Implementation of speaker verification model on low-power hardware in real-time

Abstract

This research focuses on implementing a speaker verification (SV) model on low-power hardware in real-time. Developing neural network models for low-power devices is essential for applications in wearable technology, such as smartwatches, headphones, and smart assistant devices for individuals with disabilities. Additionally, there are potential applications in surveillance technology, including smart cameras, microphones, and smart locks. Running deep neural networks (DNNs) on resource-constrained devices, like microcontrollers used in Internet of Things (IoT) applications or edge devices without internet connection, poses significant engineering challenges due to limitations in operating frequencies, power consumption, and memory capacity [1]. SV models typically come with high computational costs, making them unsuitable for deployment on devices with limited resources [2]. Model compression techniques can help shrink the model size and reduce resource usage while maintaining performance as close as possible to the original model. This research examined two widely used SV models: ECAPA-TDNN and ResNet. We explored methods like changing architecture to reduce layers and parameter numbers, quantization ware-training (QAT), and knowledge distillation to compress the models. A tiny version of ECAPA without quantization, trained with knowledge distillation, achieved the highest performance among the tiny models and was implemented on an STM32L4R9I-DISCOVERY board-based STM32L4R9AI microcontroller. The board has no operating system; it contains 2 Mbytes of Flash memory and 640 Kbytes of RAM. To successfully implement a model on the microcontroller, reducing the model size to less than 2 Mbytes is imperative.
The distillation process improved the EER (equal error rate) by 13.5% compared to the undistilled version. The application works in real-time and compares the embedding vector of one second of audio captured from the microphone on the board after it passes the network with two enrolled speakers' voice signatures. A specific LED is illuminated if the cosine similarity score exceeds a predefined threshold, which indicates verification.

1. Introduction and Literature Review

1.1. Speaker Recognition (SR)

SR is the process of identifying a speaker based on the vocal features of their speech [3]. It can be divided into three subjects: Speaker Identification (SI), Speaker Verification (SV), and Speaker Diarization (SD). SI involves identifying unknown users based on their voice and answering the question of who is speaking. SV is the task of authenticating the claimed identity of a speaker based on some speech signal and enrolled speaker record and answering the question of whether the person speaking now is the enrolled speaker [4]. SD identifies a person's voice from a given population and determines when the speaker speaks. In the past few decades, traditional statistical methods such as GMM-UBM and i-vector model have been widely used for speaker verification until neural network-based models emerged [5]. The neural network employs a feedforward technique to convert variable-length utterances into fixed-dimension speaker embeddings [6]. One widely used model based on this architecture is known as X-Vectors [7].

ECAPA-TDNN [8] is another widely used speaker verification model based on the x-vectors model and incorporates two mechanisms: SE-Block [6] and attentive statistic pooling [5], along with residual connections and 1D convolution. Attentive statistic pooling leverages an attention

mechanism to assign varying weights to different frames, generating weighted means and standard deviations. This enables the model to capture long-term variations in speaker characteristics more effectively [5]. A speaker embedding model has been enhanced by replacing a pooling layer with an attentive statistic pooling layer. SE-Block is another method trying to improve the subtle differences in the learning capability of a speaker embedding system [9]. Besides those mechanisms, the authors utilized 80-dimensional MFCCs as input features, applied augmentations, and trained with the AAM-SoftMax function [10]. ResNet is another widely used SV model, which uses residual connections to improve the learnable ability of the network [11].

1.2. Compression methods

Neural network compression includes methods that reduce storage and speed up inference while minimizing loss in accuracy or performance. The methods can be categorized into eight groups [12]:

- Magnitude-Based Pruning Methods
- Similarity and Clustering Methods
- Sensitivity Analysis Methods
- Knowledge Distillation Methods
- Low-Rank Methods
- Quantization Methods
- Architectural Design Methods
- Hybrid Methods

