SPIDER: Lightweight Speaker Identification on Resource-Constrained Embedded Devices

Markus Gallacher
Institute of Technical Informatics,
Graz University of Technology
Graz, Austria
markus.gallacher@tugraz.at

Carlo Alberto Boano
Institute of Technical Informatics,
Graz University of Technology
Graz, Austria
cboano@tugraz.at

Arun Sankar Muttathu Sivasankara Pillai
South East Technological University
Carlow, Ireland
arun.sankar@setu.ie

Utz Roedig
University College Cork
Cork, Ireland
u.roedig@cs.ucc.ie

Willian Lunardi
Technology Innovation Institute
Abu Dhabi, United Arab Emirates
willian.lunardi@tii.ae

Michael Baddeley
Technology Innovation Institute
Abu Dhabi, United Arab Emirates
michael.baddeley@tii.ae

Abstract

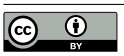
Voice-based Speaker Identification (SI) can be framed as the problem of Closed-Set Speaker Identification (CSSI), recognizing a speaker from a known set, or OSSI, additionally recognizing unknown speakers. Precise and accurate Open-Set Speaker Identification (SI) can enable a variety of applications, ranging from human presence detection to authentication. Existing SI solutions are typically driven by deep learning approaches, which involve computationally demanding models often running in cloud back-ends. Enabling local SI models running directly on resource-constrained embedded devices can enable new use cases while preserving speaker privacy. In this work, we fill this gap and present `SPIDER`, a lightweight, on-device CSSI and OSSI solution capable of running on the off-the-shelf Nordic nRF52840 and nRF5340 system-on-chip microcontrollers, which feature as little as 256 and 512 kB of RAM, respectively, and 1 MB of flash memory. `SPIDER` is *16x-67x smaller than currently-available SI models*, and yet, the 16x smaller version achieves a comparable accuracy of 94.33 % for CSSI and 91.8% for OSSI. Our evaluation across multiple datasets confirms the viability of performing accurate SI directly on resource-constrained embedded devices using only low-cost microphones. To foster further research and development, we open source our implementation of SPIDER, empowering the community to explore new SI use cases where cloud connectivity or backhaul infrastructure is impractical or undesirable.

CCS Concepts

• **Computer systems organization** → **Embedded software**; • **Computing methodologies** → *Speech recognition.*

Keywords

Open-Set Speaker Identification, Lightweight Models, Embedded AI, Information Processing, Machine Learning, Datasets.



SenSys '26, Saint Malo, France

ACM ISBN 979-8-4007-2309-4/26/05
https://doi.org/10.1145/3774906.3800492

ACM Reference Format:
Markus Gallacher, Carlo Alberto Boano, Arun Sankar Muttathu Sivasankara Pillai, Utz Roedig, Willian Lunardi, and Michael Baddeley. 2026. SPIDER: Lightweight Speaker Identification on Resource-Constrained Embedded Devices. In *ACM/IEEE International Conference on Embedded Artificial Intelligence and Sensing Systems (SenSys '26), May 11–14, 2026, Saint Malo, France.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3774906.3800492

1 Introduction

Speaker Identification (SI) is the process of identifying who is speaking within a given audio signal. SI may consider Closed-Set Speaker Identification (CSSI), where the speaker is one out of a set of known speakers, or Open-Set Speaker Identification (OSSI), where the speaker may be *either* known *or* unknown to the system.

Although OSSI considers speakers that are unknown to the system and thus also covers use-cases addressed by CSSI, the use of either an OSSI or CSSI approach depends primarily on the desired performance of the SI solution. For example, one could employ CSSI to identify *who* is speaking, *when*, and for *how long* in a meeting. For convenience, speakers should be identified quickly, while occasional misidentification is acceptable. In contrast, if one considers an *authentication and access control* system based on OSSI, misidentification is concerning, even if it occurs rarely. Convenience may be less of an issue in this setting, where the cost of processing additional speech samples shall be preferred to misidentification.

SI and particularly OSSI could enable many novel applications if they could be executed directly on resource-constrained embedded devices, rather than relying on edge or cloud processing. Such use cases include tracking individuals in a meeting room, inferring interaction profiles in smart home systems or smart earbuds like the OpenEarable v2[1], and authentication to control access to devices or areas, e.g., allowing to start a communication channel on smart earbuds or unlocking a smart bike lock. Some of these applications need a solution running on tiny microcontrollers, e.g., the OpenEarable v2 houses an nRF5340 SoC or smart bike lock, while other applications can be run on a Raspberry Pi or similar class of device. However, they would still require a connection to the power grid or a very large battery to run for an extended period of time. This entails a more limited freedom of use, more installation requirements, and additional regulations for companies to adhere

[1] https://open-earable.teco.edu

to. For this reason we require a very small CSSI and OSSI solution that can run on resource-constrained embedded devices that have 500-15625 times less RAM than traditional Raspberry Pis.

Limitations of existing works. Unfortunately, current SI approaches typically rely on computationally-demanding Deep Learning (DL) models [16]. Although these models can be highly accurate to within the 99th percentile [4], their memory and processing demands make them unsuitable for small devices. Many existing works target larger devices like Raspberry Pis, and focus only on the related speaker verification task [1, 30] or only on CSSI [4, 13] that have access to sophisticated USB microphones. TinyCNN [32] does run CSSI, but not OSSI, on resource-constrained embedded devices, and provides performance metrics only for 10 speakers. Consequently, there are no detailed studies available that show the performance of CSSI and OSSI on resource-constrained embedded devices. Hence, we aim to answer the research question: *is CSSI, and especially OSSI, feasible using cheap but ubiquitous PDM microphones, and the limited processing capabilities and limited memory of resource-constrained embedded devices*?

Contributions. We present SPIDER (SPeaker IDEntification on Resource-constrained devices), an open-source[2], lightweight, on-device CSSI and OSSI solution capable of running on any platform able to run TensorFLow Lite for microcontrollers, has access to a microphone, and offering as little as 256 kB of RAM and 1 MB of flash memory, which we demonstrate on the Nordic nRF5340 System-on-Chip (SoC) featuring only 512 kB of RAM and 1 MB of flash memory and the Nordic nRF52840 SOC featuring 256 kB of RAM and 1 MB of flash memory. Our work bridges the gap between advanced DL models for SI and the limitations of embedded hardware. Our target platforms, the nRF52840 and nRF5340, have ~10 times less memory than is typically required by state-of-the-art models. We explore how model size can be reduced without significantly sacrificing accuracy, and assess the performance of these reduced models in both CSSI and OSSI tasks to demonstrate their feasibility for real-world applications. We also provide a dataset through which we illustrate and showcase these real-world applications in a practical context.

To illustrate the capabilities of SPIDER, we highlight one single data point of our study, tested on a real-world dataset that we collect to show an example of on-device authentication. Using an nRF5340, we can identify the speaker with an *accuracy of 91.8 %*, while achieving a *False Rejection Rate (FRR)* of 4.4 % and a *False Acceptance Rate (FAR) of 4.9 %*. This data point represents an application where FRR and FAR are kept as balanced as possible.

Our main contributions can be summarized as follows:

- Through detailed ablation studies using a reversed compound scaling approach, we reduce the size of state-of-the-art models (such as the MobileNetV2 [4], ResCNN [33], and xResNet18 [14]) in a structured manner. This extends the works of [27, 31], who focus on image classification tasks, to the challenging OSSI task with 32 and 630 speakers in resource-constrained embedded devices.
- We study in detail three essential aspects of SPIDER:
  (i) Pre-processing: while modern SI solutions usually process the recorded waveforms directly, this entails more computations to handle the larger input space. Hence, we also explore Mel-frequency Cepstral Coefficients (MFCC) and Log Mel Spectrogram (LMS), which both reduce the input space. We show that LMS is preferable, achieving the best trade-off with over 91.8 % OSSI and CSSI accuracy and needing 1.99 s to process a 1 s audio sample with a model that is shrunk 16 times from its original size.
  (ii) Consensus: to increase OSSI performance, we propose a consensus threshold among consecutive speech segments. This allows us to reduce the false acceptance rate by rejecting only partially matching speakers.
  (iii) Sample length: we consider the impact of the audio sample's length on training and inference. To enroll a new speaker, a minimum of 13.4 s should be added to the training set and at least 2-5 s should be recorded to test a speaker.
- We showcase how the output and consensus thresholds can be used to tune SPIDER either towards more lenient or more strict security use-cases. For strict OSSI, the FAR is limited to 0.81 % and FRR equals 12.1 %. For lenient OSSI, the FAR is allowed to be 18.2 %, which reduces the FRR to 2.3 %.
- We release the SPIDER implementation as open-source[2] software, enabling the community to investigate novel SI applications. In addition, we collected and make publicly available our real-world dataset[3] with 32 test subjects, and evaluate it with SPIDER using a Nordic Thingy:53 (housing an nRF5340 SoC) using a Pulse Density Modulation (PDM) microphone. We evaluate performance and other key factors such as storage, energy, and runtime on our real-world dataset, as well as on two publicly available datasets with 630 test subjects.

Paper outline. The remainder of this paper is organized as follows. Section 2 provides the necessary background and outlines related work. Section 3 discusses our methodology. Section 4 describes the shrinking of the selected models and the feasibility of their deployment on a resource-constrained embedded device. Section 5 considers real-world constraints and evaluates the role of the speech length for training and testing, as well as SPIDER's runtime and energy consumption. Section 6 discusses how our approach enables new use-cases. Section 7 concludes the paper, along with a discussion of future work.

2 Background and Related Work

2.1 Speaker Recognition

Speaker recognition is a broad field that involves determining or verifying a speaker's identity based on their voice. This could be Speaker Verification (SV) or SI in form of CSSI or OSSI [2]. In this work, we focus on SI and OSSI.

Speaker verification (SV) is the process of confirming a speaker's claimed identity based on their voice. A voice sample is matched against a specific voice profile of the claimed identity.

Speaker Identification (SI) is the general process of identifying who is speaking in a given audio signal. If the number of speakers to be recognized is 1, then SI becomes SV.

