TinyCNN: An Embedded CNN Model for Speaker Identification Using ESP32

3 authors:

Hatem Zehir
Badji Mokhtar-Annaba University
15 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Hafs Toufik
Badji Mokhtar-Annaba University
38 PUBLICATIONS **104** CITATIONS
SEE PROFILE

Sara Daas
Badji Mokhtar-Annaba University
18 PUBLICATIONS **110** CITATIONS
SEE PROFILE

TinyCNN: An Embedded CNN Model for Speaker Identification Using ESP32

Hatem Zehir
L.E.R.I.C.A Laboratory
Badji Mokhtar University
Annaba, Algeria
hatem.zehir@univ-annaba.dz

Toufik Hafs
L.E.R.I.C.A Laboratory
Badji Mokhtar University
Annaba, Algeria
hafstoufik@gmail.com

Sara Daas
L.E.R.I.C.A Laboratory
Badji Mokhtar University
Annaba, Algeria
sara.daas@yahoo.com

***Abstract*—This research paper introduces a novel Convolutional Neural Network (CNN) architecture optimized for speaker recognition on resource-constrained devices. The proposed model was evaluated on a subset of 10 speakers from the LibriSpeech database. To process the audio signals, preprocessing techniques to remove silent parts and segmented the signals into 1-second windows with a 500 ms overlap were employed. The Mel Frequency Cepstral Coefficients (MFCC) of each window were then extracted and fed into the CNN model. After training, the CNN model was quantized, converted to TensorFlow Lite (TFLite) format, and integrated into an Arduino-compatible library. This deployment was successfully executed on the ESP32 board. The proposed quantized model achieved an impressive accuracy of 98.58%, outperforming the original model by being 53% faster and 73% smaller in size. The findings of this study prove the efficiency of deploying accurate and fast speaker recognition systems on edge devices.**

***Index Terms*—Speaker Identification, Mel Frequency Cepstral Coefficients, Convolutional Neural Network, TinyML, ESP32**

I. Introduction

Speaker recognition is a challenging problem in the biometric field. It is the process of automatically recognizing a user based on specific features of his voice. Speaker recognition is used in many applications, such as access control and security control.

There are two major categories of speaker recognition tasks: speaker identification [1], [2] and speaker verification [3], [4] as shown in Fig. 1. Speaker identification is the process of finding the identity of a speaker from a database of known speakers. Speaker verification is the process of checking a speaker's claimed identity by comparing its voice characteristics to those stored in a database.

Both speaker identification and speaker verification can be further classified into text-independent [1] and text-dependent [4] tasks. Text-independent speaker recognition does not require the user to say any specific word, sentence, or password. It focuses on voice characteristics. Text-dependent speaker recognition, on the other hand, expects the speaker to say a specific predefined word or sentence. This type is more common in security systems.

Text-independent and text-dependent speaker recognition systems can also be classified into open-set and closed-set systems. Open-set speaker identification is a type of speaker identification where the system is required to recognize the identity of a speaker from a large group of individuals, including those who are not present in the original database used to train the system. Closed-set speaker identification is a type of speaker identification where the system is only required to identify a known speaker. Unlike open-set speaker identification, the system does not need to be able to identify speakers who are not in the training database.