Extensive research has been conducted in network compression to minimize the resource demands of CNNs, which are widely used in recognition systems, making them more suitable for deployment on devices with limited resources. Broadly, CNN compression techniques can be classified into four primary categories: low-rank approximation, knowledge distillation, network pruning, and quantization [12].
In [2], the researchers investigate weight quantization to compress DNN models for SV, specifically focusing on the ECAPA-TDNN and ResNet architectures. The research finds that weight quantization effectively reduces model sizes while maintaining performance. The experiments conducted on the VoxCeleb [13] dataset demonstrates that ResNet exhibits more robustness to low-bitwidth quantization than ECAPA-TDNN, particularly at 4-bit quantization. Additionally, the study evaluates the generalization ability of the quantized models using a language-mismatched SV task and employs information probing to assess retained speaker-relevant knowledge. The quantization model size in their study is between 3.13Mbytes and 6.96Mbytes.
In [14] the authors used a process where weights below a certain threshold are removed from the network. After removal, the network is fine-tuned, and this process is repeated until its accuracy decreases. They applied this iterative pruning method to AlexNet and VGG16 trained on ImageNet data. The results showed that it's possible to reduce the number of weights by a factor of 9 and 13 for AlexNet and VGG16, respectively, without sacrificing accuracy. The experiments also indicated that the earlier layers (closer to the inputs) are the most sensitive to pruning and that iterative pruning is more effective than one-shot pruning (which means pruning all at once after the model is trained).

Knowledge distillation is a learning framework that involves a teacher-student structure. In this framework, the teacher network, which is an efficient neural network or a collection of neural networks, provides its output as a label to train a smaller, more compact neural network known as the student network [15]. The knowledge obtained from the teacher network is defined as softened outputs, which differ from traditional one-hot labels. Softened outputs provide additional supervision of both intra-class and inter-class similarities learned by the teacher.

Unlike one-hot labels that map samples to singular points in the label space, softened labels project samples into a continuous distribution. This allows softened labels to capture intra-class variation by representing each sample according to its class distribution and comparing inter-class similarities among different classes. The soft target of a network T is defined by (1) $p_T^\tau = softmax(\frac{aT}{\tau})$ where α is the vector of teacher logits (pre-SoftMax activations) and τ is a temperature. Increasing τ retains inter-class similarity by preventing predictions from being too certain (0 or 1). The student network is trained using a combination of the softened and original SoftMax outputs [16]. Other loss functions can be used in knowledge distillation, like MSE and cosine similarity.

In [15] The authors combined iterative pruning, quantization, and knowledge distillation to compress speech enhancement networks. They used the original model as the teacher to fine-tune the pruned student model by combining the outputs of the teacher with the true labels. Additionally, they applied clustering-based quantization to reduce the neural network's size further. To evaluate their method, they selected two representative models, LSTM and Gated CRN, for speech enhancement. The experimental results demonstrate a significant reduction in model size by about 10x and 40x for LSTM and GCRN, respectively, with no significant performance loss.

Quantization involves converting floating-point numbers into fixed-point numbers with fewer bits [15]. Two mechanisms of quantization are commonly used: post-training quantization and quantization-aware training (QAT).

In post-training quantization, the neural network is trained using floating-point computation. After training, the network's parameters are quantized. This process introduces quantization errors, potentially leading to decreased accuracy.

In QAT**,** the model is trained while pretending to be already using these lower-precision numbers. This means that during the training process, the calculations are simulated as if using 8-bit integers, even though the actual training might still use 32-bit floating points for stability [17].

In [18], the researchers aimed to make running large networks on mobile devices more efficient by reducing the required storage and energy. They presented "deep compression," a three-stage pipeline designed to minimize storage needs without compromising the network's original accuracy. This involved pruning the network, quantizing weights, and leveraging Huffman coding to take advantage of the distribution of effective weights. With this method, the researchers compressed AlexNet and VGG-16 by 35x and 49x with no accuracy loss. In [1], the researchers introduced an open-source framework named MicroAI. This framework facilitates end-to-end training, quantization, and deployment of deep neural networks on microcontrollers. They aimed to create a framework that is easy to adapt and extend while maintaining a balance between accuracy, energy efficiency, and memory footprint. They presented comparative results using two microcontrollers and three frameworks: TensorFlow Lite for Microcontrollers, STM32 X-Cube AI, and their own MicroAI. The results were compared based on memory footprint, inference time, and power efficiency. The researchers used the ResNet architecture [11]. This allowed them to apply the same DNN to various data types, including time

series, audio, and images, thus simplifying the implementation.