Closed-Set Speaker Identification (CSSI) or in-set identification is the process of determining who is speaking from a set of known enrolled speakers by comparing a given voice sample against a set

[2]Link to SPIDER's GitHub repository can be found under https://iti.tugraz.at/spider.

[3]Link to open-source dataset can be found under https://iti.tugraz.at/spider.

of enrolled speakers and selecting the best match. It is assumed that the incoming voice sample always belongs to one speaker in the set of enrolled speakers.

Open-Set Speaker Identification (OSSI) is based on CSSI, but allows also for the possibility that the speaker might not be from the pool of enrolled speakers. In such cases, it can reject the speaker as unknown. In OSSI, an incoming voice sample is first analyzed to determine whether it is accepted as an enrolled speaker (in-set) or is rejected to be out-of-set (unknown), and is only identified if it has been accepted.

2.2 Speaker Recognition Models

SI algorithms have evolved significantly over the years, leveraging different machine learning techniques. Traditional methods often relied on Gaussian mixture models and hidden Markov models to model speech features [22, 24]. However, recent advancements have introduced deep neural networks, which gained attention due to improved performance [16]. Techniques like i-vectors and x-vectors have become popular for extracting speaker-specific features, while convolutional neural networks and residual neural networks enhance performance by handling temporal and spectral variations in speech [17, 25]. Hybrid approaches that combine these networks are also being explored to enhance performance by combining the benefits of both systems [11, 29]. ECAPA-TDNN [7] incorporates Res2Net modules and Squeeze-and-Excitation (SE) blocks to obtain robust speaker embeddings. The Enhanced Res2Net (ERes2Net) [6] architecture, originally used for SV introduces Local Feature Fusion (LFF) within residual blocks and Global Feature Fusion (GFF) across scales, generating discriminative embeddings suitable for other speech-related tasks. Also transformer models such as WavLM [5], Whisper [9], and Wav2vec [10] provided improved performance for speaker recognition tasks due to their contextual representation capabilities.

In addition to improving the accuracy of models for SV and SI, the research community has also looked at reducing the complexity of these models with the aim to execute these on smaller platforms such as Internet of Things (IoT) or mobile devices.

Georgiev et al. [13] developed a multi-task learning framework for optimizing deep audio models by selectively sharing hidden layers across related tasks with the objective of reducing resource usage on embedded devices. Although CSSI is across the tasks considered in this work, the targeted platform is a Qualcomm Snapdragon, which is significantly larger than the nRF5340 we consider. Sandler et al. [23] introduce MobileNetV2, a small mobile convolutional neural network that performs well in many different tasks using inverted residual structures. Nunes et al. [4] adapted MobileNetV2 to create the AM-MobileNet1D model for speaker recognition. They report a model size of 11.6 MB of flash memory, which is more than one order of magnitude larger than the memory available on of the nRF5340 platform we target in `SPIDER`. Moreover, this work focuses exclusively on SV and CSSI, leaving OSSI out of scope. Velez et al. [30] design a lightweight model for SV, but the model size is large (several MB) and is therefore not suitable for the nRF5340 platform, which only embeds 1 MB of flash memory. Balian et al. [1] provide a model for SV that targets a Raspberry Pi 3B with 1 GB of RAM. TinyCNN [32] introduces an optimized Convolutional Neural Network (CNN) architecture designed to operate efficiently on resource-constrained devices such as the ESP32 microcontroller with 320 kB of RAM and 4 MB of flash memory. However, this work only evaluates 10 speakers to be identified in a CSSI setting and OSSI is not considered. We consider OSSI and up to 630 speakers.

In summary, existing work focusing on execution of SI has mainly focused on SV and CSSI, and did not consider OSSI, which is crucial to identify, and accept or reject unknown speakers. Existing work also predominantly relies on existing available datasets such as TIMIT [12], which contain audio recorded using high-end microphones, whereas we also show the performance of SPIDER when using an off-the-shelf embedded platform embedding an inexpensive PDM microphone. To this end, we record our own real-world dataset, and provide it open-source[3].

2.3 Pre-Processing and Feature Extraction

Speech is either provided as raw waveform to an SI model or it is pre-processed to extract features which are then used as model input. Pre-processing aims to reduce the model input size and, in turn, the model itself. Commonly employed pre-processing techniques are MFCC and LMS, which we use in this work. For simplicity, we also refer to the raw waveform as a pre-processing technique.

Waveform. We directly use the raw, but normalized, waveform as input to the model without any further pre-processing.

Mel-Frequency Cepstral Coefficients (MFCC). Introduced by Davis and Mermelstein in the 1980s [8], it is among the most widely utilized low-level features extracted for speech processing applications. The Mel scale provides a more accurate representation of the human auditory system's response and the steps are as follows: (i) Pre-emphasis followed by framing and windowing (Hamming or Hanning); (ii) The power spectrum of frames are obtained using N-point Fast Fourier Transform (FFT); (iii) The power spectrum is passed through Mel filter banks [21]; (iv) Discrete Cosine Transform (DCT) on the logarithm of the filter bank energies and MFCCs correspond to the amplitudes of the resulting spectrum.

Log Mel Spectrogram (LMS). In MFCC, the DCT reduces feature correlation, allowing representation with fewer coefficients (typically 12-13) instead of the 20-40 used in Mel filter banks. LMS are the log energies obtained from Mel filters, which are then transformed using DCT to derive MFCCs. LMS are often preferred in deep learning models, as they preserve spatial relationships within the spectral domain [19].

3 Methodology

We target device classes that have roughly one order of magnitude less memory available than existing models, such as the nRF5340 SoC. We also evaluate how far we can shrink the model and evaluate the even smaller nRF52840 SoC (Section 3.1). In order to produce an SI model with small memory footprint, we select promising candidate models that we then shrink using a reverse compound scaling approach (Section 3.2). This approach allows us to pick the most suitable model for a specific device, i.e., for a given amount of RAM and flash memory on the device (Section 3.3). The models with the highest CSSI accuracy for each device can then be tuned using thresholds to differentiate in-set and out-of-set speakers. These

thresholds need to be parametrized depending on the target use case (Section 3.4). We use three datasets to evaluate models: two publicly available and our own real-world dataset recorded on a Nordic Thingy:53 running an nRF5340 SoC (Section 3.5).

3.1 Target Devices

We do not aim for large embedded devices, such as Raspberry Pis, but for resource-constrained devices powered by small batteries. However, we do need to be able to run TensorFlow Lite for microcontrollers, have access to a microphone, and have enough RAM to store and process the recorded audio and to run inference, and enough flash memory to store the model, limiting how small of a device can be chosen. Our search revealed the nRFThingy:53 IoT prototyping platform that comes with a PDM microphone as well as a ready-to-use software pipeline using EdgeImpulse[4] to collect our own dataset, housing an nRF5340 SoC that is growing in popularity. Therefore, we target the nRF5340 SoC as well as its smaller counterpart the nRF52840 SoC to show the generality of our solution for two distinct devices: (i) the nRF52840 has 256 kB of RAM and 1 MB of flash memory and has a 64 MHz Cortex-M4; (ii) the nRF5340 SoC has 512 kB of RAM and 1 MB of flash memory and has a 128 MHz Arm Cortex-M33[5]. Assuming that we use full integer quantization, each model parameter would be stored in 1 Byte. This allows us to estimate the model size in flash memory. However, the final memory requirements for RAM and flash of a model depend on the target architecture, compiler settings, and TFLM optimizations. We opt to first filter out clearly unsuitable models by estimating the flash memory size of the models, and to then deploy hand-selected models on our target platforms to report their actual memory requirements.

3.2 Candidate Models

We employ established state-of-the-art models as baselines for evaluating our proposed solutions. We select potential SI models from a large pool (see Section 2.2) based on factors such as high CSSI accuracy, limited memory, low latency, and minimal energy consumption. Large models such as ECAPA-TDNN (∼25 MB) [7] and ERes2Net (which is even larger) [6] are not considered in this work. Therefore, we choose two of the smallest models that have been used for SI tasks; the AM-MobileNet1D (∼12 MB) [4] (based on MobileNetV2) and xResNet (∼12 MB) [14] (an updated version of the ResNet). We also select the ResCNN (∼1.2 kB) [33] that has not been used for SI, yet it is a 10x smaller than the other two models and has achieved promising results across multiple fields.

`MobileNetV2` [4] is chosen as its architecture is explicitly designed for mobile and embedded use. It employs depthwise separable convolutions, which drastically reduce both the number of parameters and computational cost compared to standard convolutions, making it highly efficient for real-time inference in low-resource environments. It is a compact model designed to run on mobile phones and is one of the smallest (∼12 MB) that has been used for SI so far. Nunes et al. [4] adapted it to use 1D data and additionally use an additive margin (AM) softmax loss function. In our case, we consistently obtain 1-5% better accuracy scores with the Cross -Entropy Loss (CEL) compared to the AM softmax, and hence, we opt to use a standard CEL.

`xResNet18` [14] is a versatile convolutional network that uses residual connections to improve optimization in deeper layers. It is a modified version of ResNet models, which in general help to maintain accuracy even if the network is compressed or scaled down [18]. Lightweight versions of xResNet such as `xResNet18` have been successfully applied in various vision and audio tasks, showing a good balance between accuracy and efficiency [3]. The `xResNet18` also has a very similar size to the `MobileNetV2` (∼12 MB).

`ResCNN` [33] is selected due to its proven effectiveness in speech and audio processing tasks [26]. Its modular design, built on residual convolutional blocks, enables flexible scaling and selective pruning, which is ideal for optimization and adaptation in embedded systems. It is a tiny model (∼1.2 kB) combining residual and convolutional neural networks along with three different activation functions, and is considered for time series classification. To the best of our knowledge, it has not been used for speaker recognition-related tasks so far, but it achieves the best accuracy from 19 out 44 public datasets by the University of California in different fields (including image, sensor, and motion) [33]. The model also has ∼95% fewer parameters than the `MobileNetV2`, making it a good candidate.

3.3 Model Reduction and Selection Process

In their original state[6], `MobileNetV2` and `xResNet18` need around ∼12 MB of flash memory to store the model, making them unusable on microcontrollers such as the nRF52840 and nRF5340 with 1 MB of flash memory. The ResCNN is around 10 times smaller with around 1.2 MB, which is better, but still too large.