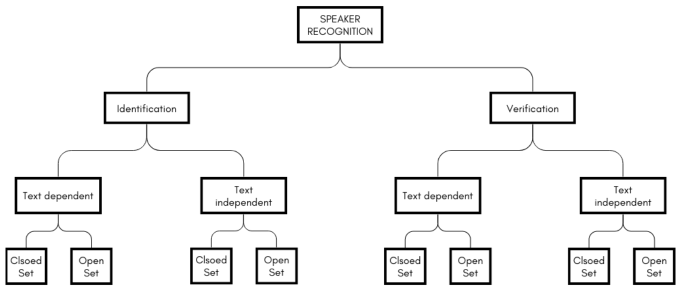


Fig. 1. Schematic Representation of the Different Speaker Recognition Types

This paper focuses on developing a novel closed-set text-dependent speaker identification system. Deep learning has been widely explored for speaker identification, Prachi et al. [5] proposed a DNN-based model for speaker identification. They started by extracting spectrograms from audio files of both the TIMIT [6] and LibriSpeech [7] databases using Mel Frequency Cepstral Coefficients (MFCC). The authors studied both open-set and closed-sets scenarios, To evaluate their proposed approach, they used two deep learning architectures, one based on a convolutional neural network (CNN) and the other one on a long short-term memory network (LSTM). Using CNN, the authors were able to achieve the best accuracy of 97.85%, while they achieved the best accuracy of 71.54% using LSTM. In another paper, Bunrit et al. [8] proposed a CNN-based Text-Independent Speaker Identification system, they extracted the spectrogram images of 2 seconds segments of wav audio files collected from YouTube videos and fed them directly to a CNN model, they were able to achieve an average accuracy of 95.83%. In a third paper, Ye and Yang [9] proposed a deep learning model with CNN and gated recurrent unit (GRU) for speaker identification, the CNN layer was used for feature extraction and dimensionality reduction and the

GRU layer is used for classification. The proposed model was tested on the Aishell-1 speech dataset [10], and it achieved 98.96% accuracy.

Upon reviewing the literature, we can notice that despite deep learning for speaker identification being very popular, there is still a lack of optimizing deep learning models for constraint devices such as microcontrollers. This is due to many factors, including the large size of the trained deep-learning models, which makes them hard to fit in the small memory of the embedded devices, and also the heavy computational power needed for the inference which is a problem for a device with limited processing power.

To overcome the previous problems, this paper attempts to optimize a CNN-based deep-learning model for closed-set text-independent speaker identification and fit it in the small memory of a constrained device. The proposed approach mainly focuses on quantizing a regular model into an INT 8 bits format, which will reduce its size significantly and also speed up the inference time while keeping a high identification accuracy. The model is then deployed and tested on an ESP32 microcontroller.

The rest of this paper is organised as follows: in Section 2, the proposed system is presented. Section 3 discusses the experimental findings, and finally, Section 4 presents the conclusion and the prospects of this paper.

II. Proposed Method

The proposed method as shown by Fig. 2 starts with preprocessing the voices to eliminate any silent part from the signals and dividing them into one-second windows with a 500 ms overlap. Next, we extract the spectrograms using MFCC. These extracted spectrograms are then fed as features into a CNN model. To optimize for resource-constrained devices, we convert the trained CNN model into a TensorFlow Lite format and perform 8-bits quantization. Finally, to deploy the model on an ESP32 microcontroller, we convert the quantized model into an Arduino library.

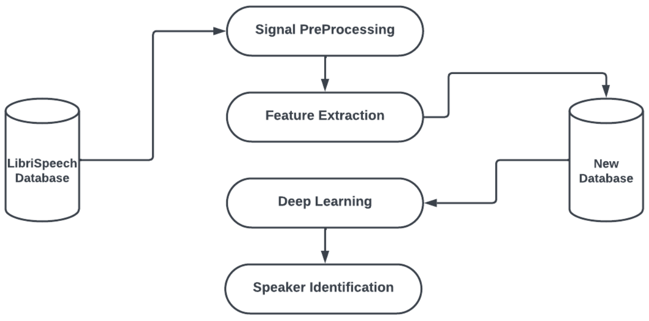


Fig. 2. Proposed System Diagram for the Embedded Speaker Recognition System

The model

A. Database Used

In speaker recognition tasks, there are two major types of databases, clean [7], [11] and in-the-wild [12] databases. Clean databases are a collection of high-quality audio recordings of speakers with minimal background noise levels and controlled recording conditions. In-the-wild databases are the most challenging database type for speaker recognition tasks, as the audio signals are recorded in various uncontrolled environments and have varying background noise levels.

The LibriSpeech dataset was chosen for the evaluation of the proposed method. It consists of a large collection of around 1,000 hours of recordings collected from English audiobooks and sampled at 16 kHz. The audio files in this database are divided into two categories: clean and other. Due to the hardware limitations of the ESP32 and to avoid any complicated data processing, only 10 subjects (1,189 audio files) from the clean subcategory were used to train the proposed deep learning model.

B. PreProcessing

To preprocess the audio files and prepare them for further analysis, we first remove any silent parts. This is done by thresholding the audio signal and removing any segments that are below a certain amplitude. Once the silent parts have been removed, we segment the audio into 1-second windows with 500 ms overlap. This means that each window will overlap with the previous window by 500 ms.