In [19] The researcher aimed to evaluate various neural network models to identify the most suitable type for recognizing a Yaris car horn. The evaluation considered simple network types, including DNN and several CNN. The most suitable network was implemented on an STM32F746NG MCU for real-time audio input classification. The researcher utilized the STM32 X-Cube AI framework.

2. Data Description

Dataset of speakers:
VoxCeleb1 [13] Contains over 100,000 utterances from 1,251 celebrities.
VoxCeleb2 [20] Contains over 1 million utterances from over 6,000 celebrities. The utterances were extracted from videos uploaded to YouTube. The dataset is gender-balanced, with 55% of the speakers male (voxceleb1) and 61% male (voxceleb2). The speakers span a wide range of different ethnicities, accents, professions, and ages. In this research, we used VoxCeleb1. Dataset for augmentation: room impulse responses (RIR) for creating reverberation [21], for additive noise: MUSAN dataset, which consists of over 900 noises, 42 hours of music from various genres, and 60 hours of speech from twelve languages [22].

3. Contribution

This project incorporates theoretical and practical parts.

Compression speaker verification model into a very compressed model of less than 2Mbytes and deploying it on low-power hardware in real-time is something we didn't find in the literature.

The development of neural network models for low-power devices falls under an ongoing research area known as TinyML, and this project is a part of that field.

4. Methods

The research consists of two main parts. The first part involves compressing and modifying the models to enable hardware implementation. The second part focuses on programming the application of speaker verification to function in real-time on low-power hardware. To implement the speaker verification model on the hardware, we needed to reduce its size to less than 2 MB due to hardware memory limitations. The effective methods identified for reducing model size included altering the architecture to have fewer layers and parameters, along with quantization. The pruning method zeros out the weights without reducing the model size, making it less effective for this research. The methods employed in this research included modifying the model architecture to get fewer layers and parameters, quantization-aware training, and knowledge distillation that attempted to minimize the model loss after compression. We trained several models from scratch, introduced in the experiment section.

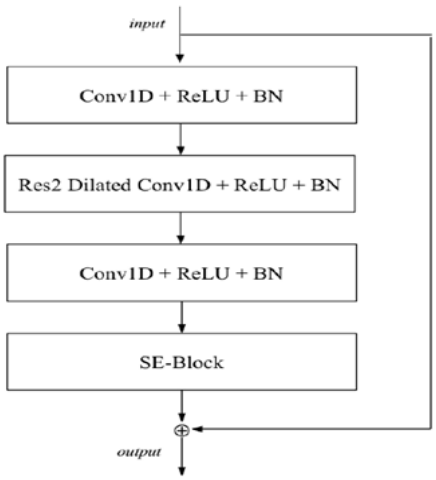
4.1. Model architecture

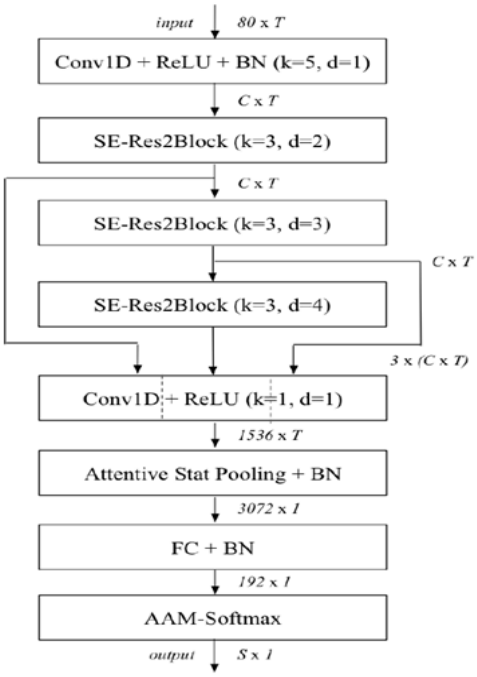
ECAPA and ResNet models included repeating structures such as bottleneck layers. We reduced the repetition of these bottleneck layers and decreased the number of filters in the convolution layers. This reduction was done by iteration until we got a result that was compatible with the hardware memory limitation. Additionally, we modified the shape of the Mel-spectrogram feature to minimize RAM usage on the microcontroller. Since not all functions or layers in the original architecture are supported by PyTorch quantization and the hardware itself, we had to adjust those functions or layers for proper functionality. For instance, PyTorch quantization does not support loop operations or certain arithmetic operations,

