# Blind Source Separation and Audio Classification

Mariachiara Acconcia
Drexel University
Philadelphia, PA, USA
ma3768@drexel.edu

Natasha Lalwani
Drexel University
Philadelphia, PA, USA
nl498@drexel.edu

Edward Kim
Drexel University
Philadelphia, PA, USA
ek826@drexel.edu

## ABSTRACT

Blind Source Separation by Independent Component Analysis (ICA) has been gaining a lot of attention in the past few years. The main idea is to recover independent sources of signal from a mixed signal, without any prior knowledge about the source or mixing process. Blind Source Separation methods allow to recover all individual sources from the mixed signal but, in most situations, it would be useful to not only separate the individual sources but also to recognize and classify them. In this paper, apart from separating an audio consisting of mixed signals we aim at classifying the separated audio signals. Separation and classification are two different approaches which include both Machine Learning (FastICA) and Deep Learning (Artificial Neural Network and Convolutional Neural Network) models, respectively. We worked on "Speaker Recognition Audio Dataset" by extracting the $mfcc$ features and $spectrograms$. Good results were obtained after training these models using the extracted features.

## Keywords

Blind Source Separation, FastICA, Artificial Neural Network, Convolutional Neural Network, MFCC, Spectrogram

## 1. INTRODUCTION

Blind Source Separation (BSS) is the process of recovering unseen signals or sources from a number of observed mixes. The word *blind* emphasizes that 1) the source signals are not noticed, and 2) no information about the mixture is provided. It separates the signals from a set of mixed signals with very little information or none at all, regarding the source signals or mixing process. This approach is also known as Independent Component Analysis (ICA) when multiple components of the source signal have separate characteristics. ICA tries to maximize independence by computing the linear transformation of the feature space into a new feature space, such that each of the individual new features are mutually independent. The blind signal separation approach based on ICA is now widely utilized in wireless communications, radar, sonar, image, speech, and medical applications, among others [4, 13].

Along with blind source separation, our focus is also to classify the signals. Speaker recognition is a fundamental task in speech processing that has numerous applications in real-world scenarios. It is used, for example, for voice-based authentication of personal smart devices such as cellular phones, vehicles, and laptop computers. It has been widely used in forensics, as well as surveillance and automatic identity tagging. It is useful in audio-based information retrieval for broadcast news, meeting recordings, and phone calls. It can also be used as a front end for automatic speech recognition (ASR) to improve the transcription of multi-speaker conversations [3].

A lot of work has been done the field of speaker recognition incorporating the use of various deep learning models. [9] implemented three kinds of deep learning models; Multi-Layer Perceptron(MLP), Convolutional Neural Network(CNN) and Long Short-Term Memory(LSTM; a kind of RNN). A maximum accuracy of 92% was obtained using MLP. [2] implemented a Convolutional Neural Network in order to classify speech signals. They obtained an accuracy of 96%. [5] introduced a novel method for modelling speakers using restricted Boltzmann machine (RBM) to extract weight matrices which were then fed as input to CNN. Convolutional deep belief nerwork was applied to audio data in [7] and evaluated them on different audio classification task. For speech data, the learned features, which corresponded to phonemes, performed well on the task. Application of neural networks has also been seen for topics like Speech Emotion Recognition. [8] investigated the problem statement of Speech Emotion Recognition by training CNN and RNN, using an emotional speech database. They also proposed a time distributed CNN structure which gave better results than the original CNN and RNN.

Our paper consists of mainly two parts:
(1) the Blind Source Separation part, which was approached with the implementation of a FastICA model, to which we fed mixed audio signals obtained by combining individual speakers' voices
(2) the classification part, for which we explored neural networks like Artificial Neural Network and Convolutional Neural Network, into which we fed the results from the FastICA as inputs.