Model Reduction. We follow the work of Tan and Le [27] to reduce the size of any convolutional neural network model, while retaining as much accuracy as possible. Within their work, they systematically show that best performance is achieved when scaling width, depth, and input resolution together in a balanced way. They use the constants α, β, and γ and a variable scaling factor ϕ to represent the depth ($d = \alpha^{\phi}$), width ($w = \beta^{\phi}$) and resolution ($r = \gamma^{\phi}$), such that $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ and $\alpha \geq 1,\ \beta \geq 1,\ \gamma \geq 1$. The weighted constants describe how much influence the scaling should have, and are determined with a small grid search. The constraints are chosen to approximately increase the Floating Point Operations (FLOPS) by 2^{ϕ}. This approach, however, focuses on scaling up a model and increasing the performance. In our case, we want to *scale down a model by shrinking its width and depth*, similarly to the works of Wang et al. [31] and Tan et al. [28] that scale down models for image classification. We further investigate this methodology to new data types, in our case, 1D raw waveforms and their related 2D spectrograms. Apart from focusing on image and not audio data, the models of [27, 28] have between 1.3-208 M parameters, while our target devices only have 1 MB of flash memory available that can theoretically only store 1M 8 bit parameters and hence, require further down-scaling. We opt to not scale the input resolution, because we do not want to lose information in our already limited audio resolution (i.e., 16 kHz sample rate). Hence, we empirically

[4]https://edgeimpulse.com

[5]The nRF5340 SoC also has a second networking core with half of the memory and clock frequency, but it is not used in our evaluation.

[6]With floating point operations instead of full integer quantization.

choose the weighted constants[7] $\alpha = 0.6$ and $\beta = 0.6$, and scale these with five different ϕ values [0, 0.75, 1.5, 2.25, 3]. These values for phi were chosen to get a wide span of information without exploding the search space, i.e., from the original model ($phi = 0$) that is ~10x too large to a breaking point, where the model's accuracy becomes questionable. Once the size-reduction parameter region, where the desired models are located, a more fine-grained search, e.g., based on grid search or other optimization techniques, can be employed. In this work, we forgo this refinement step, since our focus is on validating the feasibility of our approach.

We noticed that reducing the depth of our candidate models with high ϕ values severely impacts the accuracy, as most of the model's layers are removed for high ϕ values. For this reason, we ensure a minimal depth for all models. Specifically, for `xResNet18` and `MobileNetV2` we only reduce the number of stacked identical blocks, but keep a minimum of 1. `ResCNN` is only reduced in width, as the original `ResCNN` is already as small as our smallest xResNet version and is affected the most by a depth reduction. As the `ResCNN` is already very small, we also explore the option of scaling up the depth and width and empirically choose $\alpha = 1.3$ and $\beta = 1.3$ and use the same ϕ values.

Selection Process. The best model for a device heavily depends on the target application and the tasks it needs to perform. For instance, a device dedicated to SI can use all its memory and runtime solely to the SI task, while for an integrated SI solution, an engineer needs to be aware of memory and processing requirements of the firmware and other tasks running. What this entails will be discussed further in Section 6. Therefore, we select the two models with the highest CSSI accuracy that still fit into the memory of two different device classes, where the tinier model can be used as a dedicated SI solution or an integrated SI solution. The CSSI accuracy is used in this step rather than OSSI accuracy to not disproportionately expand the analysis and keep complexity reasonable, as OSSI is based on the CSSI model with additional thresholds.

3.4 SPIDER SI Metrics

We define CSSI and OSSI and their performance metrics used by `SPIDER`.

3.4.1 `SPIDER` *CSSI Approach.* Let $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ denote a dataset of N raw audio signals, where each x_i represents waveform data. Each audio signal x_i undergoes a pre-processing stage in which its amplitude is normalized, and starting and trailing silent segments are removed. Additionally, we augment the samples to improve the robustness of the models by randomly scaling the samples by a factor between 0.8 and 1.2. We partition $\mathcal{D}$ into two subsets: a training subset $\mathcal{D}_{\text{train}}$ (70 %) and a testing subset $\mathcal{D}_{\text{test}}$ (30 %). For training purposes, each $x_i \in \mathcal{D}_{\text{train}}$ is segmented into fixed-length chunks s_{ij} with a 95% overlap, where j indexes the segments derived from sample i. Moreover, each segment s_{ij} is labeled with the identity of the speaker y_i from which it originates.

During the testing phase, each audio sample x_i from $\mathcal{D}_{\text{test}}$ is similarly segmented into chunks s_{ij}. Each chunk s_{ij} is then fed into the trained model, which outputs a probability distribution $\hat{p}_{ij}$ over the potential speaker classes. The predicted class label $\hat{y}_{ij}$ for each segment s_{ij} is determined by $\hat{y}_{ij} = \arg\max_c \hat{p}_{ij}(c)$, where c indexes the possible speaker classes. The final prediction $\hat{y}_i$ for the entire audio file is determined using majority voting among the valid segments. *The CSSI accuracy then denotes the percentage of final predictions that match the speakers from a dataset.*

3.4.2 `SPIDER` *OSSI Approach.* In out-of-set detection, we are interested in the ability to detect unknown speakers. We base our approach on the definition of the Maximum Softmax Probability (MSP) as introduced in [15], which is a widely-used and well-known baseline for out-of-set detection. The scoring function is defined as $s(x_i)$ to determine if a given audio file x_i is in-set or out-of-set based on the segments that contributed to the majority vote. Let $S_{\text{maj}} = \{s_{ij} : \hat{y}_{ij} = \hat{y}_i\}$ denote the subset of segments that contributed to the majority decision for x_i. We compute the sum of softmax probability over S_{maj} and scale it with the number of segments as follows $s(x_i) = \frac{1}{|S_{\text{ij}}|} \sum_{s_{ij} \in S_{\text{maj}}} \hat{p}_{ij}(\hat{y}_i)$.

The scoring function $s(x_i)$, where higher values express the confidence of an identified speaker being in-set, can be used alongside a threshold (τ) to accept or reject an audio file x_i. τ is found by analyzing the interplay of the false acceptance rate (FAR)[8] and false rejection rate (FRR)[9].

In SPIDER OSSI, we modify this approach in two ways: (i) we use τ to only output probabilities ($\hat{p}_{ij} > \tau$) to contribute to the final majority decision, (ii) we introduce consensus among x_{ij} to ensure that enough segments participate in the majority decision. Hence, a speaker can only be accepted and identified if more segments of x_{ij} than the consensus threshold have a probability higher than τ. This approach ensures that the model needs to be confident when predicting a speaker and avoids speakers being accepted and identified with, e.g., only 2 out of 10 segments, and enforces a speaker to provide enough voice samples to reach consensus. *The OSSI accuracy is thereby defined as the percentage of accepted and correctly identified speakers of the known dataset.*

The acceptable performance of OSSI needs to be defined by its use-case, as a more strict model will reject more unknown speakers, but is more likely to also reject known speakers. For example, access control to a building or a room will prioritize to reject unknown speakers and will prefer that the user potentially needs more tries to be accepted. Vice versa, automatically loading a user profiles for home automation cares more to correctly accept and identify a user on the first try, than rejecting unknown users and potentially having to repeat oneself to be accepted. We can adapt the τ threshold of the model and enforce consensus among voice samples to tune OSSI to suit a specific use-case. To showcase this, we evaluate our results with three use-cases:

Balanced: the aim is to find the thresholds where the FRR and FAR intersect. This leads to a trade-off between leniency and strictness.

More `strict`: the aim is to increase security by reducing the FAR further at the cost of convenience, as some known speakers need to speak again after being rejected. We empirically do not allow a higher FAR than 1 % when selecting the τ threshold and the required consensus threshold.

More `lenient`: the aim is to increase convenience by reducing the FRR at the cost of of higher FAR. Since OSSI enables an SI model to

[7] As the original authors [27] suggest.

[8] Percentage of out-of-set samples falsely accepted as in-set samples.

[9] Percentage of in-set samples falsely rejected as out-of-set samples.

Table 1: Audio length statistics per dataset. Both have a similar mean, but LibriSpeech has >4 times longer audio samples.

Dataset	Max	Min	Mean
TIMIT	7.13 s	0.65 s	2.7 s
LibriSpeech	29.3 s	0.24 s	12.25 s
Real-World	4.89 s	1.5 s	3.1 s

reject unknown speakers, we still do not want the FAR to increase too much. We empirically do not allow a FAR higher than 20 % when selecting the softmax output threshold and the required consensus threshold.

3.5 Evaluation Datasets

For our evaluations, we use two public datasets and one self-recorded real-world dataset (Table 1). These datasets are very different in nature and are carefully chosen to validate our solution in different settings. The CSSI accuracy is evaluated by training and testing on all three datasets separately. To test OSSI accuracy, FRR, and FAR we use the largest of the datasets (LibriSpeech) as the unknown dataset and only train the models on the other two.

TIMIT [12] is an acoustic-phonetic continuous speech corpus, which contains clean audio samples (i.e., no background noise) from 630 speakers of eight major dialects of American English recorded at 16 kHz. Each person speaks 10 individual sentences with a mean sample length of 2.7 seconds, resulting in around 3.5 hours of speech.

LibriSpeech [20] is a collection of audio books that contains 360+ hours of English speech data recorded at 16 kHz. It is divided into subsets of varying levels of background noise. We use the clean subset that is 100 times bigger than TIMIT and contains 961 speakers. For comparability with TIMIT, we limit these to 630 speakers.

Real-World dataset[10]**.** We use the Nordic Thingy:53 (housing an nRF5340 SoC) prototyping platform to collect data with a PDM microphone with 16 kHZ sampling rate , to validate that even cheap, ubiquitous PMD microphones can be used for CSSI and OSSI tasks. We use four Nordic Thingy:53 prototyping platforms and place them at 30 cm, 1 m, 2.5 m, and 4 m away in a noise-free office. We collect 25 sentences of 32 people, where all 32 are in English and 28 in German. The sentences are between 1 and 6 seconds long and we incorporate 68.8 % male and 31.2 % female speakers.