necessitating changes in the architecture as well as the SoftMax function that is replaced with sigmoid. This sometimes resulted in the need for dequantization before performing operations like addition or multiplication. Furthermore, since quantization does not support the BatchNorm layer, we had to fuse it with Convolution and ReLU into a single layer. There were also constraints regarding hardware limitations, such as the absence of concatenation, expansion, and clamp function, which the hardware is not supported by. The ECAPA-TDNN paper states [8], "*We concatenate the local input with the global non-weighted mean and standard deviation of the input across the time domain. This context vector should allow the attention mechanism to adapt itself to global properties of the utterance, such as noise or recording conditions*." In our tiny version, we modified this so that the attention mechanism only considered the local input, excluding the mean and the standard deviation, because of the absence of concatenation. This led to a difference in the attention mechanism layer. Additionally, we had to alter how the standard deviation was calculated due to the lack of a clamp function. The best-performing tiny model among the three tiny versions was the distillation tiny version of ECAPA, which was deployed on the microcontroller. The model's architecture is shown in Figure 1, as well as the original ECAPA-TDNN architecture [8].

4.2. feature extraction

Our preprocessing steps trained the models using one-second and two-second audio segments. Each segment was randomly sliced from audio files in the VoxCeleb1 dataset and passed through a pre-emphasis filter, which acts as a high-pass filter to emphasize high-frequency components while attenuating lower frequencies. Next, we created Mel-spectrograms using Librosa with the following parameters: FFT size of 1024, window length of 1024, hop length of 512, and a frequency range of 20 Hz to 8000 Hz. We set the Mel number to 80, the center to False, the power to 2, and the window type to 'Hamming.' After obtaining the Mel-spectrogram, we applied the logarithm function to the results. The subsequent step involved normalization; we employed a sophisticated method where each row (channel) of the Mel-spectrogram was normalized independently. Additionally, we applied masking augmentation in both the frequency and time domains. The normalized spectrogram was then fed into the neural network to get each speaker's voice signature, called the embedding vector. Each embedding contains 152 values. We applied augmentation techniques to enhance further the training data, including reverberation from the RIR dataset, babble noise, music, and television noise sourced from the MUSAN dataset. Due to the extensive training time required, we fine-tuned the model with these augmentations over a limited number of epochs.





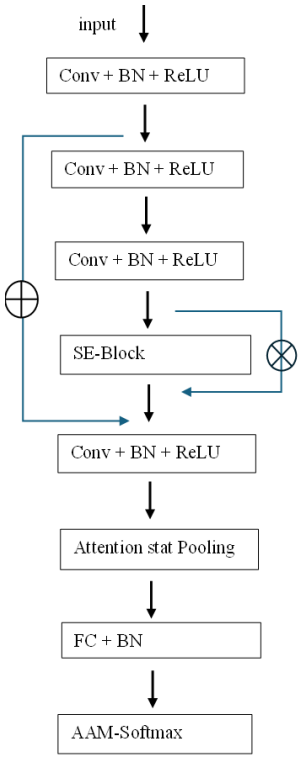


Figure 1: Left up - SE-Res2Block of the ECAPA-TDNN architecture. Right up - Network topology of the ECAPA-TDNN. Left down: Tiny version ECAPA deployed on the microcontroller