## 2. BACKGROUND

## 2.1 Artificial Neural Network

The structure of an Artificial Neural Network (ANN), simply known as Neural Network, is inspired by the human brain as it resembles the way biological neurons communicate with one another. An ANN comprises of an input layer (also known as input node to which input/ information is provided to in order to learn and derive conclusions from out model. It passes the data to the hidden layer), one or more hidden layers (consists of a set of neurons where all the computation is performed on the input data) and an output layer (contains the model's output/conclusions produced from all calculations/computation. It is binary classification problem when there is 1 output node, while in a multi-class classification problem, the output nodes might be more than 1). Each node, or artificial neuron, is connected to the next and has a weight and threshold associated with it. If a node's output exceeds a certain threshold, the node is activated, and data is sent to the next layer of the network. If this is not the case, no data is sent on to the network's next layer. Among various variations in ANN, Multi-Layer Perceptron is a common one. It basically consists of multiple hidden layers. Each node, with the exception of the input nodes, is a neuron with a nonlinear activation function. MLP is distinguished from a linear perceptron by its multiple layers and non-linear activation. It can tell the difference between data that isn't linearly separable. More specifically, we have implemented a Multi-Layer Perceptron.

## 2.2 CNN

Convolutional Neural Networks (ConvNet/CNN) are a type of Deep Learning algorithm that are distinguished from other neural networks by their superior performance with detection and classification of images. There are three main layers: Convolutional Layer, Pooling Layer and Fully Connected (FC) Layer. This is the initial stage in obtaining useful information from an image. The convolution action is performed by many filters in a convolution layer. Every image is seen as a pixel value matrix. Say we have a $5*5$ image and a filter matrix with dimension $3*3$, we slide the filter matrix over the image in order to compute the dot product so as to obtain the convolved feature matrix. This is then passed on to the ReLU layer which performs element wise operation and sets all the negative pixels to 0. We obtained a rectified feature map which is next fed to the pooling layer in order to obtain a pooled feature map. Various filters are used by this layer to identify different parts of the image. The resultant 2-dimensional array is flattened into a single long continuous linear vector. This is then fed to the FC layer which leverages the softmax function in order to perform the task of classification. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object. Our model is also based on the same architecture.

## 2.3 Fast ICA

The most popular problem, $CocktailPartyProblem$, is tackled using Independent Component Analysis (ICA). But what is ICA? ICA, which is a class of Blind Source Separation [11], maximizes independence by computing the linear transformation of the feature space into a new feature space, such that each of the individual new features are mutually independent. It basically converts a mix of audio signals into unmixed signals of each independent source. It can be noted that the number of inputs and outputs are the same and the outputs are mutually independent. ICA makes two key assumptions; first being that the source signals are statistically independent of each other and second, the value in each source signal has a non-Gaussian distribution. This fast-expanding approach is now finding applications in biomedical signal analysis (e.g., ERP, EEG, fMRI, optical imaging), as well as visual receptive field models and speech signal separation [10]. Our model more specifically makes use of an algorithm called $FastICA$, which is a computationally efficient approach for computing ICA. It employs a fixed-point iteration strategy that has been shown to be 10-100 times quicker than traditional gradient descent algorithms for ICA. FastICA has the majority of the benefits of neural algorithms: It's distributed, parallel, and computationally easy, and it just takes up a little amount of memory [6].

## 3. METHODOLOGY

Throughout this section, we will explain the methods used for every part of our problem, going from the extraction and pre-processing of features (MFCC and Spectrograms) to the implementation of classification models (ANN and CNN), and Blind Source Separation techniques (FastICA). We will go through every model in details, including a discussion on the choice of parameters and methods.

## 3.1 Data

For this project we used the audio files from eight different speakers from the "Speaker Recognition Audio Dataset" on Kaggle. The data acquisition was done through the use of the Kaggle and Kaggle Api modules, which simply downloaded all the audio samples in a .zip folder, and then unzipped them automatically. For each speaker, we collected 30 to 50 .wav audio files with length of 60 seconds each.

## 3.2 Feature Extraction

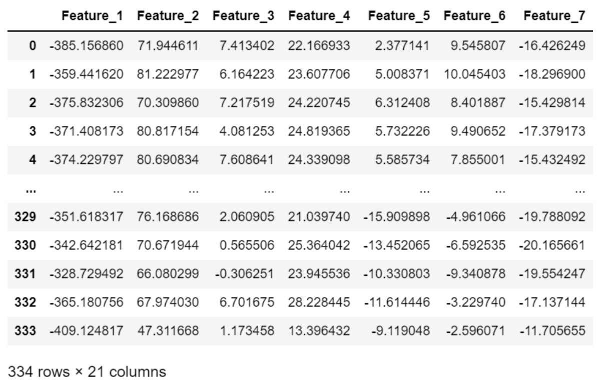
### *3.2.1 Mel Frequency Cepstral Coefficients (MFCC)*