1) Silence Removal: The silence removal stage starts by converting the amplitude of the sounds into decibel values. Once this is done, the mean, which is the average decibel value of the audio file, is calculated according to the following:

$$mean_{db} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

After that, the standard deviation, which is a measure of how spread out the decibel values are, is calculated according to this equation:

$$std_{db} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{2}$$

The threshold for the silent parts is calculated by subtracting the standard deviation from the mean as shown in (3). This means that any sound that is below the threshold is considered to be silent.

$$threshold_{db} = mean_{db} - std_{db} \tag{3}$$

The resulting audio signal after the silence removal step is shown in Fig. 3.

2) Segmentation: After silence removal, the audio files are segmented into 1-second windows with 500 milliseconds of overlap. The segmentation is done using a sliding windows algorithm. It starts by putting a sliding window of 1 second at the beginning of the audio file, after that it moves forward

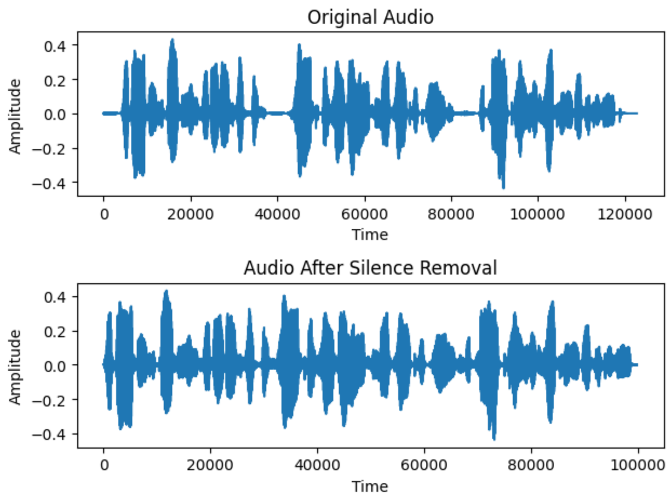


Fig. 3. Audio Before and After Silent Parts Removal

by steps of 500 ms, and each window is saved as a separate segment.

C. Feature Extraction

After pre-processing the audio signals, we use MFCC to extract the most relevant information about the frequency content of the audio signals, and their calculation process can be summarized as follows:

- Pre-Emphasis: this part involves applying a high pass filter to emphasize higher frequencies.
- Framing: the analysed audio is divided into overlapping frames.
- Windowing: after the framing step, there will be some discontinuous effect on the signal, the aim of windowing is to remove that effect by multiplying each frame by a windowing function, such as the hamming frame.
- Fast Fourier Transform: each frame is transformed into the frequency domain by applying the FFT.
- Mel Filterbank: A set of triangular filters applied on the frequency spectrum to simulate the reaction of the human auditory system to different frequencies.
- Log10: this is done to convert the magnitude values into the logarithmic scale.
- Discrete Cosine Transform: The final step consists of applying the DCT to obtain the MFCC coefficients.

Fig. 4 shows a sample MFCC extracted from a preprocessed audio file of a random speaker from the LibriSpeech database.

D. Proposed Deep Learning Architecture

The generated MFCCs are fed as features to a CNN-based model. The model consists of two CNN layers with 8 filters each of size 3x3 and a ReLU activation, after each CNN layer, a dropout of 25% is added to prevent overfitting, before the classification layer, a Flatten layer is used to convert the 2D feature maps into a 1D vector. The full architecture of the deep learning model is shown in table I.

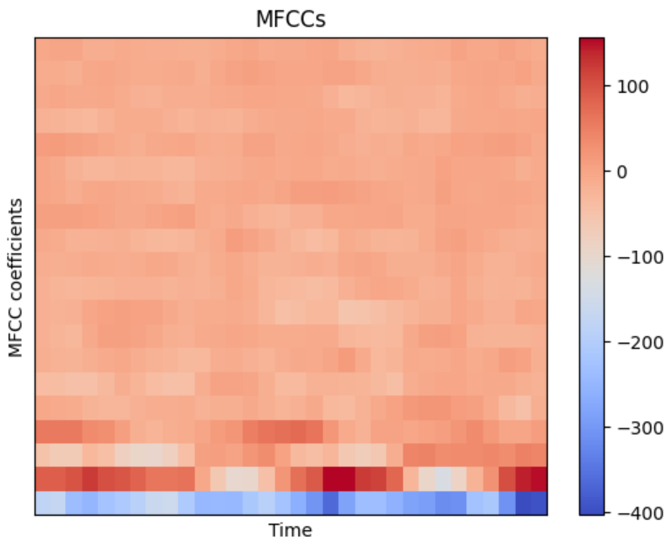


Fig. 4. MFCC coefficients of a randomly selected audio segment.

