POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Cascading on-device keyword spotting and speaker verification in TinyML

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Gioele Mombelli

Advisor: Prof. Manuel Roveri

Co-advisors: Ing. Massimo Pavan, Ing. Marco Cova

Academic year: 2021-2022

1. Introduction

In recent years, there has been an increasing demand for human-machine interfaces that operate through voice. These interfaces are becoming necessary in a wide range of systems and are increasingly required in resource-limited devices, and this is why speech processing is a field that is particularly prone to be analyzed under the Tiny Machine Learning (Tiny-ML) domain. This work aims at investigating techniques for integrating speech processing capabilities on ultra-low power microcontrollers, focusing on the tasks of keyword spotting (KWS) and speaker verification (SV). Research in this field is essential as memory, computational power, and energy consumption constraints pose significant challenges to the complexity of algorithms that can be executed. While the ability of recognizing commands has already been extensively targeted by research, the verification of a speaker's identity through voice is a needed feature which has not been organically analyzed in a Tiny-ML context yet; we framed the speaker verification task in a novel Tiny-ML oriented one-class few shot classification context. This has been done to take into account the real use-cases for a tiny speaker verification system: there is the need to perform on-device incremental adaptation to the voice of a speaker by observing only a limited set of samples from his/her voice (*few shot*), and without having information about other voices as comparison (*one-class*). This work demonstrates the possibility of designing a system that combines both tasks with a request for resources compatible with the capabilities offered by modern microcontrollers, proposing a comparison between different approaches: some already present in the literature, but never used in an organic Tiny-ML approach, others identified and tested for the first time in this work.

2. High-level overview

The goal is to develop a speaker verification system that can work cascaded with a keyword spotting system, to provide a sort of personalized user interaction: if a keyword is deemed coming from the enrolled speaker, a personalized answer is given. Otherwise, a general answer is returned by the system. The high-level block scheme of the application is described in figure 1:

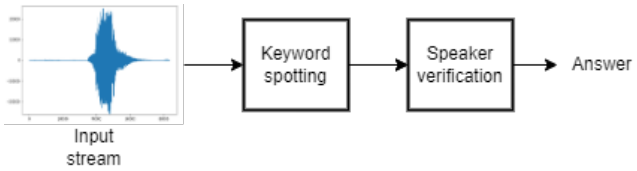

Figure 1: High-level scheme of the KWS-SV application.

To reach the desired behavior, KWS and SV blocks must be designed to work together. Given the Tiny-ML context this work refers to, the more parts can be shared between the two applications, the less space and computation will be required on the microcontroller to run the system. Regarding the KWS part, we relied on the implementation described by [4], with some fine tuning regarding neural network architecture and data preprocessing, porting the system on the target architecture for the first time.

The SV part was analyzed by comparing different algorithms and choosing the most promising one for on-device implementation. Among all the algorithms tested, some were never been used before for speaker verification strictly in a Tiny-ML context, and one is also proposed for the first time in this work.

3. KWS system

The core of the KWS system is composed by a machine learning model that has to process audio data to provide a confidence value on the presence of the chosen keywords. The model adopted is a Convolutional Neural Network (CNN) that processes input audio data in the shape of MFCC features:

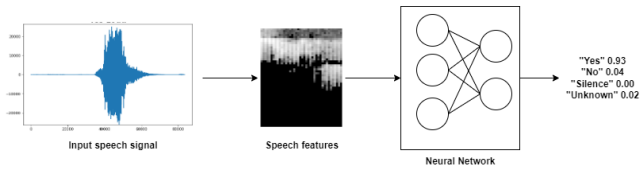

Figure 2: Steps of the KWS system based on deep learning.

The following steps are the main transformations applied to a 1-second long audio window to obtain the related MFCC spectrogram, in the shape of a 49x40 matrix:

- Hanning Windowing;
- Fast Fourier Transform;
- Mel-Frequency Downsampling.

The neural network for keyword spotting is trained on a supervised manner with the target of recognizing different keywords, but also the presence of silence in input audio and unknown speech not associated to a specific keyword. The training of the model has been performed on keyword samples from the Google Speech Commands dataset [5], augmented with some additional noise to make the model more robust to different environments.

The choice of the CNN architecture has been guided by the final goal of developing a model small enough to fit into microcontrollers. This posed a limitation on deepness and number of parameters of the models, such as filters' size and neurons per layer. For our purposes we adopted a network with two convolutional layers, each followed by a maxpooling layer, with a final softmax classifier with as many neurons as the desired keywords plus silence and unknown conditions. Table 1 reports accuracy scored during testing according to the number of chosen keywords of the best performing architecture.

Model: conv-kws-nn			
Keywords	**Accuracy**	**Loss**	**Params**
1	0.9450	0.1743	25,971
2	0.9269	0.2319	26,932
3	0.8960	0.2900	27,893

Table 1: Accuracy results for KWS CNN.

Different quantization levels could be applied to the aforementioned model in order to reduce its size and fit into the target device, as shown in table 2. However, the floating point version is still small enough to be executed on the target architecture adopted for this work.