4 Model Reduction and Performance Evaluation

First, we describe the library versions we used to train our models (Section 4.1). Then, we use the reverse compound scaling approach (Section 3.3) to shrink the candidate models derived in Section 3.2 and select the models with the best CSSI accuracy that run on an nRF52840 and an nRF5340 (Section 4.2). These models are then used to further evaluate OSSI performance (Section 4.3).

4.1 Model Implementation

`xResNet18` and `ResCNN` are readily available on `time series ai (tsai)`, an open-source deep learning package using PyTorch and fastai, and the `MobileNetV2` is also readily available in PyTorch. Therefore, we use PyTorch version 1.13.1, time series AI (TSAI) version 0.3.5, and fastai version 2.7.13 to train our models offline on our university's high performance server. To eventually run the models on a microcontroller, which is necessary to confirm flash and RAM usage, we need to convert the models from PyTorch to TensorFlow Lite for microcontrollers (TFLM), as PyTorch does not provide a workflow for microcontrollers. For the conversion to TFLM, we use quantization aware training (QAT) and per-channel quantization, the TFLM default for convolutional neural networks.

4.2 CSSI Accuracy and Memory Feasibility

First we evaluate how the CSSI accuracy changes as a function of the model size (i.e., stored in flash memory) to find suitable models for the nRF52840 and nRF5340. We then check if these selected few models actually fit in RAM by converting them from PyTorch to TFLM and evaluate their memory usage.

4.2.1 Flash Memory. Figure 1 shows the decreasing CSSI accuracy over the number of parameters (i.e., model size) of each pre-processing method for each model scaled with a different ϕ values (0, 0.75, 1.5, 2.25, or 3), with $\phi = 0$ being the original model and $\phi = 3$ being the smallest. Please note that the number of parameters is only a coarse approximation of the needed flash memory, and may change after converting the model to TFLM. The evaluation is done with the models `xResNet18`, `MobileNetV2`, and `ResCNN` for the TIMIT (top row) and LibriSpeech (bottom row) datasets, using the Waveform, the LMS, and the MFCC methods. We use the two more established datasets to filter out suitable models and only use our Real-World dataset in a second step.

Dataset differences. In general, models trained on the LibriSpeech dataset achieve higher accuracies compared to those trained on the TIMIT dataset. This variation in performance is expected, since LibriSpeech is approximately 100 times larger, providing significantly more training data for the models. The difference in accuracy between the two datasets is more pronounced when using MFCC as model inputs than when using the Waveform and LMS features as model inputs. Furthermore, this accuracy gap between TIMIT and LibriSpeech is more noticeable for smaller model sizes and gradually decreases as model size increases.

Model differences. Among the evaluated models, the `MobileNetV2` has the highest CSSI accuracy variation across pre-processing methods, and only achieves >90 % when using the waveform as input. The `xResNet18` is more stable across pre-processing methods and, apart from MFCC, it always achieves 89- 99 % CSSI accuracy. The original `ResCNN` performs similarly well as the smallest `xResNet18`, apart from MFCC trained on TIMIT. Down scaling the `ResCNN` drops the CSSI accuracy significantly, ~33 % at worst, while the most up scaled version has the highest CSSI accuracy in LMS and MFCC (>98 % and >93 %, respectively), and is on par in the Waveform (>99 %).

Across models and pre-processing methods, we show the feasibility of running >90 % accurate CSSI even on resource-constrained embedded hardware.

4.2.2 RAM Usage. Two aspects contribute to RAM usage: (i) processing recorded audio samples, and (ii) running inference on them.

[10]Ethical approval obtained (GZ: EK-83/2025); all participants signed a consent form and were informed about the use of the anonymized collected voice samples.

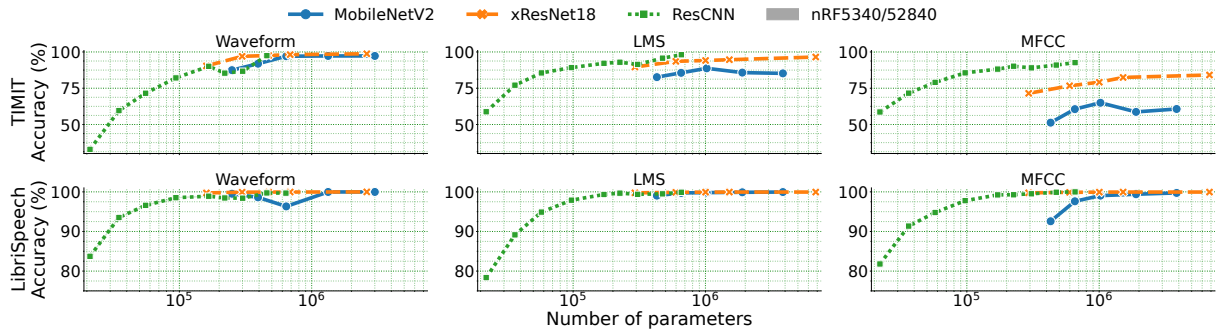


Figure 1: CSSI accuracy over the model size (i.e., number of parameters) of `xResNet18`, `ResCNN`, and `MobileNetV2` using the Waveform, LMS, and MFCC method. The flash memory size of nRF52840 and `nRF5340` estimate how many parameters fit in their flash memory (assuming full integer quantization where one parameter needs 1 B). The evaluation was done on the TIMIT (top row) and LibriSpeech (bottom row) datasets. Our results show that models with >90 % accuracy could potentially run on an nRF5340/nRF52840. The models are scaled with different ϕ values and trained on the TIMIT and LibriSpeech dataset.

We need ~66.6 kB of RAM per second of audio data to process it. Furthermore, the average audio length of our dataset is 3.1 s (Table 1), leading to a ~200 kB of RAM overhead[11]. The PyTorch models are converted to TFLM models that can be run on our target devices. Now we can test if our models actually fit in RAM. The peak RAM usage of a model depends on the target device architecture, compiler settings, and TFLM optimizations. We load individual models onto the nRF52840/nRF5340 and determine the required RAM size through experimentation[12].

`ResCNN` needs too much RAM. The `ResCNN` is the best performing model and fits into flash memory, however, its peak RAM usage exceeds the 512 kB RAM of our target platform, the nRF5340. Only the smallest `ResCNN` (i.e., scaled down with $\phi = 3$) fits into RAM (using 200 kB of RAM), but achieves the lowest accuracy of all models. Additionally, we would only have ~56 kB and ~312 kB of RAM left to process audio samples on the nRF52840 and nRF5340, resulting in <1 s and <5 s of audio than can be recorded, respectively.

`xResNet18` fits in RAM. Despite our simple flash memory approximation in Figure 1, the model conversion and compiler optimizations shrink the model far enough to fit on the device, as can be seen in Table 3. The Waveform, LMS, and MFCC `xResNet18` scaled with $\phi = 0.75$ (from now on referred to as `xResNetSmall`) require 135, 60, and 45 kB of RAM and all need 741.5 kB (TIMIT), ~548 kB (Real-World), or 726.7 KB (LibriSpeech) of flash memory. The `xResNet18` scaled with $\phi = 3$ (from now on referred to as `xResNetTiny`) requires 75, 30, and 20 kB of RAM and 179.2 kB (TIMIT), 109.9 kB (Real-World), or 170.8 KB (LibriSpeech) of flash memory. Considering that we also need to process an average of 3 s of recorded audio samples (Table 1) in RAM, the `xResNetSmall` with MFCC would have a maximum of 11 KB to spare on the nRF5840 for other tasks or an operating system. Hence, the `xResNetSmall` fits onto the nRF5340 for all three pre-processing methods, while the `xResNetTiny` only fits onto the nRF52840 with LMS and MFCC[13]. Compared to `xResNet18`, `xResNetSmall` achieves within ~1 % CSSI accuracy, while needing 16x less flash memory. The `xResNetTiny` models lose between 8-24 % CSSI accuracy compared to `xResNet18` and `xResNetSmall`, but need 67x and 3x less flash memory, respectively. The Waveform achieves the highest CSSI accuracies, with LMS falling shortly behind with 4-6 % less accuracy, and the MFCC losing up to 12-32 % accuracy. For `xResNetTiny` and `xResNetSmall`, Waveform and LMS are preferable. We summarize the characteristics of the most suitable models in Table 2 that will be used for further evaluation.

Table 2: Summary of our three selected models, their target device, and their size reduction.

Model	Device	ϕ	Size reduction	Quantized
`xResNetTiny`	nRF52840	3	67x	Yes
`xResNetSmall`	nRF5340	0.75	16x	Yes
`xResNet18`	-	0	-	No

`MobileNetV2`. As `MobileNetV2` achieves lower accuracies than `xResNet18`, we do not evaluate it's memory impact and continue our evaluations with `xResNet18`, `xResNetSmall`, and `xResNetTiny`.

4.3 OSSI Performance

After finding suitable CSSI models, we need to modify them in order to reject unknown speakers to achieve OSSI. We perform our evaluation on two dataset combinations. The models are trained either on TIMIT or Real-World (in-set) and use the LibriSpeech as the out-of-set dataset. The TIMIT and Real-World test datasets contain ~1.43 and ~1.36 hours of speech, while the LibriSpeech test dataset contains ~74.25 hours, hence, we test on a large portion of out-of-set speakers. First, we evaluate the FAR and FRR (missed in-set speaker) to find the Equal Error Rate (EER) that minimizes both metrics and provides a good trade-off threshold (τ) to detect out-of-set samples. Second, we decrease the FAR by enforcing consensus among segments of an audio sample. We use the results of `xResNetSmall` with TIMIT as in-set and LibriSpeech as out-of-set dataset to showcase our results and then show a table to summarize all results of the `xResNet18`, `xResNetSmall`, and `xResNetTiny` models with the TIMIT and Real-World datasets used as in-set.

[11] Pipelining the recording, pre-processing, and running inference can further optimize the RAM usage, but they are beyond the scope of this work.
[12] This is the advised method by TensorFlow themselves: https://ai.google.dev/edge/litert/microcontrollers/get_started accessed 22.5.2025.
[13] Using the Waveform is feasible, but limits the use of less than 3 s of audio.