TABLE I
THE ARCHITECTURE OF THE NEURAL NETWORK USED

Layer	Output Shape	Param. #
Conv2D	(None, 12, 32, 8)	80
Dropout	(None, 12, 32, 8)	0
Conv2D	(None, 12, 32, 8)	584
Dropout	(None, 12, 32, 8)	0
Flatten	(None, 3072)	0
Dense	(None, 10)	30730

E. INT8 Quantization

Inference, the time a deep learning model takes to classify input data, requires a lot of memory and computational resources. This makes deploying a model on devices with limited resources a challenging task. To overcome this, the model size needs to be reduced and the performance should be improved to get a better inference time. To do so, a quantization is applied to the proposed model described is section II-D. Quantization is the process of rounding the values in the model to the nearest multiple of a fixed step. This reduces the number of bits needed to represent the values and will significantly reduce the model size. In other words, a quantized model is a lower-precision representation of the original model. For example, a model represented as 8-bit integers or 16-bit float helps to leverage some of the hardware limitations of edge devices.

The weights and activations of the proposed model are represented as a 32-bit floating point which takes up 4 bytes of memory. To reduce memory usage, an 8-bit integer quantization is applied, as 8-bit integers need only 1 byte of memory. This means that quantizing the 32-bit floating point model to an 8-bit integer can reduce the model size by a factor of 4.

In addition to reducing the model size, the inference time will also be reduced as quantization improve the performance because the calculations will be simpler and performed efficiently on a device with constrained resources.

F. Conversion to a TFlite Format

After a model is quantized, it is converted to a TFLite format. TFLite is a lightweight format for machine and deep learning models. It is designed for embedded and edge devices and based on TensorFlow which is a popular Python framework, but it is optimized for smaller size and faster performance. Making it ideal for these small and less powerful devices. The simpler format of the TFLite model makes it easier to deploy on a hardware-constrained device such as the ESP32

G. Arduino Library Generation and Deployment on ESP32

For the Arduino library generation, the Edge Impulse Framework [13] is used. It is a platform that enables the deployment of deep learning models on edge devices. The quantized model is uploaded into this framework and it was efficiently converted to an Arduino-compatible library. Arduino IDE is then used to develop the code that runs the inference and classifies the extracted audio features and recognizes speakers as discussed in section III.

The ESP32 board is used in its standalone configuration without any additional components. After training the model, it was deployed on the board alongside data from the LibriSpeech database to test its performance on the microcontroller.

III. Results and Discussion

For the evaluation of the proposed approach, a subset of 10 speakers from the LibriSpeech database is used. Before quantization and deployment of the deep learning model on the ESP32 as described in section II, the model was trained on a more powerful machine with a GeForce GTX1650 graphics card with 4GB of GDDR6 memory, 8GB of DDR4 RAM, and an Intel Core i3-10100F processor. The model was developed using Python and the two reliable frameworks Keras and TensorFlow and compiled with the appropriate hyperparameters, which included using sparse categorical cross-entropy as the loss function, Adam as the optimizer, and a learning rate of 0.0001.

Following the preprocessing of audio files and extraction of MFCC coefficients, the data was divided into three sets: 70% for training, 10% for validation, and 20% for testing. During training, The model is trained for 50 epochs, and the data was divided into mini-batches of 32 samples. After each epoch, the model's performance was validated. The training curves are shown in Fig. 5.

The accuracy metric was used to evaluate the model, and it can be calculated according to (4). The unoptimized model has an accuracy of 98.63%, while the quantized model achieved an accuracy of 98.58% which outperforms state-of-the-art methods as shown in table II.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

The fact that the accuracy achieved by the quantized model accuracy is nearly identical to that of the unoptimized model