When working with audio files, especially those including human recordings and speeches, Mel Frequency Cepstral Coefficients (MFCC) is a critical component to be considered. The shape of the vocal tract determines any sound produced by humans (including tongue, teeth, etc.). If this shape is correctly determined, any sound produced can be accurately represented. The envelope of the time power spectrum of a speech signal represents the vocal tract, and MFCC accurately represents this envelope. The Mel scale relates a pure tone's perceived frequency, or pitch, to its actual measured frequency. At low frequencies, humans are much better at detecting small changes in pitch than at high frequencies. By incorporating this scale, we can better match our features to what humans hear. In order to obtain MFCCs from the typical frequency values, the signals are converted from Hertz Scale to Mel Scale. A frequency measured in Hertz

(f) can be converted to the Mel scale using the following formula:

$$Mel(f) = 2595 \log(1 + \frac{f}{700}) \quad (1)$$

Other features like chromagram, melspectrogram, spectral contrast, tonnetz are extracted as well from the audio so that the emotion can be determined efficiently. In order to be able to do this, there one such package in python known as Librosa. Not only does it serve as a necessary building element for the creation of music information retrieval systems, but also aids in the visualization of audio signals as well as feature extraction utilizing various signal processing techniques. With the Librosa module we loaded every single .wav file and extracted the MFCC features for each of them. We then computed the mean of these features, obtaining 20 MFCCs for each audio file, and inserted these features in a dataframe along with the *Speaker Label*.



| | Feature_1 | Feature_2 | Feature_3 | Feature_4 | Feature_5 | Feature_6 | Feature_7 |
|---|---|---|---|---|---|---|---|
| **0** | -385.156860 | 71.944611 | 7.413402 | 22.166933 | 2.377141 | 9.545807 | -16.426249 |
| **1** | -359.441620 | 81.222977 | 6.164223 | 23.607706 | 5.008371 | 10.045403 | -18.296900 |
| **2** | -375.832306 | 70.309860 | 7.217519 | 24.220745 | 6.312408 | 8.401887 | -15.429814 |
| **3** | -371.408173 | 80.817154 | 4.081253 | 24.819365 | 5.732226 | 9.490652 | -17.379173 |
| **4** | -374.229797 | 80.690834 | 7.608641 | 24.339098 | 5.585734 | 7.855001 | -15.432492 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **329** | -351.618317 | 76.168686 | 2.060905 | 21.039740 | -15.909898 | -4.961066 | -19.788092 |
| **330** | -342.642181 | 70.671944 | 0.565506 | 25.364042 | -13.452065 | -6.592535 | -20.165661 |
| **331** | -328.729492 | 66.080299 | -0.306251 | 23.945536 | -10.330803 | -9.340878 | -19.554247 |
| **332** | -365.180756 | 67.974030 | 6.701675 | 28.228445 | -11.614446 | -3.229740 | -17.137144 |
| **333** | -409.124817 | 47.311668 | 1.173458 | 13.396432 | -9.119048 | -2.596071 | -11.705655 |

334 rows × 21 columns

**Figure 1: Sample of the dataframe containing 20 MFCCs for each speaker**

### *3.2.2 Spectrograms*

Models that usually take images as input, such as CNN, can also be used to work with audio data. In order to do so, we need to compute the spectrogram of each .wav file and use this image as input into our model. Spectrograms are a visual representation of the spectrum of frequencies of a signal as it varies with time. Sounds can be represented as waves, and waves have two important properties: frequency and amplitude. The horizontal direction in a spectrogram of an audio clip represents time, and the vertical direction represents different frequencies. Finally, the amplitude of sounds of a specific frequency that exist at a specific point in time is represented by the point's color, which is determined by the corresponding x-y coordinates. For this study, we used *Pylab* module to create spectrograms for each audio file from all the eight different speakers, which we then used as input data for our model. For the spectrograms, we changed the NFFT value from the default 256 to 51200. The NFFT value of 256 takes Fourier Transforms over 0.01 second of audio, which shows it is highly dependent on what the speaker is saying but it is not what we are concerned with. We are mainly concerned with the characteristics of the speaker's voice for "speaker classification", hence extending the time frame by increasing the NFFT value. The spectrogram images initially had size of $256 * 256$, but we reduced them to $64*64$. Below is the representation of spectrograms obtained from one single audio file from each speaker.