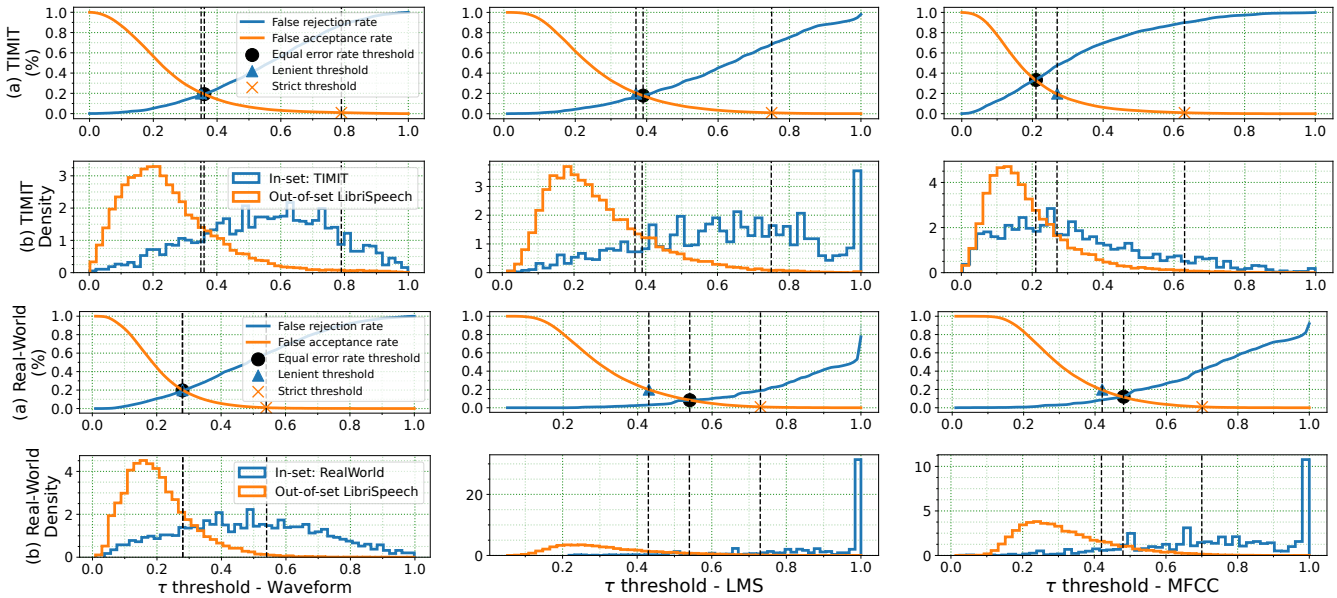


Figure 2: Figures a) show the equal error rate (EER) that denotes the intersection of the false rejection and false acceptance rate. The EER provides a good threshold estimate to separate the in-set and out-of-set distribution shown in the τ threshold histogram of Figures b). The threshold is used to separate in-set and out-of-set samples, the higher the separation, the lower the false rejection and acceptance rates. The in-set dataset is either Real-World or TIMIT and the out-of-set dataset is LibriSpeech.

Table 3: RAM, flash memory, and CSSI accuracy of the `xResNetTiny`, `xResNetSmall`, and `xResNet18` models trained on three different datasets. These models can be run on different device classes, such as the nRF52840, nRF5340, or on a larger device such as the Raspberry Pi, respectively.

Dataset	Model	Device	Method	RAM (kB)	Flash (kB)	Accuracy (%)
TIMIT	`xResNetTiny`	nRF52840	Waveform	75	179.2	82.1
			LMS	30	179.2	76.6
			MFCC	20	179.2	50.9
	`xResNetSmall`	nRF5340	Waveform	135	741.5	97.5
			LMS	60	741.5	92.7
			MFCC	45	741.5	73.9
	`xResNet18`	-	Waveform	-	~12000	96.5
			LMS	-	~12000	92.01
			MFCC	-	~12000	74.7
LibriSpeech	`xResNetTiny`	nRF52840	Waveform	75	170.8	91
			LMS	30	170.8	92.9
			MFCC	20	170.8	84.5
	`xResNetSmall`	nRF5340	Waveform	135	726.7	99.6
			LMS	60	726.7	99.8
			MFCC	45	726.7	98.7
	`xResNet18`	-	Waveform	-	~12000	99.9
			LMS	-	~12000	99.9
			MFCC	-	~12000	99.8
Real-World	`xResNetTiny`	nRF52840	Waveform	75	109.9	82.02
			LMS	30	109.9	89.2
			MFCC	20	109.9	83.8
	`xResNetSmall`	nRF5340	Waveform	135	551.6	89.6
			LMS	60	548	95.6
			MFCC	45	548	92.02
	`xResNet18`	-	Waveform	-	~12000	95.9
			LMS	-	~12000	97.4
			MFCC	-	~12000	93.2

Softmax output (τ) threshold. Figure 2 shows two sets of plots, the upper two rows show the results when using the TIMIT as in-set and LibriSpeech as out-of-set, while the lower two rows show the results with Real-World as in-set and LibriSpeech as out-of-set.

The respective upper rows present the FAR, FRR, and EER for MFCC, LMS and waveform methods as functions of the τ. As the threshold increases from 0 to 1, the FAR decreases while the FRR increases. Based on our defined use-cases in Section 3.4.2, we also highlight the thresholds for the EER (balanced), more strict, and more lenient use-cases. These highlighted thresholds are extended into the respective bottom row. These bottom row figures show the histograms of the τ probability which is not an average output probability but a sum of the output probability over the segment length for both in-set and out-of-set samples. This is done to illustrate not only the confidence of the prediction, but also to show how many segments contribute to the output. For instance, 8 out of 10 segments predicting the same speaker contribute to higher scores than 4 out of 10 segments. Ideally, in-set softmax scores should yield probabilities close to 1, while out-of-set softmax scores should be near 0. In general, a larger overlap between the in-set and out-of-set histograms indicates higher EER, as it becomes more difficult to distinguish between the two.

It is evident that the LMS method results in the smallest overlapping region in the histogram and achieves the lowest EER, indicating better separation between in-set and out-of-set speakers compared to the waveform and MFCC methods. Furthermore, the in-set Real-World and out-of-set LibriSpeech datasets have a better separation compared to TIMIT and LibriSpeech.

Enforcing consensus. The required consensus among audio segments impacts the FAR, which means that a higher number of

Table 4: Threshold τ and consensus threshold used for each of the three use-cases with either the Real-World or TIMIT as in-set and the LibriSpeech as out-of-set datasets. The results are shown as "Real-World | TIMIT".

Model	Method	Lenient		Balanced		Strict	
		τ (%)	con. (%)	τ (%)	con. (%)	τ (%)	con. (%)
xResNetTiny	Waveform	26 | 35	40 | 45	21 | 18	42 | 33	54 | 100	65 | 10
	LMS	37 | 35	50 | 50	39 | 23	56 | 42	69 | 69	75 | 75
	MFCC	36 | 21	50 | 40	34 | 10	49 | 33	70 | 57	75 | 65
xResNetSmall	Waveform	28 | 35	40 | 45	28 | 36	43 | 45	54 | 79	60 | 80
	LMS	43 | 37	50 | 45	54 | 39	66 | 50	73 | 75	80 | 80
	MFCC	48 | 27	50 | 40	48 | 21	59 | 36	70 | 63	80 | 70
xResNet18	Waveform	37 | 38	45 | 45	43 | 39	55 | 47	72 | 91	75 | 90
	LMS	42 | 27	50 | 40	58 | 32	68 | 44	76 | 57	85 | 70
	MFCC	40 | 25	50 | 35	49 | 22	60 | 35	74 | 60	80 | 70

segments within an audio sample must confidently agree on the predicted identity for the system to accept and identify the speaker.

For instance, a consensus of 60% in a 10-segment audio sample means that 6 out of the 10 segments must predict the same speaker for the speaker to be accepted. As the consensus threshold increases, the FAR correspondingly decreases by reducing the likelihood of incorrect acceptances. However, this comes at the cost of an increased FRR, as the system becomes more stringent and may reject legitimate users more frequently. Thus apart from τ, the consensus threshold can be used as a tuning knob depending upon the nature of the application. The FAR and FRR metrics as a function of required consensus for the three methods are shown in Figure 3.

While each method has its unique characteristics in the plot, there are some general trends. The Waveform models and MFCC model trained on TIMIT have a more symmetric shape, meaning that the FRR increases at a similar rate as the FAR decreases, even if the FAR decreases slightly faster. This means that selecting a consensus threshold for more lenient or more strict use-cases will have a a positive effect on the FAR or FRR, but a similar counter effect on the opposing metric. In contrast, the MFCC model trained on the Real-World and LMS model trained on Real-World or TIMIT have a more skewed trend. Here the FRR increases slower than the FAR decreases, especially with the Real-World dataset. This indicates that these models can use more lenient or strict thresholds without having a similar counter effect on the FAR or FRR.

Table 4 summarizes the τs and the consensus thresholds for the `xResNet18`, `xResNetSmall`, and `xResNetTiny` models. The results with TIMIT and Real-World used as in-set are separated with a "|".

4.4 Summary of CSSI and OSSI Results

Table 5 shows the CSSI and OSSI results per pre-processing method per model per in-set dataset. The OSSI results are split in the OSSI accuracy of correctly identifying accepted speakers, the FRR, and FAR and calculated using the balanced, strict, and lenient thresholds from Table 4.