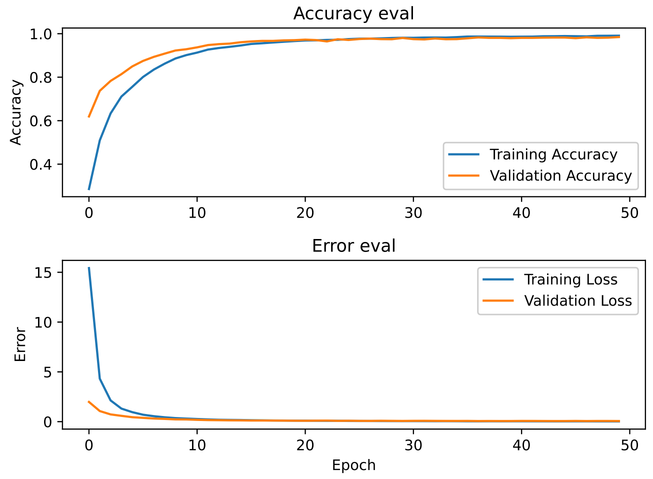


Fig. 5. Accuracy and Loss Plots for Both Training and Validation Sets

TABLE II
Comparison with other state-of-the-art methods

Method	Database	Accuracy Achieved
[14]	Private	98.34%
[5]	LibriSpeech	97.85%
[15]	LibriSpeech	91.19%
Proposed method	LibriSpeech	98.58%

demonstrates the effectiveness of the quantization process in significantly improving the model's performance and size while maintaining its accuracy. These results are also supported by the confusion matrix shown in Fig. 6, the model was able to identify the 1st, 6th, and 9th speaker with 100% accuracy. We also notice that only 58 sound features are misclassified from a total of 4235. This is proof of the robustness of the proposed model.

ESP32 is a system-on-a-chip (SoC) microcontroller, it has a number of hardware characteristics that make it well-suited for TinyML applications including a Dual-core Xtensa LX6 microprocessor running at a maximum frequency of 240 MHz, 320 KiB of SRAM, and 4MB of flash memory. In order to evaluate the proposed model on this board, inference time, model size, and RAM usage were taken into consideration as shown in table III.

TABLE III
A performance comparison between the quantized and unoptimized models.

	Inference time (ms)	Model Size (KB)	RAM Usage (KB)	Accuracy
Quantized (int8)	30	34.33	24.55	98.58%
Unoptimized (float32)	64	125.16	24.73	98.63%

Table III show that the quantization of a deep learning model can significantly reduce its inference time and model size while maintaining a nearly identical accuracy. In this case, the quantized model is 53% faster than the original model

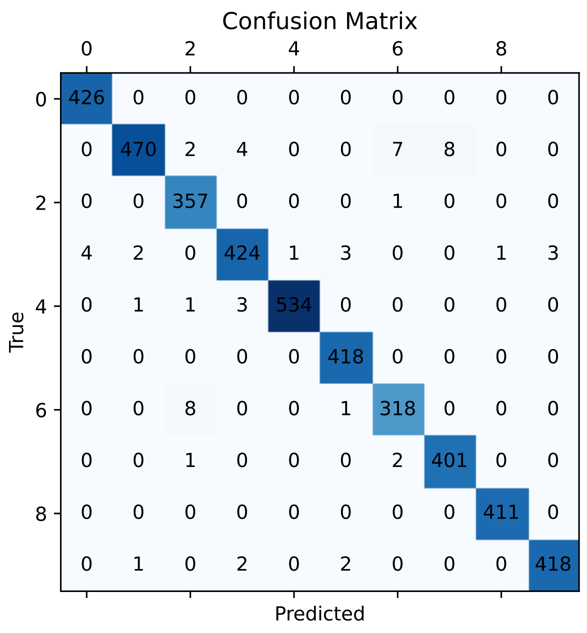
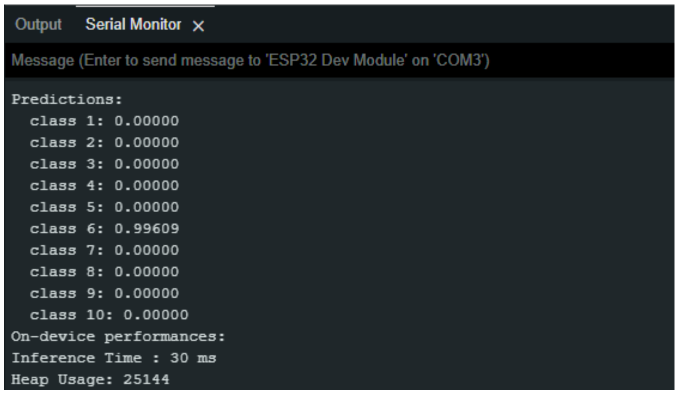


Fig. 6. Confusion Matrix of the Unoptimized Model


Output
Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')

```
Predictions:
  class 1: 0.00000
  class 2: 0.00000
  class 3: 0.00000
  class 4: 0.00000
  class 5: 0.00000
  class 6: 0.99609
  class 7: 0.00000
  class 8: 0.00000
  class 9: 0.00000
  class 10: 0.00000
On-device performances:
Inference Time : 30 ms
Heap Usage: 25144
```

Fig. 7. Classification Results Shown on the Serial Monitor of the Arduino IDE.

and 73% smaller in size, while the accuracy is only 0.05% lower. This means that the quantized model can be easily deployed on resource-constrained devices such as the ESP32 without sacrificing accuracy. This demonstrates the feasibility and effectiveness of the proposed approach to improving the on-device performance of a deep learning model on edge devices while keeping the same classification accuracy.