4.3. Model training

As mentioned, the network aimed to learn the unique embeddings of each speaker. During evaluation, we verified two embeddings and calculated their cosine similarity within the range of -1 to 1. Model performance was assessed using Equal Error Rate (EER) and Minimum Detection Cost Function (MinDCF), standard metrics in speaker verification [8]. The EER is the point at which the rate of false positives (false acceptance rate, FAR) equals the rate of false negatives (false rejection rate, FRR). This value is commonly used as a performance indicator in biometric systems, such as fingerprint and facial recognition, and in classification tasks where there is a trade-off between different types of errors. A lower EER indicates a better-performing system, showing a good balance between false positives and false negatives at a low error rate. The MinDCF is a more flexible measure that accounts for the different costs associated with errors. This is particularly useful when false positives and negatives have unequal consequences.

The MinDCF represents the minimum value of the Detection Cost Function (DCF), a weighted sum of the False Acceptance Rate and False Rejection Rate. The weights in this function reflect the relative costs of the two types of errors and the prior probabilities of the target classes. The training comprised 80-100 epochs for two-second audio segments, followed by another 80-100 epochs for one-second segments. For the distillation model, after completing the initial 100 epochs (for a two-second audio segment) and 200 epochs (for a one-second audio segment), we trained an additional 100 epochs using knowledge distillation, which involved calculating the Mean Squared Error (MSE) between the embeddings of the student and teacher models in addition to the cross-entropy loss. We conducted the training on an Nvidia GPU, and each model took approximately 80 hours to complete. The learning rate was set to 0.005, with a decay factor of 0.97, using the Adam optimizer with a weight decay of 2e-5 and AAM-Softmax as the loss function, with a margin and scale set to 0.2 and 30, respectively. Following training, we saved the model in ONNX format.

4.4. Hardware deployment

In the second part of the research, we focused on implementing the model on the low-power hardware. We utilized the evaluation board 32L4R9IDISCOVERY, an STMicroelectronics Arm® Cortex®-M4 core-based STM32L4R9AI microcontroller development platform. This board has no operating system and contains 2 MB of Flash memory and 640 KB of RAM in a UFBGA169 package, along with ST-MEMS digital microphones and two user LEDs, among other components that are not relevant to this research. Using X-Cube-AI [23], a software tool from STMicroelectronics, we converted the ONNX model into C files. We then programmed the application using the STM32Cube IDE in a C language. As mentioned, one of the challenges we faced was finding a model architecture compatible with the hardware. Another challenge was recreating similar preprocessing steps to those in the Python code, including generating the Mel-spectrogram, applying the log function, pre-emphasis, and normalization as done during training. We randomly selected two speakers from the test dataset to utilize the model in a real-time application, calculated their average embeddings, and saved these vectors in the microcontroller's memory. The MEMS microphone on the board samples audio at a rate of 16 kHz. We used only one microphone. The microphone records 16,000 samples in 1024 milliseconds (not 1000 milliseconds because of the chip limitation). It records two seconds of audio, and using Direct Memory Access (DMA), the microcontroller processes half of this segment at a time, i.e., one second of audio. We played through loudspeaker records of the enrolled speakers among other speakers. The audio is then converted into a Mel-spectrogram and passed through the network to generate an embedding of 152 values while the audio sampling continues in parallel. Using cosine similarity, the generated vector is compared with the two saved vectors on the hardware. If the similarity exceeds a predefined threshold, a specific LED is illuminated for one second to indicate the corresponding speaker. This setup allows us to receive a real-time indicator of whether a speaker is one of the two enrolled individuals.

Table 1: Performances of speaker verification models on VoxCeleb1 test dataset

				Training on 2-second audio		Training on 1-second audio	
Model name		Number of parameters	Model ONNX size (Mbytes)	EER (%)	MinDCF (%)	EER (%)	MinDCF (%)
Gold standard ECAPA		15,315,400	61.37	4.26	0.2962	4.59	0.3021
Gold standard Resnet		23,837,144	95.1	8.96	0.5885	12.65	0.7511
Distillation tiny version ECAPA	Before distillation	449,480	1.81	8.4	0.4652	10.65	0.5609
	After distillation			7.28	0.4228	9.22	0.5284
QAT tiny version ECAPA		1,019,736	1.9	16.51	0.7124	26.88	0.9990
Tiny version Resnet		444,784	1.77	11.63	0.7031	14.43	0.8023

5. Experiments

We examined and trained a few models from scratch. Each model required a few iterations to set the parameters and layers according to the hardware's restrictions and quantization.