Model: conv-kws-nn			
Quantization	**Weights**	**Activations**	**Latency**
float	112.5 kB	79.3 kB	0.336 s
int16x16	56.4 kB	45.6 kB	0.268 s
int16x8	28.3 kB	45.6 kB	0.302 s
int8x8	28.3 kB	25.8 kB	0.309 s

Table 2: Memory and latency estimations for KWS CNN - 3 keywords version.

4. SV system

Speaker verification has been tackled coupling a deep neural network model to be used as feature extractor for speaker characteristics and a similarity algorithm to associate them to a specific speaker, called *enrolled speaker*. Speaker features extracted by the neural network are called *d-vectors* and have been proposed for the first time in [3].

Speaker verification task is composed by two distinct phases:

1. Enrollment phase;
2. Verification phase.

During the enrollment phase, a small subset of speech samples is collected from the enrolled speaker. The feature extractor neural network processes them and extracts a set of d-vectors from such samples, forming the enrollment set. During verification phase, a similarity algorithm acts on new input speech by extracting the related d-vector and comparing it with the enrollment set according to a similarity algorithm.
This particular setting allows us to frame the speaker verification task into two specific contexts:

- One-Class Classification;
- Few-shot Classification.

In one-class classification (OCC), machine learning algorithms have to learn to distinguish objects of a specific target class amongst all objects belonging to all possible existing classes, by learning from a training set containing only samples from the target class. Few-shot classification aims to learn a classifier to recognize unseen classes during training with limited labeled examples. This is what led us to testing some algorithms specifically designed for such conditions.

4.1. D-Vector Extractor

The d-vector extractor adopted in this work is a convolutional neural network trained on a sufficiently large speakers dataset to perform classification among them. As explained in [3], d-vectors can be taken as activations of hidden layers of such a model. The core idea is that each speaker is able to produce a unique d-vector, even if the related voice was not present in the training dataset.
To maintain compatibility with the KWS system, we designed the d-vector extractor to be able to take as inputs the same MFCC spectrograms of the KWS model. The neural network adopted is similar but deeper than the ones used in keyword spotting task. A batch notmalization layer has been introduced right after the input layer. The network presents two 2DConvolutional-maxpooling blocks, ending with two 2DConvolutional layers followed by a dense layer. Dropout is added to control overfitting, and the last output layer is a softmax classifier with as many nodes as the number of speakers in the training dataset, which is 92 in the chosen implementation, taken from the *librispeech-train-100* partition of [2]. D-vectors have been taken as the flattened activation of the last convolutional block, before the first fully connected layer of the model.

The full classifier scored an accuracy of 0.5778 on the testing set, and the d-vector extractor model contains 24,388 parameters, which make it extremely lightweight considering the complexity of the task to be solved. The size of the d-vectors is 256 elements. In table 3, memory and latency estimations for the d-vector extractor model are provided.

Model: d-vector-extractor-256			
Quantization	**Weights**	**Activations**	**Latency**
float	98.08 kB	70.5 kB	0.036 s
int16x16	49.2 kB	43.1 kB	0.028 s
int16x8	24.8 kB	43.1 kB	0.032 s
int8x8	24.8 kB	25.5 kB	0.033 s

Table 3: Memory and latency estimations the d-vector extractor NN.

4.2. SV Classification Approaches

The d-vector extractor backbone must be coupled with a classification or similarity algorithm that has the burden of actually verifying the claimed identity of a speaker, via the analysis of the produced d-vectors. Four different similarity algorithms have been tested.

4.2.1 Mean Cosine Similarity

This approach is the state-of-the-art regarding speaker verification systems. A pre-determined

number of enrollment samples from the enrolled speaker is obtained, and is processed through the d-vector extractor to produce a set of enrollment d-vectors. The element-wise average of the d-vectors is computed to obtain a mean d-vector to be used as speaker model.
Every time a new audio utterance is detected, the system extracts the related d-vector and confronts it with the mean d-vector of the enrolled speaker via cosine similarity (equation 1):

$$cossim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}} \tag{1}$$

If the similarity is above a certain threshold, determined by the designer in an empirical way according to sensitivity needs, the utterance is evaluated as belonging to the enrolled speaker.

4.2.2 Best-Match Cosine Similarity

This is a novel approach to comparing d-vectors of different speakers. Similarly to the previous method, a pre-determined number of enrollment samples from the enrolled speaker is used to produce a set of d-vectors. This set is kept in memory and is called enrollment set. During inference, every time a new audio utterance is received, the system extracts the related d-vector and compares it with each d-vector in the enrollment set via cosine similarity. The system returns the best-matching similarity, i.e. the highest similarity scored during the one-to-one comparison. If the similarity is above a certain threshold, the utterance is attributed to the enrolled speaker.

4.2.3 One-Class Neural Networks

This approach has been inspired by the work done in [1], and is a classification method that involves training a one-class Neural Network (OCNN) to distinguish between d-vectors produced by the authenticated speaker and by other unknown speakers. The set of enrollment d-vector is used to produce a small training set by coupling them with fictitious d-vectors representing non-authenticated samples. The architecture chosen for the one-class neural network is composed by a batch normalization layer, a dense layer with as many neurons as the size of the d-vector and a softmax layer with two outputs for the enrolled/unknown classes. The network has a total of 67,330 parameters.