Apart from the strict use-case with `xResNetTiny`, LMS trained on Real-World always reaches at lest 82.4 % OSSI accuracy, and can have an FRR as low as 2.2 in the lenient and an FAR as low as 0.65 in the strict use-case. `xResNetSmall` and `xResNet18` perform mostly within a few percent of one another, making `xResNetSmall` a viable solution for microcontrollers, such as the nRF5340 SoC. However, when trained on the TIMIT dataset, all methods and models drop their OSSI accuracy below 40.95 % in the strict use-case. We suspect, that due to the larger number of speakers in TIMIT (630 versus 32 in Real-World) the models have a harder time differentiating each speaker, which leads to poor OSSI accuracy when paired with strict τ and consensus thresholds. For CSSI on the TIMIT dataset, the Waveform can outperform the LMS and can be a contender, but it underperforms the LMS in OSSI accuracy (apart for `xResNetSmall` in the balanced use-case). For Real-World, LMS also outperforms Waveform in CSSI accuracy. For the strict use-case, Waveform and MFCC drop below 69.5 %, while the LMS remains with 85.2 and 86.2 % for `xResNetSmall` and `xResNet18`, respectively. In general, the `xResNetTiny` model can be used for CSSI, or for OSSI in balanced and lenient use-case reaching at least 82.4 %, if trained on the Real-World dataset with 32 speakers [14]. However, the `xResNetTiny` models perform worse than `xResNetSmall` by 5.4-16.1 % in CSSI, and 9.1-17.4 % in lenient OSSI, 9.4-17.7 % in balanced OSSI, and 33.11-33.9 % in strict OSSI.

5 Consideration of Real-World Constraints

Our evaluation has shown the feasibility of using the `xResNetTiny` with LMS pre-processing for CSSI and OSSI tasks (lenient and balanced thresholds), as well as the `xResNetSmall` for CSSI and OSSI tasks (lenient, balanced and strict thresholds). Additionally, we want to evaluate the implications of collecting varying speech sample lengths for training (enrolling a speaker) and testing (rejecting or identifying a speaker), and the energy and runtime running SPIDER on an nRFThingy:53.

5.1 Varying Speech Length for CSSI

The length of the audio samples matters in two ways for training and testing the model. For training, we want to know how much training data is required to correctly identify the speaker. This is particularly interesting when enrolling new people. For testing, if fewer seconds of audio are required per sample to correctly identify a speaker, then the processing can be done faster which reduces the waiting time of the user for the results. Additionally, longer audio samples require more memory to store and pre-process them, which could require the system to develop a scheme to record and process the data in parallel.

In Table 1, we show that both TIMIT and LibriSpeech datasets have an average audio length of 2.7s and 12.25s respectively, but the TIMIT dataset contains up to 7.13s of audio and LibriSpeech up to 29.3s. For both datasets, the duration of each audio sample is limited to 5, 3, 2, 1, 0.5, and 0.2 seconds and the impact on training and testing the `xResNetSmall` model is evaluated. The effect of audio sample length on the available training data is illustrated in Table 6. For example, when the audio length is capped at 5s, the TIMIT dataset retains nearly 99% while LibriSpeech retains 40%.

Train audio length. The experimental results are presented in Figure 4. Overall, accuracy declines as audio length decreases, with the impact being more pronounced for the TIMIT dataset than for LibriSpeech. This is primarily due to the smaller amount of training data available in TIMIT compared to LibriSpeech for different audio lengths. For both datasets, waveform-based models experience less

[14] We suspect that the model cannot handle the 630 speakers of the TIMIT dataset.

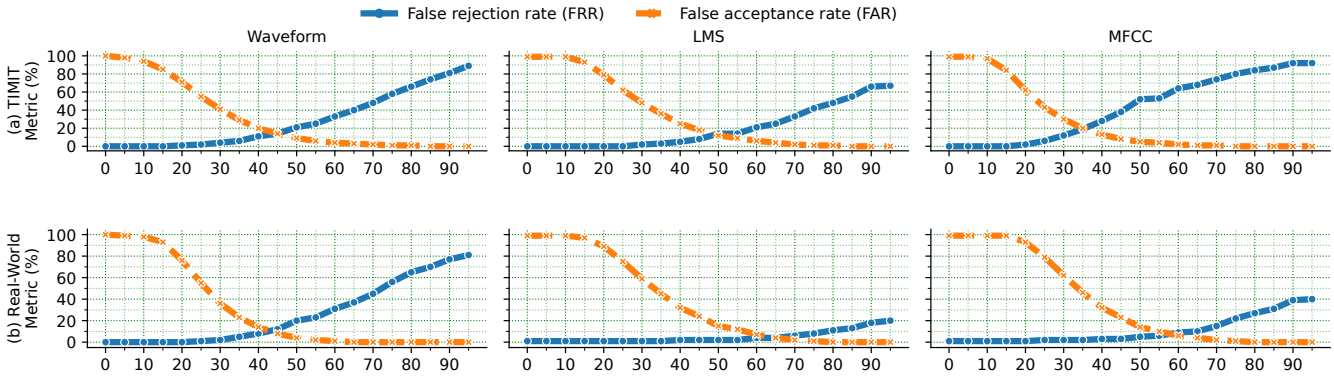


Figure 3: Impact of required consensus on FRR and FAR using, i.e., 60 % consensus of a 10 segment audio sample means 6/10 segments need to predict the same speaker. We use the EER thresholds of Figure 2. x=0 % mimics the default approach without consensus. Increasing the required consensus decreases the FAR, i.e., increases security at the cost of increasing the FRR. Figures (a) use TIMIT and (b) use Real-World as in-set and both use LibriSpeech as out-of-set dataset.

Table 5: CSSI results (accuracy) and OSSI results (accuracy, FRR, FAR) for the tiny, small and full model trained using the Waveform, LMS, or MFCC pre-processing method. The models are trained on the Real-World or TIMIT dataset (in-set), while the LibriSpeech dataset is used as out-of-set. We show the OSSI results in function of the lenient, balanced, and strict use-cases.

Dataset	Model	Method	CSSI results	OSSI results								
				Lenient			Balanced			Strict		
			Accuracy	Accuracy	FRR	FAR	Accuracy	FRR	FAR	Accuracy	FRR	FAR
Real-World	xResNetTiny	Waveform	82	70.4	0.21	17.8	72.4	17.3	16.5	14.4	85.1	0.76
		LMS	89.2	84.2	0.09	16.2	82.4	11.2	11	51.3	46.8	0.85
		MFCC	83.8	72.8	18.9	13.7	77.4	11.3	16.5	33	64.7	0.8
	xResNetSmall	Waveform	90.2	85.2	8.7	14.4	82.8	11.4	10.9	49.9	48.1	0.87
		LMS	**94.6**	**93.3**	**2.3**	**18.2**	**91.8**	**4.4**	**4.9**	**85.2**	**12.1**	**0.81**
		MFCC	91.7	87.9	5.8	14.7	86.4	7.4	8.1	63.8	32.7	0.61
	xResNet18	Waveform	96	93.8	0.31	15.7	89.2	8.3	6.7	52.9	46.6	0.9
		LMS	97.4	95.72	2.2	16.8	93.9	4.4	3.8	86.2	12.5	0.65
		MFCC	93.2	90.9	4.1	15.3	87.9	7.6	7.2	69.6	27.9	0.73
TIMIT	xResNetTiny	Waveform	82.1	26.9	73	17.1	59.4	38.9	40.1	5.7	91.5	0.25
		LMS	76.6	44.3	5.3	15	66.2	25.5	28.4	7.83	92.1	0.82
		MFCC	50.9	32.1	59.5	14.3	47	21.4	27.4	3.8	95.9	0.9
	xResNetSmall	Waveform	97.5	85.4	14.4	14.6	85.2	14.6	14.6	13.81	86.2	0.96
		LMS	**92.7**	**88.3**	**8.6**	**18.8**	**83.9**	**14.1**	**12.1**	**40.95**	**58.8**	**0.87**
		MFCC	74	60.9	30.7	13.2	67.7	19.4	19.3	14.97	84.7	0.85
	xResNet18	Waveform	96.5	86.4	13.1	18.15	84.4	15.1	16.2	5.24	94.7	0.99
		LMS	92	89.5	6.2	12.5	88	8.6	9.1	59.2	40.5	0.6
		MFCC	74.8	69.9	16	16.3	69.3	15.8	16.3	22.7	76.8	0.7

Table 6: Audio length impact on training dataset size. TIMIT and LibriSpeech have a total training dataset size of 3.3 and 170 hours, respectively.

Audio limit (s)		5	3	2	1	0.5	0.2
Train dataset (%)	TIMIT	99	92	71	37	18	7
	LibriSpeech	40	24	16	8	4	2

performance degradation compared to those using LMS and MFCC features. In the case of TIMIT, accuracy drops significantly when the audio length is reduced below 2s, whereas for LibriSpeech, this decline becomes noticeable below 0.5s. Limiting TIMIT audio to 1s reduces the available training data to 71% (approximately 2.3 hours), whereas capping LibriSpeech at 0.5s still results in 6.8 hours of data, despite representing only 4% of the total training dataset.

Test audio length. Identification accuracy declines as the length of test audio decreases. Across various audio lengths, LibriSpeech consistently outperforms TIMIT which is likely due to the significantly larger volume of training data available in LibriSpeech. As observed during training, waveform-based models continue to deliver better performance compared to LMS and MFCC models, particularly when dealing with test samples shorter than 1s. When capped at 2s, both LMS and MFCC models (1 s for LMS trained on LibriSpeech) show a modest accuracy decrease of less than 5 %, which may still be acceptable for some applications. However, reducing the audio length any further, leads to a sharper decline in accuracy of at least 15 %. Across all models and datasets, the performance deteriorates significantly as audio duration falls below 2s. Therefore, to ensure

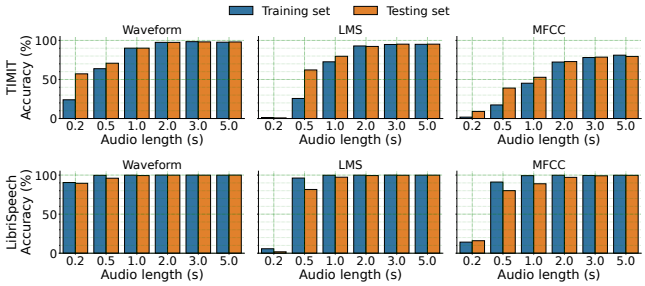


Figure 4: CSSI Accuracy decreases with limited audio length per training and testing audio sample. Using the raw waveform or LMS we can limit the length of audio samples to at least 2s without decreasing the accuracy by more than 3.7 %.