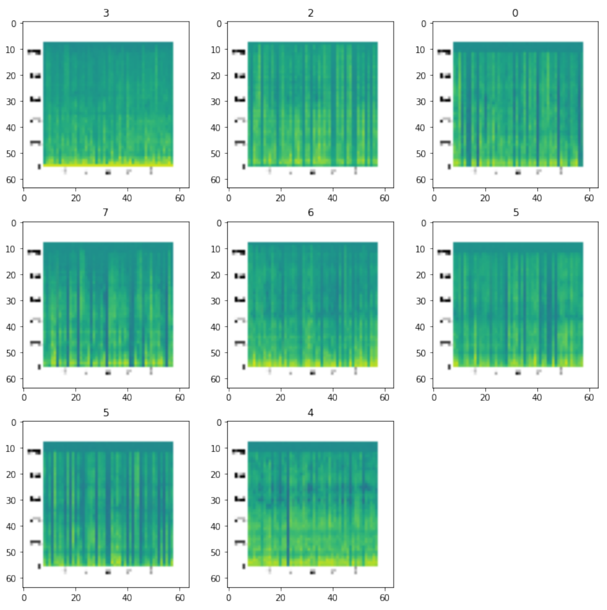


**Figure 2: Spectrogram samples from the eight different speakers**

## 3.3 ANN

The first model we implemented for the classification part of this project is an Artificial Neural Network. Because the MFCCs were the input of this model, we added an input layer to the Neural Network of input size equal to 20, which is the number of the features. We then added two hidden layers, both incorporating a *Relu* activation function, setting the size parameter to 40. The *Relu*, Rectified Linear Activation Function, is a linear function that outputs the input directly if it is positive, otherwise, it outputs zero. We set the parameters of final output layer of this model to 8 as the output size, referring to the eight different classes of this classification problem, and *Softmax* as the activation function. The *Softmax* function turns numbers into probabilities that sum to one. It outputs an array that represents the probability distributions of a set of potential outcomes. Because it is a multi-class classification task, we decide to use *Categorical Crossentropy* as the loss function, and the *Adam* optimizer. The *Adam* optimizer is a replacement optimization algorithm for the stochastic gradient descent in deep learning models. *Adam* combines the best features of the AdaGrad and RMSProp algorithms to create an optimization algorithm that can deal with sparse gradients on noisy problems. After each hidden layer we also added a Dropout Layer to avoid overfitting, setting the parameter to 0.2. When fitting the model to our training dataset, containing the MFCC features, we set the batch size to 40 and run the model for 1000 iterations.

## 3.4 CNN

As previously mentioned, it is possible to use Deep Learning algorithms, which typically work with images, for audio recognition problems. In this alternative approach for the classification part of our problem, we transformed all the audio files into spectrograms, and used these images as input for a Convolutional Neural Network Model. For our model,

