</b>	72.37	71.33	70.66
	1500	81.83	72.33	74.32	78.08	76.03	73.29	75.22	70.76	72.60	72.22	73.04	72.03
	2000	<b>82.72</b>	<b>75.75</b>	<b>76.03</b>	80.86	74.61	72.49	77.29	74.89	72.15	<b>73.94</b>	<b>74.32</b>	<b>73.06</b>



**Table 9** Accuracy and loss graphics for fusion MFCCs using LSTM with AdaDelta



copy-move forgery using any threshold value. In our study, a larger data set was compared with data sets in the literature as well.

## 5 Discussion

As demonstrated in the literature, researchers have mostly extracted features from audio signals for audio forgery detection and used statistical methods in the decision-making process. Because such decision-making in those methods depends on a threshold value that is not dynamic, decisions may be affected by environmental conditions, speaker’s characteristics (e.g., gender), recording devices, and other factors. Such factors negatively affect copy-move forgery detection systems using audio signals. In our study, we proposed a system for detecting audio copy-move forgery according to all kinds of variable conditions without depending on any threshold

**Table 10** Comparison of methods

Literature	Features	Decision system	Threshold	Data name, Data count
Li et al.	Pitch frequency	PCC	Yes	– 500 audio records
Yan et al.	Pitch frequency and formant feature	Dynamic time warping	Yes	TIMIT and WSJ database 20,000 duplicated segments, 20,000 non-forged segment
Huang et al.	DFT	Compared each segment	Yes	–
Xie et al.	Gammatone feature, MFCCs, pitch feature and DFT	C4.5, Decision Tree, PCCs and average difference	Yes	Self-generated dataset 1000 forgery, 1000 original records
Imran et al.	ID LBP	Compared between the histograms	Yes	King Saud University Arabic Speech Data
Wang et al.	DCT, SVD	Distance between singular vectors	Yes	– Self-generated dataset 100 forgery, 100 original audio records
Liu et al.	DFT, MFCC	PCC	Yes	Self-generated dataset 1000 audio records
Yan et al.	Pitch feature	PCC, average difference	Yes	Self-generated dataset 1000 forgery records
Xiao et al.	–	Fast convolution algorithm	Yes	–
Proposed methods	Fusion features	RNN LSTM	No	TIMIT 2189 forgery, 2189 original audio records

value. In short, audio forgery can be detected with high accuracy, independent of all factors, with our system. The novelty of our study is that it has generated the first audio copy-move forgery detection system using RNN and LSTM models. Different optimization parameters, including batch numbers and different epoch numbers for RNN and LSTM models, were tested in the study, and the results, detailed in Sect. 4, are supported with tables and graphs. Our study was also the first to involve a detailed analysis and generate results during audio copy-move forgery detection using sequence models.

The deep learning methods proposed in our study were compared with other methods in the literature. The main differences are summarized as follows:

- In the literature, most methods are traditional methods. However, the impact of the latest deep learning methods for big data in studies in every field is apparent. Methods in the literature differ considerably from deep learning methods that are trending today. Sequence deep learning models, frequently used in time-series problems but not previously tested in the field, were used to overcome that deficiency.
- Our proposed method's results were compared with the one-dimensional LBP method [26], the pitch feature method [20], the pitch feature and formant feature methods [7], the DFT method [9], the gammatone feature method, the MFCCs method, the pitch feature and DFT method [8], the DCT method, the SVD method [14], DFT method, and the results of using MFCCs [19]. Results appear in Table 10.
- When the results are examined, our proposed algorithm offers highly accurate results without a threshold value due to training on hybrid features data.
- AdaGrad is an adaptive gradient learning algorithm that can adapt the learning rate of the parameters. The algorithm thus learns the learning rate itself. Although AdaGrad has a dynamic structure, it operates using a different learning coefficient at each step. The most important advantage of the algorithm is that it does not require manually adjusting the learning rate. The optimization algorithm has been proposed as a solution to the fixed learning step problem in the SGD and momentum learning algorithms. Because the learning rate is gradually decreased in the algorithm, the developed model terminates learning at any time  $t$ . That benefit is the greatest disadvantage of AdaGrad [56–59].
- AdaDelta is a variation of AdaGrad produced as a solution to the fixed learning rate problem. The optimization method does not require choosing a learning coefficient different from AdaGrad [56]. In addition, AdaDelta is more robust than algorithms using noisy gradient information, different model architectures, and different types of data [59, 60].
- In the training phase using AdaGrad, the manual selection of the learning rate and the continuous decrease in the rate reveal the inadequacy of AdaGrad compared with AdaDelta. The results show that AdaDelta gives more successful rates than AdaGrad.