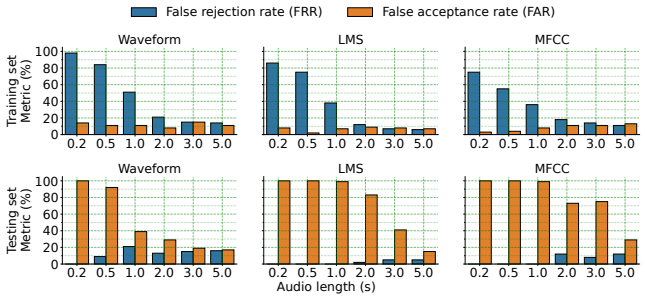


Figure 5: Impact of the training and testing audio length accepting or rejecting speakers. We evaluate the FRR and FAR using the balanced τ and consensus thresholds of Table 4.

reliable speaker identification using lightweight models, recording at least 2s of audio is recommended.

5.2 Varying Speech Length for OSSI

Train audio length. We depict the number of training samples needed to enroll a speaker by capping the audio length of our training set (as we did for SI). Ideally, we want to reduce the number of required training data to a minimum without affecting performance. The top row of Figure 5 shows the FRR and FAR for different training set sizes. Across all three methods, the training set size has little impact on the FAR, with a mean of 11.7, 6.83, and 8.33 % and only a standard deviation of 2.5, 2.48, and 4.08 % for the Waveform, LMS and MFCC, respectively. However, the FRR decreases a lot with increasing training set size from at least 86 % down to at least 14 %. Similarly to SI, at 3 s audio length, the FRR starts to settle at low percentages of 15, 7, and 14 % for Waveform, LMS, and MFCC respectively. These 3 s of audio length in the TIMIT dataset translate into 3.036 hours of training data (~17.3 s per speaker) to ensure that the FRR drops below 15 %. Out of the three methods at 3 s audio capping, LMS achieves half the FRR (with 7 %) of MFCC and Waveform, and the lowest FAR with 8 % compared to MFCC's 11 % and Waveform's 15 %, making it the preferable pre-processing method.

Test audio length. Varying the test audio length tells us how quickly our models can identify a speaker. An increase in testing set audio length improves the FAR significantly from 100 % down to 17, 15, and 29 % for Waveform, LMS, and MFCC, respectively. Waveform is the first to drop below 20 % FAR at 3 s audio length, while LMS needs 5 s, and MFCC does not manage to have a FAR lower than ~29 % within 5 s of testing audio length. The FRR does not increase much with increased audio length with a mean of 12.3, 2, and 5.33 %, and a standard deviation of 7.2, 1.45, and 6.02 % for Waveform, LMS, and MFCC respectively. At least 3-5 s of audio samples are needed to decrease the FAR below 20 %, or a higher consensus threshold is required for stricter applications. Both, however, at the cost of slightly increasing FRR. The Waveform is the best method to reduce the testing audio length with 3 s.

5.3 On-device Execution

We toggle GPIO pins of the Nordic nRF52840 SoC and Nordic nRF5340 SoC (built in to the nRFThingy:53 to measure the average current during pre-processing and inference, and record the required runtime. We then calculate the energy needed by multiplying the runtime, current consumption and voltage (3.3 V for both boards). For ease of use, we report the energy and runtime of a 1 s audio sample, which can then be scaled for arbitrary lengths of speech.

Table 7 shows the energy and runtime usage of the `xResNetTiny` and `xResNetSmall` models running on the nRF52840 SoC and the nRF5340 SoC. Additionally, we evaluate the `xResNetTiny` model on the nRF5340 SoC to exemplify a use case, where SI needs to run side by side with other tasks running on the same hardware and cannot dedicate all its memory and CPU to the SI task. The runtime and energy are affected by the time it takes to pre-process the audio data and to run inference. Most notably, the Waveform does not require any pre-processing like the LMS or MFCC method, however, the model has to deal with a larger input space. This larger input space leads to more RAM usage that makes the Waveform method unusable on the nRF52840 SoC to also record and store the audio samples, which is why it is omitted in the table. On the nRF5340 SoC, the `xResNetTiny` and `xResNetSmall` models take 3.38 and 16.6 s of inference time to process a 1 s audio signal, resulting in 212.7 and 1026 mJ of energy needed, respectively. On the contrary, the LMS and MFCC methods are 11-25 times faster during inference, but require additional time to pre-process the data. Using the built-in DSP instructions, this extra pre-processing time is always below 0.76 s and comparatively negligible compared to the overhead of using the Waveform method. Notably, the extra discrete cosine transformation (DCT) done by the MFCC and not by the LMS method, reduces the frequency features from 40 to 13, resulting in around half the inference runtime, while only needing 2-3 % longer for pre-processing. In total, MFCC is the most efficient needing between 0.92-1.24 s of runtime and 42.52-81 mJ of energy to pre-process and evaluate a 1 s audio sample. LMS needs 20-60 % more runtime (1.1-1.99 s) and 20-54 % more energy (50.9-124.8 mJ) than MFCC, while the Waveform method needs 5-13 times more runtime (3.38-16.6 s) and 4-13 times more energy (212.7-1025 mJ).

Table 7: Runtime and energy of the nRF52840 running `xResNetTiny` with LMS and MFCC, and the nRF5340 running the `xResNetTiny` and `xResNetSmall` model using the Waveform, LMS, and MFCC as pre-processing methods.

Model	Device class	Method	Runtime / Energy for 1 s audio					
			Pre-processing		Inference		Total	
			(s)	(mJ)	(s)	(mJ)	(s)	(mJ)
xResNetTiny	nRF52840	LMS	0.74	34.2	0.36	16.7	1.1	50.9
		MFCC	0.76	35.5	0.16	7.02	0.92	42.52
xResNetTiny	nRF5340	Waveform	-	-	3.38	212.7	3.38	212.7
		LMS	0.56	37.5	0.32	20.4	0.88	57.9
		MFCC	0.57	38.2	0.16	9.86	0.73	48.6
xResNetSmall	nRF5340	Waveform	-	-	16.6	1026.03	16.6	1026.03
		LMS	0.56	37.5	1.43	87.3	1.99	124.8
		MFCC	0.57	38.7	0.67	42.3	1.24	81

6 Use Case Considerations

We set out to identify and answer key questions a real world deployment of SPIDER would face.

How small can a useful model be? The LMS `xResNetTiny` is 93.4 % smaller than the `xResNet18`, and still achieves 89.2 % CSSI accuracy and ≥ 82.4 % OSSI accuracy. The FRR and FAR also stays below 11.2 % and 16.2 %, respectively, making it a tiny yet useful model for CSSI and lenient/balanced OSSI tasks. However, the `xResNetTiny` seems to lose the complexity to handle many speakers, as these results hold true for the Real-World dataset (32 speakers) but fall below 76.6 % for the TIMIT dataset (630 speakers).

How much accuracy do we lose when shrinking the model? The LMS `xResNetTiny` loses 8-24 % CSSI accuracy and 11.5-35.9 % OSSI accuracy for 93.4 % model size reduction. In contrast, the LMS `xResNetSmall` is 75.8 % smaller than `xResNet18`, but only loses 2.8 % (Real-World) and even gains 0.7 % (TIMIT) CSSI accuracy, and loses 1-18.2 % OSSI accuracy.

How many speakers can we support? The LMS `xResNetSmall` can support CSSI and OSSI (lenient and balanced) for 32 (Real-World) and 630 (TIMIT) speakers, reaching CSSI and OSSI accuracies between 83.9-94.6 %. The OSSI strict use-case can only be supported for 32 speakers reaching 85.2 % OSSI accuracy, as it falls to 40.95 % OSSI accuracy with 630 speakers.

How long do we need to enroll a new speaker? From Figure 4 we know that we should not cap TIMIT audio samples by more than 2 s. According to Table 6 this translates into a dataset containing 2.3 hours of audio, meaning that for 630 speakers we need at least 13.4 s per speaker for CSSI. Using the same procedure for OSSI, we need at least 17.3 s per speaker to have an FRR of 7 % and an FAR of 8 % when using the LMS `xResNetSmall`.

How long do they need to speak? At least 2 s (Figure 4) of audio are required for CSSI, and at least 5 s (Figure 5) of audio are required to keep a low FAR in OSSI, based on an evaluation carried out on the TIMIT dataset (630 speakers).

How likely are false acceptances, and how long would it take to fool the system? For the strict use-case, the LMS `xResNetSmall` can keep the FAR as low as 0.81 %, meaning that an unknown speaker needs to speak roughly 123.5 times to bypass the system. In comparison, in balanced and lenient use-cases an unknown speaker would need to speak 20.4 and 5.5 times to fool the system, respectively. Furthermore, it is advised to record 5 s or more of speech for more strict OSSI, which takes ∼21.3 s to accept and identify the speaker. This results in a potential adversary spending at least ∼54, ∼9, or ∼2.4 minutes to fool strict, balanced, or lenient SPIDER, respectively. This might not be enough for secluded environments; however, in busy places, standing around for 54 minutes speaking to oneself will likely draw attention.

What example use-cases could be practical for IoT devices? One additional benefit of on-device CSSI and OSSI is privacy, as no personal data needs to be sent to a cloud. Hence, any use-case focused on not sharing personal data can benefit from SPIDER. CSSI or lenient/balanced OSSI tasks could be to keep track of meeting attendees where mostly internal and some external visitors could join. Another CSSI or lenient/balanced OSSI task can be personalized home automation based on the speaker, such as changing the temperature of a room or reading aloud the agenda of today's calendar at home, or personalizing the sound profile select when wearing headphones/earbuds. In these cases, it is not as critical to reject every unknown speaker, yet it improves the accuracy of the attendee list or does not perform unnecessary actions if the person is unknown.

In contrast, access to a device (e.g., smart bike lock) or a room, access control to machinery, or varying security functions based on the present speaker (e.g., employee versus head of the company) are more strict OSSI use-cases. For these use-cases, strict SPIDER keeps the FAR below 0.81 %, which would take ∼54 minutes to fool the system, while achieving 85.2 % OSSI accuracy.

Which model is the best choice for which device? We provide insights into hand-selected solutions based on the chosen ϕ values, however, the best model depends on the target device and the device usage. For instance, a dedicated device focusing on SI can use all the memory and processing time for the model and audio recordings (66 kB in RAM per 1 s of audio), while other devices might perform multiple tasks and need to share resources. If our provided solutions do not match the user's application and/or device, the user can define the specific memory constraints of their target device, adapt our open-source[2] code, and use a more fine-grained ϕ search to derive a more appropriate model.