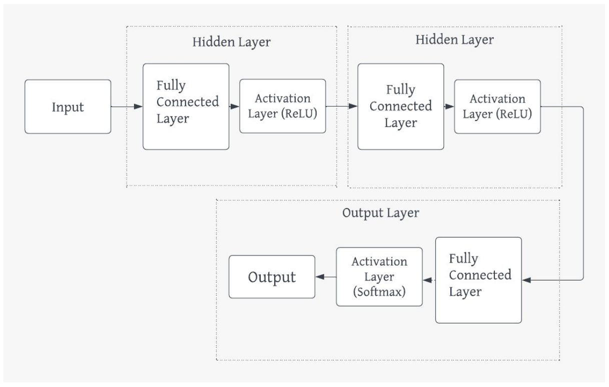


**Figure 3: Block diagram of our ANN**

we decided to include two convolutional layers, two pooling layers, one dense layer and one final fully connected layer before obtaining the output. For both convolutional layers we used a kernel of spacial size $11 * 3$, with strides equal to $(1, 3)$, and size as 16 in the first one and 32 in the second one. The activation function used in these two layers was $Relu$, a linear function that outputs the input directly if it is positive, otherwise, it outputs zero. For both pooling layers, we used Max Pooling operation with pool size of $4 * 2$. After each convolutional and pooling layer, we also added a Batch Normalization layer, which standardizes the inputs for each mini batch. Before the output layer we also included a flattening layer and another dense layer of size 64, incorporating a $Relu$ activation function. The flattening layer converts the input into a 1-dimensional array before outputting it to the next layer. Finally, we added a dropout layer to avoid overfitting, setting the dropout parameter to 0.2. The final output layer has, as the output size parameter, the number of classes we are trying to predict, and as activation function a $Softmax$, which outputs an array that represents the probability distributions of a set of potential outcomes. The loss function used for this model was the $Sparse\ Categorical\ Crossentropy$, and the $RMSprop$, Root Mean Squared Propagation, as optimizer. When fitting the model to our training dataset, all the spectrograms obtained from the audio samples, we set the learning rate to 0.0001 and run the model for 100 iterations.

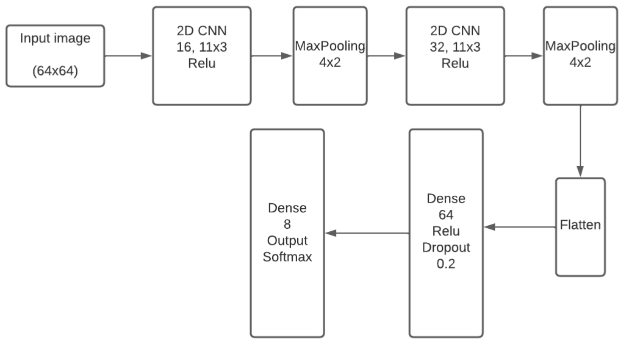


**Figure 4: Block diagram of our CNN model**

## 3.5 FastICA

After training the classification models, we moved to the Blind Source Separation problem. For the second part of this study, we decided to create our own mixed signals, by simply taking one audio sample from each speaker, and combine them all together using a randomly generated mixing matrix. By using eight different audio files and a matrix with dimensions $8 * 8$, we obtained eight different mixed signals, all with different weights associated to each speaker, and we fed these signals to our BSS algorithm, the FastICA. We used the FastICA algorithm from the $scikit - learn$ python module and set the number of components to 8. From the implementation of this algorithm on the mixed signals, we obtained eight different unmixed components, which represent the eight different speakers. The problem with FastICA is that it doesn't tell us which speaker is associated with each independent component, but only the number of speakers that were originally talking in the mixed audio file and separate those individual signals.

## 3.6 Combining the Models

The main novel part of this study is precisely the combination of the Separation and Classification models together. After training the classification model on all the individual audio samples, and obtaining independent components from the mixed audio signals with the FastICA, we wanted to classify each independent component and associate it to one of the eight speakers. In order to do so, a little pre-processing of all the independent components was necessary. We created audible audio files for each one of them, and then used again the $librosa$ module to extract the MFCC features from each one of them. We computed the mean of all MFCCs, exactly like we did in the training our model, and obtained an array of dimensions $(1, 20)$ for each audio sample, which we then inserted into a dataframe, this time without adding the Speaker's label in the last column. After this pre-processing step, we had a dataframe sized $8x20$. We took the classification parameters obtained from training the model and applied them to all the independent components dataframe resulted from the BSS technique, in order to predict the speaker talking in each independent audio file. The result was a 1-dimensional array, of size $(1, 8)$, with numbers ranging from 0 to 7. As an example, the matrix below shows the one-hot encoding representation of the mentioned array, which represents the association between the independent components (on the y-axis) and the speakers (on the x-axis). Independent component 0 is associated to speaker 7, independent component 1 is associated to speaker 3, independent component 2 is associated to speaker 5, and so on.