The classification output of the quantized model is shown in Fig. 7. It shows that the model correctly classified MFCC coefficients belonging to the 6th speaker result with 99.609% as a level of confidence.

IV. Conclusion

This paper presented a novel optimized CNN architecture for speaker recognition. The proposed model was evaluated on a subset of 10 speakers extracted from the LibriSpeech database. The audio signals were first preprocessed by removing silent parts and then segmented into 1-second windows with a 500 ms overlap. Afterward, the MFCC coefficients of each window were extracted and fed into a CNN model. The trained CNN model was quantized, converted to TFLite format, and transferred to an Arduino-compatible library. Finally, the model was successfully deployed onto the ESP32 board. The proposed quantized model achieved an accuracy of 98.58%, and it was 53% faster and 73% smaller than the original model. Future work will focus on improving the system by adding another modality, such as fingerprint data, to create a multimodal biometric system.

References

[1] R. Jahangir, Y. W. Teh, N. A. Memon, G. Mujtaba, M. Zareei, U. Ishtiaq, M. Z. Akhtar, and I. Ali, "Text-independent speaker identification through feature fusion and deep neural network," *IEEE Access*, vol. 8, pp. 32187–32202, 2020.

[2] R. Jahangir, Y. W. Teh, H. F. Nweke, G. Mujtaba, M. A. Al-Garadi, and I. Ali, "Speaker identification through artificial intelligence techniques: A comprehensive review and research challenges," *Expert Systems with Applications*, vol. 171, p. 114591, 2021.

[3] T. Zhou, Y. Zhao, and J. Wu, "Resnext and res2net structures for speaker verification," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 301–307, IEEE, 2021.

[4] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5115–5119, IEEE, 2016.

[5] N. N. Prachi, F. M. Nahiyan, M. Habibullah, and R. Khan, "Deep learning based speaker recognition system with cnn and lstm techniques," in *2022 Interdisciplinary Research in Technology and Management (IRTM)*, pp. 1–6, IEEE, 2022.

[6] J. S. Garofolo, "Timit acoustic phonetic continuous speech corpus," *Linguistic Data Consortium, 1993*, 1993.

[7] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 5206–5210, IEEE, 2015.

[8] S. Bunrit, T. Inkian, N. Kerdprasop, and K. Kerdprasop, "Text-independent speaker identification using deep learning model of convolution neural network," *International Journal of Machine Learning and Computing*, vol. 9, no. 2, pp. 143–148, 2019.

[9] F. Ye and J. Yang, "A deep neural network model for speaker identification," *Applied Sciences*, vol. 11, no. 8, p. 3603, 2021.

[10] X. N. B. W. H. Z. Hui Bu, Jiayu Du, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *Oriental COCOSDA 2017*, p. Submitted, 2017.

[11] W. Fisher, "The darpa speech recognition research database: specifications and status," in *Proc. DARPA Workshop on speech recognition*, pp. 93–99, 1986.

[12] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[13] S. Hymel, C. Banbury, D. Situnayake, A. Elium, C. Ward, M. Kelcey, M. Baaijens, M. Majchrzycki, J. Plunkett, D. Tischler, A. Grande, L. Moreau, D. Maslov, A. Beavis, J. Jongboom, and V. Janapa Reddi, "Edge impulse: An mlops platform for tiny machine learning," 11 2022.

[14] S. Kadyrov, C. Turan, A. Amirzhanov, and C. Ozdemir, "Speaker recognition from spectrogram images," in *2021 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, pp. 1–4, IEEE, 2021.

[15] H. K. Pentapati and K. Sridevi, "Log-melspectrum and excitation features based speaker identification using deep learning," in *2022 International Conference on Industry 4.0 Technology (I4Tech)*, pp. 1–6, IEEE, 2022.