7 Conclusion and Future Work

In this paper we have introduced SPIDER, an open-source, lightweight, on-device CSSI and OSSI solution capable of running on constrained embedded platforms embedding as little as 256/512 kB of RAM and 1 MB of flash memory, such as the nRF5340 and nRF52840 SoC. Our own open-source real-world dataset, as well as two large publicly available datasets are used to validate the CSSI and OSSI performance. We have shown that `SPIDER`'s OSSI implementation can also be adapted towards more lenient and more strict use cases, which affects the FRR and FAR. SPIDER works on any device supporting TFLM, has 360/390 kB of RAM and 110/548 kB of flash memory, and has access to cheap PDM microphones that can be found or easily mounted on resource-constrained embedded devices. This enables user identification on a scale that is not possible with alternative technologies such as cameras or fingerprint readers. Additionally, our results underline the feasibility of applying a reverse compound scaling approach for SI tasks, while achieving high CSSI and OSSI accuracy and reducing the model size to fit on devices with less than 1 MB of flash memory. This shows potential to be applied to other speech tasks, and, also, classification on time

signals (beyond audio), for instance, for other wearables (ECG/EEG classification in smartwatches, as well as event classification for continuous glucose monitors), for industrial applications (e.g., vibration sensors, acoustics and acoustic emissions monitoring), and for biomedical signals (e.g., phonocardiograms, and sleep stage identification).

Limitations. Our FRR and FAR metrics are based on the widely-used MSP method, however, more distance-based metrics (e.g., Mahalanobis or Deep k-NN) could be explored. Our model selection process is not automatic as it depends on the available memory and accuracy requirements of the application, i.e., is it a dedicated device for a single tasks or does it also need to run other tasks? A way to automate this with user specific requirements would be beneficial, as well as optimizing the hyper-parameter search to replace our pre-defined ϕ-value selection. This study is conducted using noise-free audio samples to assess the fundamental feasibility of the proposed solutions. Dedicated experiments to tackle real-world noise conditions need to be done using data augmentation techniques and noise reduction filters. The audio length required for speaker enrollment depends on the number of speakers. With TIMIT (630 speakers) 13.4s of audio length is sufficient. With our RealWorld dataset ~12.8 minutes are required as only 32 speakers are included. In future work, we will investigate how to reduce the required training data, as it might be limiting in practical settings to obtain long audio samples for enrollment. We used a 630 and a 32 speaker dataset, but did not gradually investigate the role of the number of enrolled speakers. A more fine-grained analysis on the same dataset could yield more insights of SPIDER's performance as a function of the number of speakers.

Acknowledgments

The authors would like to thank Jesús Pestana for providing feedback and guidance on this paper, and Tobias König for his initial investigation into the viability of other pre-processing methods in terms of CSSI, energy, and runtime, which helped to shape the direction of this paper. This work was conducted within the SPiDR project ("Secure, Performant, Dependable, and Resilient Wireless Mesh Networks") financed by the Technology Innovation Institute. This publication has also emanated from research conducted with the financial support of Research Ireland under Grant number 19/FFP/6775.

References

[1] Julien Balian, Raffaele Tavarone, Mathieu Poumeyrol, and Alice Coucke. 2021. Small Footprint Text-Independent Speaker Verification For Embedded Systems. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6179–6183. doi:10.1109/ICASSP39728.2021.9413564

[2] J.P. Campbell. 1997. Speaker recognition: a tutorial. *Proceedings of the IEEE* 85, 9 (1997), 1437–1462. doi:10.1109/5.628714

[3] Diego Castán Lavilla, Md Rahman, Sarah Bakst, Chris Cobo-Kroenke, Mitchell McLaren, Martin Graciarena, and Aaron Lawson. 2022. Speaker-Targeted Synthetic Speech Detection. In *The Speaker and Language Recognition Workshop (Odyssey 2022)*. 62–69. doi:10.21437/Odyssey.2022-9

[4] João Antônio Chagas Nunes, David Macêdo, and Cleber Zanchettin. 2020. AM-MobileNet1D: A Portable Model for Speaker Recognition. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–8. doi:10.1109/IJCNN48605.2020.9207519

[5] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. 2022. WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing. *IEEE Journal of Selected Topics in Signal Processing* 16, 6 (2022), 1505–1518. doi:10.1109/JSTSP.2022.3188113

[6] Yafeng Chen, Siqi Zheng, Hui Wang, Luyao Cheng, Qian Chen, and Jiajun Qi. 2023. An Enhanced Res2Net with Local and Global Feature Fusion for Speaker Verification. In *Interspeech 2023*. 2228–2232. doi:10.21437/Interspeech.2023-1294

[7] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. 2020. ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification. In *Interspeech 2020*. 3830–3834. doi:10.21437/Interspeech.2020-2650

[8] Divy Dwivedi, Ashutosh Ganguly, and V.V. Haragopal. 2023. 6 - Contrast between simple and complex classification algorithms. In *Statistical Modeling in Machine Learning*, Tilottama Goswami and G.R. Sinha (Eds.). Academic Press, 93–110. doi:10.1016/B978-0-323-91776-6.00016-6

[9] Junaid Iqbal Emon, Md Ashraful Salek, and Khairul Taufique Alam. 2025. Whisper Speaker Identification: Leveraging Pre-Trained Multilingual Transformers for Robust Speaker Embeddings. https://arxiv.org/abs/2503.10446. doi:10.48550/arXiv.2503.10446

[10] Ziliang Fan, Meng Li, Shuo Zhou, and Bo Xu. 2020. Exploring wav2vec 2.0 on Speaker Verification and Language Identification. https://arxiv.org/abs/2012.06185. doi:10.48550/arXiv.2012.06185

[11] Shai Fine, Jiri Navratil, and R.A. Gopinath. 2001. A hybrid GMM/SVM approach to speaker identification. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, Vol. 1. 417–420 vol.1. doi:10.1109/ICASSP.2001.940856

[12] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, and Victor Zue. 1992. TIMIT Acoustic-phonetic Continuous Speech Corpus. *Linguistic Data Consortium* (11 1992). doi:10.35111/17gk-bn40

[13] Petko Georgiev, Sourav Bhattacharya, Nicholas D. Lane, and Cecilia Mascolo. 2017. Low-resource Multi-task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations. *Proceedings of the ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 50 (Sept. 2017), 19 pages. doi:10.1145/3131895

[14] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of Tricks for Image Classification with Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/CVPR.2019.00065

[15] Dan Hendrycks and Kevin Gimpel. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *International Conference on Learning Representations*. doi:10.48550/arXiv.1610.02136

[16] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

[17] Umair Khan, Miquel India, and Javier Hernando. 2020. I-Vector Transformation Using K-Nearest Neighbors for Speaker Verification. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7574–7578. doi:10.1109/ICASSP40776.2020.9053504

[18] Bei Liu and Yanmin Qian. 2024. Memory-Efficient Training for Deep Speaker Embedding Learning in Speaker Verification. *arXiv preprint arXiv:2412.01195* (2024). doi:10.48550/arXiv.2412.01195

[19] Thi-Thanh-Mai Nguyen, Duc-Dung Nguyen, and Chi-Mai Luong. 2024. Vietnamese Speaker Verification With Mel-Scale Filter Bank Energies and Deep Learning. *IEEE Access* 12 (2024), 150114–150122. doi:10.1109/ACCESS.2024.3479092

[20] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5206–5210. doi:10.1109/ICASSP.2015.7178964

[21] Lawrence Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of speech recognition*. Prentice-Hall, Inc.

[22] Elena Rodríguez, Belén Ruíz, Ángel García-Crespo, and Fernando García. 1997. Speech/speaker recognition using a HMM/GMM hybrid model. In *Audio- and Video-based Biometric Person Authentication*, Josef Bigün, Gérard Chollet, and Gunilla Borgefors (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 227–234. doi:10.1007/BFb0016000

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 4510–4520. doi:10.1109/CVPR.2018.00474

[24] M. S. Sinith, Anoop Salim, K. Gowri Sankar, K. V. Sandeep Narayanan, and Vishnu Soman. 2010. A novel method for Text-Independent speaker identification using MFCC and GMM. In *2010 International Conference on Audio, Language and Image Processing*. 292–296. doi:10.1109/ICALIP.2010.5684389

[25] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. 2018. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*

(ICASSP). 5329–5333. doi:10.1109/ICASSP.2018.8461375

[26] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur. 2016. Deep neural network-based speaker embeddings for end-to-end speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*. 165–170. doi:10.1109/SLT.2016.7846260

[27] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114. doi:10.48550/arXiv.1905.11946

[28] Mingxing Tan and Quoc Le. 2021. EfficientNetV2: Smaller Models and Faster Training. doi:10.48550/arXiv.2104.00298

[29] Jenthe Thienpondt and Kris Demuynck. 2023. ECAPA2: A Hybrid Neural Network Architecture and Training Strategy for Robust Speaker Embeddings. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 1–8. doi:10.1109/ASRU57964.2023.10389750

[30] Ivette Vélez, Caleb Rascon, and Gibrán Fuentes-Pineda. 2020. Lightweight speaker verification for online identification of new speakers with short segments. *Applied Soft Computing* 95 (2020), 106704. doi:10.1016/j.asoc.2020.106704

[31] Ching-Chen Wang, Ching-Te Chiu, and Jheng-Yi Chang. 2022. EfficientNet-eLite: Extremely Lightweight and Efficient CNN Models for Edge Devices by Network Candidate Search. *J. Signal Process. Syst.* 95, 5 (Sept. 2022), 657–669. doi:10.1007/s11265-022-01808-w

[32] Hatem Zehir, Hafs Toufik, and Sara Daas. 2023. TinyCNN: An Embedded CNN Model for Speaker Identification Using ESP32.

[33] Xiaowu Zou, Zidong Wang, Qi Li, and Weiguo Sheng. 2019. Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification. *Neurocomputing* 367 (2019), 39–45. doi:10.1016/j.neucom.2019.08.023