## 4. RESULTS

The results obtained from each of the models were promising. Specifically, the Artificial Neural Network classification model was the one that gave us better results in terms of performance. The table below shows the results obtained for the evaluation metrics of Accuracy, Precision, and Recall.

| | Accuracy | Precision | Recall |
|---|---|---|---|
| Training Set | 1 | 1 | 1 |
| Validation Set | 0.97 | 0.97 | 0.97 |

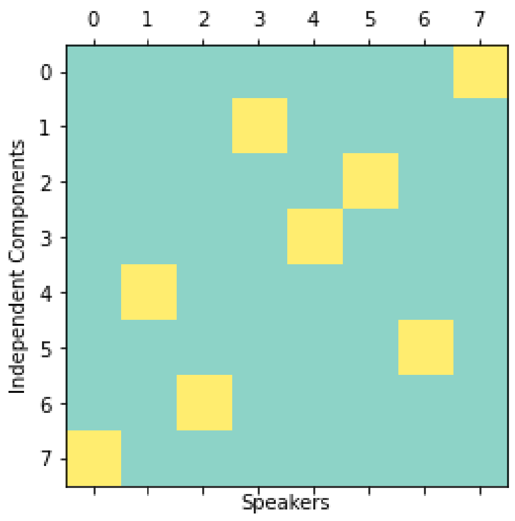


**Figure 5: Matrix representing the matching between independent components and speakers**

On the other hand, the Convolutional Neural Network gives us good results in terms of accuracy but, by looking and the plot below, it is clear that, while the training accuracy increases at a consistent rate until it reaches 100% after about 50 epochs, the validation accuracy is not improving at a steady pace at every iteration, but it is drastically changing (either increasing or decreasing) at every epoch. Even with these unstable rapid changes, the overall behavior of the curve is positive, and the average accuracy increases after 100 iterations, obtaining a final accuracy of 90.91%. This curve highlights the limitation of our CNN model, which might be overfitting on the training set. This problem could be caused by the fact that our training dataset, the set of spectrograms from the audio files, is not large enough for the CNN, and we would need more samples in order to obtain a better performance.

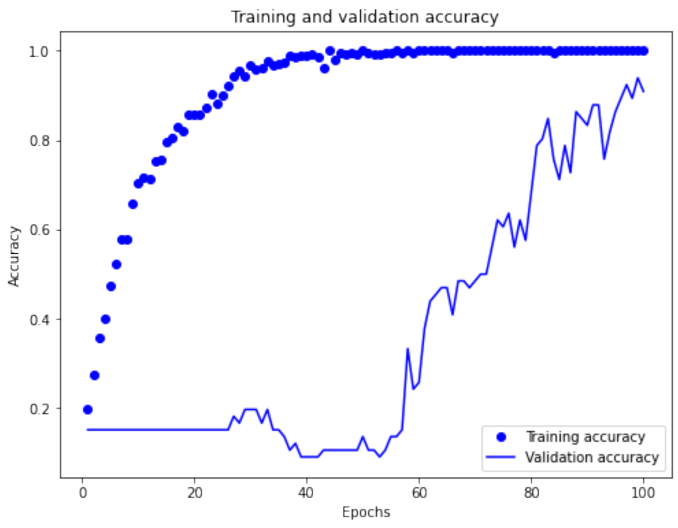


**Figure 6: Curve of Training and Validation Accuracy**

Because of the better results obtained from the ANN classification model, rather than from the CNN, we decided to move forward towards the final part of this project using this algorithm.
After the implementation of the FastICA, we had eight independent components that we needed to match to their corresponding speaker. We managed to do this last step of the Speakers Recognition problem by applying the parameters obtained from the ANN to the Independent Components dataset, and we showed the results in the matrix in *Figure* 3. However, this matrix did not tell us precisely if our predictions were correct or not, therefore, based on the matches shown in the matrix, we decide to plot the frequency wave of each independent component and the one of their corresponding speaker's audio file. As we can see in *Figure* 5, all the waves on the left match the ones on the right, stating that our model predictions were exact.