- A version of ECAPA-TDNN with 15 million parameters, 1,024 channels, and a model size of 61.3MB, referred to as the ECAPA-TDNN Gold Standard.

- A version of ResNet with 24 million parameters and a model size of 95.1MB, referred to as the ResNet Gold Standard.

- A tiny version of ECAPA-TDNN with 450,000 parameters, 64 channels, and a model size of 1.8MB. This model was trained using the knowledge distillation method, where the Gold Standard serves as the teacher, and the tiny version acts as the student.

- A tiny version of ECAPA-TDNN with 1 million parameters, 180 channels, and a model size of 1.9MB, trained using the quantization-aware training (QAT) method.

- A tiny version of ResNet with 450,000 parameters and a model size of 1.8MB.

6. Results

We observed that the model's performance improved with the two-second audio segment compared to the one-second segment. However, we still needed to train with a one-second audio segment due to hardware memory limitations that could not handle more than one second of audio. The gold standard ECAPA model achieved the best EER and MinDCF performances of 4.26% and 0.29 with the two-second audio segment and 4.59% and 0.3 with the one-second segment. Training with the two-second audio segment resulted in a 7.1% improvement in that case. The tiny version of the ECAPA model has a size of 1.81 Mbytes, making it suitable for deployment on the microcontroller. This model achieved EER and MinDCF values of 8.4% and 0.46% with the two-second audio and 9.22% and 0.52% with the one-second audio. The distillation process improved performance by 13.3% EER and 9.11% MinDCF compared to the undistilled version for the two-second audio segment and 13.5% and 6% for the one-second segment. The quantized training model exhibited the lowest performance, achieving 16.51% EER and 0.71% MinDCF with the two-second audio and 26.8% and 0.9% with the one-second audio, even though it has nearly double the number of parameters

compared to the other two tiny models. The ResNet model achieved 8.96% EER and 0.58% MinDCF with a model size of 95.1 Mbytes with a two-second audio segment and 12.65% EER and 0.75% MinDCF with a one-second segment. The tiny version performed lower, with 11.63% EER and 0.7% MinDCF for the two-second audio and 14.43% EER and 0.8% MinDCF for the one-second audio.

To evaluate the embedding vector we get from the hardware, we saved on the board a few audio samples used to test the model performance on the computer, and after debugging the board, we obtained a similar embedding output with an MSE of 0.0003, indicating that the computational process on the microcontroller was successful. Calculating the Mel-spectrogram, which includes pre-emphasis, logarithm, and normalization, takes 232 milliseconds. Calculating the embedding from the Mel-Spectrogram takes 704 milliseconds. The total process, including the cosine similarity, takes 937 milliseconds. The current consumption of the board is 30 mA when all the processes are working and 28.3 mA when the network and the Mel-spectrogram calculation are disabled. We set a threshold of 0.29 for the female speaker and 0.23 for the male speaker. This threshold influences the sensitivity of the system and determines the false positives and false negatives we get from the system in real-time, which could be changed according to the application requirements. A demonstration of the system is shown in the demo.

7. Discussion

This research focused on successfully implementing a speaker verification model on low-power hardware for real-time applications. The “Distillation Tiny Version ECAPA” model, which outperformed the other two tiny versions, was selected for deployment on the evaluation board. The model size decreased by 97%, from 61.38 Mbytes to 1.8 Mbytes, and performance decreased by 50%. The knowledge distillation method used in the project was an MSE between the embeddings that led to an improvement of 13.5% EER; other distillation methods, like soft target and cosine similarity, could be tested in future research.
The QAT model demonstrated inferior performance compared to the other two small models, even though it has nearly double the number of parameters compared to those models, which might highlight the significance of using floating-point representation when working with tiny models. The fact that two seconds of training achieved higher results than the one-second audio segment indicates that the time resolution of the Mel-spectrogram feature was low or the temporal information was insufficient. It could be tested in future work.