One of the most challenging problems in forensics is audio copy-move forgery detection. Researchers have consistently emphasized the lack of studies in audio

forensics detection. In addition, a system for detecting copy-move forgery has not been developed. In our study, a solution was therefore developed for one of the most important problems in forensic informatics.

## 6 Conclusions

The most-needed method of detecting audio forgery in audio forensics is the audio copy-move forgery detection method. Although a few studies have been conducted on that topic, there is no integrated system for forgery detection, despite its critical importance to the field, for both researchers and practitioners. In our study, sequence RNN and LSTM deep learning models were used for the first time, in contrast to traditional audio copy-move forgery detection studies in the literature. In our study, RNN and LSTM models were tested, and different hyperparameters were adjusted in those models. The best results were obtained with the LSTM method, which minimizes the vanishing (i.e., exploding) gradient problem.

In our study, the results obtained using AdaDelta were more successful than using AdaGrad. In addition, with low batch size parameters, more stable results were obtained. The best performance was obtained with a batch size of 1 in M3 based on the LSTM algorithm. The best performance results were calculated as 82.72, 75.75, and 76.03% for training accuracy, validation accuracy, and test accuracy, respectively.

**Acknowledgements** Our study was supported by the Scientific and Technological Research Council of Turkey (Grant No. 121E725).

**Author contributions** F.A: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Software, Resources, Data curation. Y.B: Supervision, Conceptualization, Review & Editing.

**Funding** Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). We received research funding from the Scientific and Technological Research Council of Turkey (Grant No. 121E725).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Haniłçi C, Kinnunen T, Sahidullah M, Sizov A (2015) Classifiers for synthetic speech detection: a comparison
2. Akdeniz F, Becerikli Y (2021) Detection of copy-move forgery in audio signal with Mel frequency and delta-Mel frequency Cepstrum coefficients. In: 2021 Innovations in Intelligent Systems and Applications Conference (ASYU), IEEE, pp 1–6
3. Kasapođlu B, Turgay KOÇ (2020) Sentetik ve Dönüştürülmüş Konuşmaların Tespitinde Genlik ve Faz Tabanlı Spektral Özniteliklerin Kullanılması. *Avrupa Bilim ve Teknoloji Dergisi*, pp 398–406
4. Aziz S, Shahnawazuddin S (2023) Effective preservation of higher-frequency contents in the context of short utterance based children’s speaker verification system. *Appl Acoust* 209:109420
5. Shi C, Li X, Wang H (2020) A novel integrity authentication algorithm based on perceptual speech hash and learned dictionaries. *IEEE Access* 8:22249–22265
6. Chamot F, Geradts Z, Haasdijk E (2022) Deepfake forensics: cross-manipulation robustness of feedforward-and recurrent convolutional forgery detection methods. *Forensic Sci Int Digital Invest* 40:301374
7. Yan Q, Yang R, Huang J (2019) Robust copy–move detection of speech recording using similarities of pitch and formant. *IEEE Trans Inf Forensics Secur* 14(9):2331–2341
8. Xie Z, Lu W, Liu X, Xue Y, Yeung Y (2018) Copy-move detection of digital audio based on multi-feature decision. *J Inf Secur Appl* 43:37–46
9. Javed A, Malik KM, Irtaza A, Malik H (2021) Towards protecting cyber-physical and IoT systems from single-and multi-order voice spoofing attacks. *Appl Acoust* 183:108283
10. Huang Y, Hou H, Wang Y, Zhang Y, Fan M (2020) A long sequence speech perceptual hashing authentication algorithm based on constant q transform and tensor decomposition. *IEEE Access* 8:34140–34152
11. Maher RC (2009) Audio forensic examination. *IEEE Signal Process Mag* 26(2):84–94
12. Xiao JN et al (2014) Audio authenticity: duplicated audio segment detection in waveform audio file. *J Shanghai Jiaotong Univ (Sci)* 19(4):392–397
13. Goyal A, Shukla SK, Sarin RK (2021) A comparative study of audio latency feature of Motorola and Samsung mobile phones in forensic identification. *Indian J Sci Technol* 14(4):319–324
14. Wang F, Li C, Tian L (2017) An algorithm of detecting audio copy-move forgery based on DCT and SVD. In: 2017 IEEE 17th International Conference on Communication Technology (ICCT), IEEE, pp 1652–1657
15. Jadhav S, Patole R, Rege P (2019) Audio splicing detection using convolutional neural network. In: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, pp 1–5
16. Chen J, Xiang S, Huang H, Liu W (2016) Detecting and locating digital audio forgeries based on singularity analysis with wavelet packet. *Multimed Tools Appl* 75(4):2303–2325. <https://doi.org/10.1007/s11042-014-2406-3>
17. Yang R, Qu Z, Huang J (2008) Detecting digital audio forgeries by checking frame offsets. In: Proceedings of the 10th ACM Workshop on Multimedia and Security, pp 21–26
18. Gupta S, Cho S, Kuo CCJ (2011) Current developments and future trends in audio authentication. *IEEE Multimed* 19(1):50–59. <https://doi.org/10.1109/MMUL.2011.74>
19. Liu Z, Lu W (2017) Fast copy-move detection of digital audio. In: 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), IEEE, pp 625–629
20. Li C, Sun Y, Meng X, Tian L (2019) Homologous audio copy-move tampering detection method based on pitch. In: 2019 IEEE 19th International Conference on Communication Technology (ICCT), IEEE, pp 530–534
21. Imran M, Ali Z, Bakhsh ST, Akram S (2017) Blind detection of copy-move forgery in digital audio forensics. *IEEE Access* 5:12843–12855. <https://doi.org/10.1109/ACCESS.2017.2717842>
22. Yan Q, Yang R, Huang J (2015) Copy-move detection of audio recording with pitch similarity. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 1782–1786
23. Kroon A (2022) Comparing conventional pitch detection algorithms with a neural network approach. *arXiv preprint arXiv:2206.14357*
24. Zahorian SA, Hu H (2008) A spectral/temporal method for robust fundamental frequency tracking. *J Acous Soc Am* 123(6):4559–4571