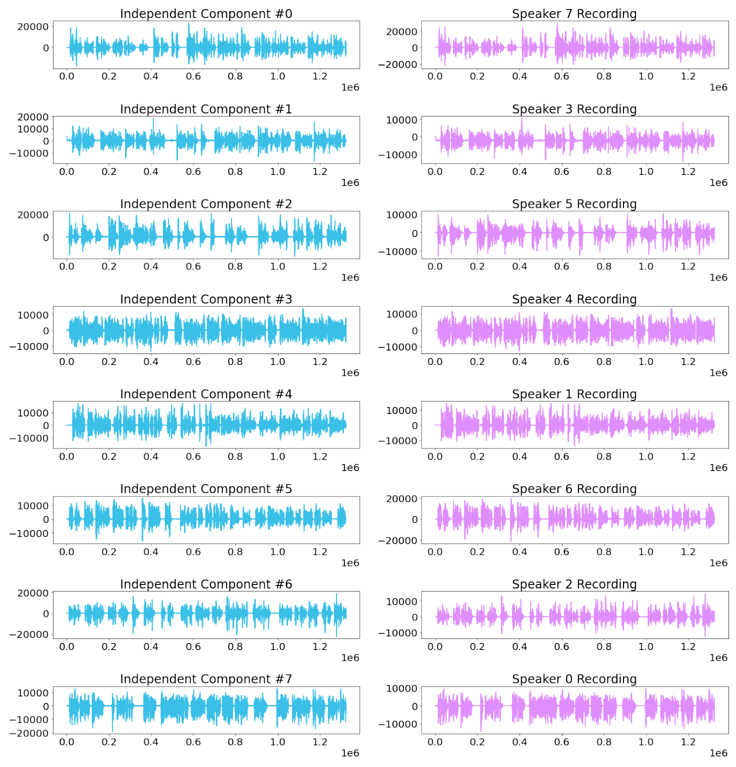


**Figure 7: Wave Plots of the Independent Components and their corresponding Speakers**

## 5. CONCLUSION

The objective of our project was to separate an audio of mixed signals from different speakers (8 speakers in our case) using FastICA and then feeding the obtained separated signals to our trained Artificial Neural Network and Convolutional Neural Network in order to classify the speakers. On comparing the two models we observed that our ANN model performs better than our CNN model with an accuracy of 97% whereas the CNN model was not very consistent with the accuracy.

## 6. LIMITATIONS AND FUTURE SCOPE

When working with our model we realized one constraint is the size of our dataset because of which our CNN model did not exactly perform the way we had expected it to. Each speaker has around 30-50 audio files, which means in total we have 240-400 spectrograms and that is not enough to train and test a model. On further analysis we realized that melspectrograms are better than spectrograms. The linear audio spectrogram is best for applications in which all frequencies are equally important, whereas melspectrograms are preferable for situations in which human hearing perception must be modeled. Audio classification programs benefit from Mel spectrogram data. Hence, in the future, instead of using spectrograms, melspectrograms can be used in order to train the CNN model. We could also train a Recurrent Neural Network to classify the speakers as they have shown great results as well [12, 1]

## 7. REFERENCES

[1] A. Amberkar, P. Awasarmol, G. Deshmukh, and P. Dave. Speech recognition using recurrent neural networks. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–4, 2018.

[2] M. Ayache, H. Kanaan, K. Kassir, and Y. Kassir. Speech command recognition using deep learning. In *2021 Sixth International Conference on Advances in Biomedical Engineering (ICABME)*, pages 24–29, 2021.

[3] Z. Bai and X.-L. Zhang. Speaker recognition based on deep learning: An overview. *Neural Networks*, 140:65–99, 2021.

[4] J.-F. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, 1998.

[5] S. Hourri, N. S. Nikolov, and J. Kharroubi. A deep learning approach to integrate convolutional neural networks in speaker recognition. *International Journal of Speech Technology*, 23(3):615–623, 2020.

[6] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[7] H. Lee, P. Pham, Y. Largman, and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[8] W. Lim, D. Jang, and T. Lee. Speech emotion recognition using convolutional and recurrent neural networks. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–4, 2016.

[9] T. J. Sefara and T. B. Mokgonyane. Emotional speaker recognition based on machine and deep learning. In *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–8, 2020.

[10] J. V. Stone. Independent component analysis: an introduction. *Trends in cognitive sciences*, 6(2):59–64, 2002.

[11] J. V. Stone. Independent component analysis: a tutorial introduction. 2004.

[12] R. Venkateswarlu, R. V. Kumari, and G. V. JayaSri. Speech recognition by using recurrent neuralnetworks. *International Journal of Scientific & Engineering Research*, 2(6):1–7, 2011.

[13] L. Yang, Z. Ming, and J. Longbin. Blind source separation based on fastica. In *2009 Ninth International Conference on Hybrid Intelligent Systems*, volume 2, pages 475–479, 2009.