To compare the QAT and the distilled versions, we considered the limitations of PyTorch’s quantization and the hardware constraints in both model architectures. We hypothesize that removing the quantization restrictions in the distilled version could lead to improved model performance. Further research is necessary to test it.

The STM32 X-Cube-AI software showed a relatively accurate conversion of the ONNX model between hardware inference and Python inference (on the computer). However, there are still layers and functions that the hardware does not support. We believe that advancements in the chip industry will address these limitations in the future. The decision mechanism of lighting up the LED is based on one-second audio; it could be

improved if we make the decision to average a few seconds instead of one.

8. References

[1] P. E. Novac, G. B. Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, "Quantization and deployment of deep neural networks on microcontrollers," *Sensors*, vol. 21, no. 9, pp. 1–34, 2021, doi: 10.3390/s21092984.

[2] J. Li, W. Liu, Z. Zhang, J. Wang, and T. Lee, "Model Compression for DNN-based Speaker Verification Using Weight Quantization," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2023-Augus, pp. 1988–1992, 2023, doi: 10.21437/Interspeech.2023-1524.

[3] S. S. Tirumala, S. R. Shahamiri, A. S. Garhwal, and R. Wang, "Speaker identification features extraction methods: A systematic review," *Expert Syst. Appl.*, vol. 90, pp. 250–271, 2017, doi: 10.1016/j.eswa.2017.08.015.

[4] S. K. D. S. D. G.-R. Daniel Povey and Center, "Deep Neural Network Embeddings for Text-Independent Speaker Verification," pp. 999–1003, 2017.

[5] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2018-Septe, pp. 2252–2256, 2018, doi: 10.21437/Interspeech.2018-993.

[6] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 7132–7141, 2018, doi: 10.1109/CVPR.2018.00745.

[7] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2018-April, pp. 5329–5333, 2018, doi: 10.1109/ICASSP.2018.8461375.

[8] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2020-Octob, pp. 3830–3834, 2020, doi: 10.21437/Interspeech.2020-2650.

[9] J. Zhou, T. Jiang, Z. Li, L. Li, and Q. Hong, "Deep speaker embedding extraction with channel-wise feature responses and additive supervision softmax loss function," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2019-Septe, pp. 2883–2887, 2019, doi: 10.21437/Interspeech.2019-1704.

[10] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 5962–5979, 2022, doi: 10.1109/TPAMI.2021.3087709.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[12] S. Vadera and S. Ameen, "Methods for Pruning Deep Neural Networks," *IEEE Access*, vol. 10, pp. 63280–63300, 2022, doi: 10.1109/ACCESS.2022.3182659.

[13] A. Nagraniy, J. S. Chungy, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2017-Augus, pp. 2616–2620, 2017, doi: 10.21437/Interspeech.2017-950.

[14] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and

connections for efficient neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 1135–1143, 2015.

[15] Z. Wei, H. Li, and X. Zhang, "Model Compression by Iterative Pruning with Knowledge Distillation and Its Application to Speech Enhancement," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2022-Septe, no. September, pp. 941–945, 2022, doi: 10.21437/Interspeech.2022-619.

[16] Z. Huang and N. Wang, "Like What You Like: Knowledge Distill via Neuron Selectivity Transfer," 2017.

[17] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2704–2713, 2018, doi: 10.1109/CVPR.2018.00286.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, pp. 1–14, 2016.

[19] A. Liss, "Implementation of a neural network on a microcontroller for recognition of warning signals," 2020.

[20] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxceleB2: Deep speaker recognition," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2018-Septe, no. ii, pp. 1086–1090, 2018, doi: 10.21437/Interspeech.2018-1929.

[21] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 5220–5224, 2017, doi: 10.1109/ICASSP.2017.7953152.

[22] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," pp. 2–5, 2015.

[23] STMicroelectronics, "X-CUBE-AI Data brief Artificial intelligence (AI) software expansion for STM32Cube X-CUBE-AI," no. September, 2019.