25. Sukhostat L, Imamverdiyev Y (2015) A comparative analysis of pitch detection methods under the influence of different noise conditions. *J Voice* 29(4):410–417
26. Kadiri SR, Yegnanarayana B (2018) Estimation of fundamental frequency from singing voice using harmonics of impulse-like excitation source. In *INTERSPEECH*, pp 2319–2323
27. Ferro M, Tamburini F (2019) Using deep neural networks for smoothing pitch profiles in connected speech. *IJCoL Italian J Comput Linguist* 5(5–2):33–48
28. Abbiyansyah MZ, Utaminigrum F (2022) Voice recognition on humanoid robot darwin OP using Mel frequency cepstrum coefficients (MFCC) feature and artificial neural networks (ANN) method. In: 2022 2nd International Conference on Information Technology and Education (ICIT&E), IEEE, pp 251–256
29. Shao H, Yuan J, Huang H (xxxx) Recognition recognition types of cracked material under uniaxial tension based on improved Mel frequency cepstral coefficients (Mfcc)
30. Jayalakshmi SL, Chandrakala S, Nedunchelian R (2018) Global statistical features-based approach for acoustic event detection. *Appl Acoust* 139:113–118
31. Liu JC, Leu FY, Lin GL, Susanto H (2018) An MFCC-based text-independent speaker identification system for access control. *Concurr Comput Practice Exp* 30(2):e4255
32. Das PP, Allayear SM, Amin R, Rahman Z (2016) Bangladeshi dialect recognition using Mel frequency cepstral coefficient, delta, delta-delta and Gaussian mixture model. In: 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI), IEEE, pp 359–364
33. Hossan MA, Memon S, Gregory MA (2010) A novel approach for MFCC feature extraction. In: 2010 4th International Conference on Signal Processing and Communication Systems, IEEE, pp 1–5
34. Eskidere Ö, Ertuş F (2009) Mel frekans kepstrem katsayılarındaki değışimlerin konuşmacı tanımaya etkisi. *Uludağ Üniversitesi Mühendislik Fakültesi Dergisi*, 14(2)
35. Vimal W (2022) Study on the behaviour of Mel frequency cepstral coefficient algorithm for different windows. In: 2022 International Conference on Innovative Trends in Information Technology (ICITIT), IEEE, pp 1–6
36. Boualoulou N, Nsiri B, Belhoussine Drissi T, Zayrit S (2022) Speech analysis for the detection of Parkinson's disease by combined use of empirical mode decomposition, Mel frequency cepstral coefficients, and the K-nearest neighbor classifier. *ITM Web Conf* 43:01019. <https://doi.org/10.1051/itmconf/20224301019>
37. Naing HMS, Miyanaga Y, Hidayat R, Winduratna B (2019) Filterbank analysis of MFCC feature extraction in robust children speech recognition. In: 2019 International Symposium on Multimedia and Communication Technology (ISMAT), IEEE, pp 1–6
38. Yücesoy E (2021) MFKK Özniteliklerine Eklenen Logaritmik Enerji ve Delta Parametrelerinin Yaş ve Cinsiyet Sınıflandırma Üzerindeki Etkileri. *J Inst Sci Technol* 11(1):32–43
39. Boussaa M, Atouf I, Atibi M, Bennis A (2016) ECG signals classification using MFCC coefficients and ANN classifier. In: 2016 International Conference on Electrical and Information Technologies (ICEIT), IEEE, pp 480–484
40. Ittichaichareon C, Suksri S, Yingthawornsuk T (2012) Speech recognition using MFCC. In: International Conference on Computer Graphics, Simulation and Modeling, Vol. 9
41. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323(6088):533–536
42. El-Moneim SA, Nassar MA, Dessouky MI, Ismail NA, El-Fishawy AS, Abd El-Samie FE (2020) Text-independent speaker recognition using LSTM-RNN and speech enhancement. *Multimed Tools Appl* 79:24013–24028
43. Takeuchi D, Yatabe K, Koizumi Y, Oikawa Y, Harada N (2020). Real-time speech enhancement using equilibrated RNN. In: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 851–855
44. İlyas ÖZER (2020) Uzun Kısa Dönem Bellek Ağlarını Kullanarak Erken Aşama Diyabet Tahmini. *Mühendislik Bilimleri ve Araştırmaları Dergisi* 2(2):50–57
45. Çakır B, Angin P (2021) Zamansal Evrişimli Ağlarla Saldırı Tespiti: Karşılaştırmalı Bir Analiz. *Avrupa Bilim ve Teknoloji Dergisi* 22(204):211
46. Borges D, Nascimento MC (2022) COVID-19 ICU demand forecasting: a two-stage Prophet-LSTM approach. *Appl Soft Comput* 125:109181
47. Syed SA, Rashid M, Hussain S, Zahid H (2021) Comparative analysis of CNN and RNN for voice pathology detection. *Biomed Res Int* 2021:1–8
48. Hassan E, Shams MY, Hikal NA, Elmougy S (2023) The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study. *Multimed Tools Appl* 82(11):16591–16633

49. Yazan E, Talu MF (2017) Comparison of the stochastic gradient descent based optimization techniques. In: 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), IEEE, pp 1–5
50. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv preprint [arXiv:1609.04747](https://arxiv.org/abs/1609.04747)
51. Brownlee J (2018) What is the difference between a batch and an epoch in a neural network. Machine Learning Mastery, 20
52. Kandel I, Castelli M (2020) The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express* 6(4):312–315
53. Garofolo JS, Lamel LF, Fisher WM, Fiscus JG, Pallett DS (1993) DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1–1.1. NASA STI/Recon technical report n, 93, 27403
54. Akdeniz F, Becerikli Y (2022) Linear prediction coefficients based copy-move forgery detection in audio signal. In: 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), IEEE, pp 770–773
55. Akdeniz F, Becerikli Y (2024) Detecting audio copy-move forgery with an artificial neural network. *Signal Image Video Proc*, pp 1–17
56. Lu J (2023) AdaSmooth: an adaptive learning rate method based on effective ratio. In: *Sentiment analysis and deep learning: proceedings of ICSADL 2022*, Singapore: Springer Nature Singapore, pp 273–293
57. Gaddam DK, Ansari MD, Vuppala S, Gunjan VK, Sati MM (2022) A performance comparison of optimization algorithms on a generated dataset. In: *ICDSMLA 2020 Proceedings of the 2nd International Conference on Data Science, Machine Learning and Applications*, Springer Singapore, pp 1407–1415
58. Seyyarer E, Karci A, Abdullah ATEŞ (2021) Stokastik ve deterministik hareketlerin optimizasyon süreçlerindeki etkileri. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi* 37(2):949–966
59. Seyyarer E, Uckan T, Hark C, Ayata F, İnan M, Karci A (2019) Applications and comparisons of optimization algorithms used in convolutional neural networks. In: 2019 International artificial Intelligence and Data Processing Symposium (IDAP), IEEE, pp 1–6
60. Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv preprint [arXiv:1212.5701](https://arxiv.org/abs/1212.5701).

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.