The advantage of this approach is that such an algorithm could learn to map d-vectors produced by a tiny feature extractor to the correct speaker in a more efficient way. However, it requires to run a full training process directly on-device. This poses serious limitations on the one-class neural network architectures that can be built, both regarding size and type of the adopted layer.

4.2.4 One-Class SVM

This method involves training a one-class Support Vector Machine (OC-SVM) to learn a boundary that separates efficiently d-vectors of the enrolled speaker from d-vectors of other speakers. The approach is similar to the One-Class Neural Network: a set of enrollment d-vectors is used for fitting the OC-SVM, which is then used as a discriminator on new samples. While a more efficient classificator could be learned with this approach, fitting a one-class SVM requires heavy computation for solving their constrained optimization problem, which may not be feasible in more constrained devices.

4.3. Testing results

Two custom datasets have been collected for testing the aforementioned classification methods. Four speakers have been asked to collect 100 text-dependent samples each, pronouncing the phrase "Hey Cypress", from which 1-second long utterances containing speech have been extracted. We also collected a custom Italian text-independent dataset from the same people, to have testing benchmarks on both applications of speaker verification on the same subset of speakers. Each approach proposed has been tested against the two different datasets.

The one-class condition has been enforced by enrolling one speaker at a time and using only samples from that speaker to perform the enrollment. Each speaker in the datasets has played the part of the enrolled speaker, and each time samples from other speakers have been used to build the set of "unknown" speakers. The few-shot conditions have been enforced by posing a

limitation on the number of enrollment samples to be used. Reasonable samples number for few-shot classification were deemed to be 8 and 16 samples, but we decided to identify also two corner cases, i.e. the case with only 1 enrollment sample and the case with 64 enrollment samples. Testing and validation datasets are composed by 60 samples each, and have been built by taking 15 samples from the enrolled speaker, and 45 samples by taking 15 samples from all the other speakers.
Evaluation metrics adopted are accuracy and F1-Score, because they are the only metrics that can be applied to all the methods identified in previous section. Given the fact that testing datasets are unbalanced, F1-Score is the metric that better represents the performance of the systems.
In figures 4 and 3, comparison of performance of different similarity systems in a text-dependent context is reported.

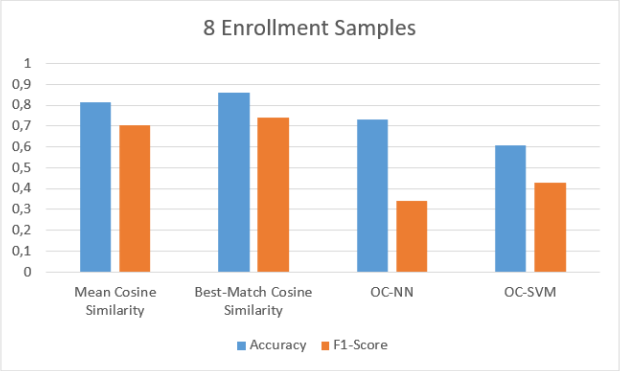

Figure 3: Testing results on TD dataset - 8 enrollment samples.

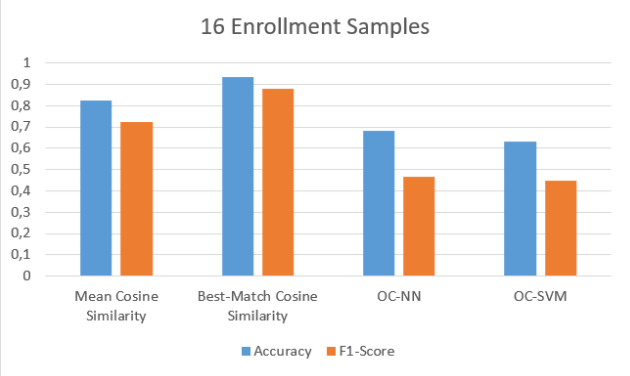

Figure 4: Testing results on TD dataset - 16 enrollment samples.

OCNN and OC-SVM see an improvement when the number of enrollment samples is increased, but mean cosine-similarity based methods tend to a stabilization in their accuracy the more enrollment samples are provided. The best performing method has been the one based on Best-Match Cosine Similarity, that with 16 enrollment samples obtained a mean F1-score among speakers equal to **0.878**, scoring an accuracy higher than **93**%. Performance on text-independent dataset is significantly worse, with the peak performance being obtained by the Mean Cosine Similarity method with 8 enrollment samples, reporting a F1-Score equal to 0.4725.

4.3.1 Improvement on the state-of-the-art

Cosine similarity based methods outperformed OCNN and OC-SVM approaches. Our proposed Best-Match Cosine Similarity algorithm obtained the best result in the text-dependent context. EER and AUC metrics for cosine similarity methods have been confronted, and plots are reported in figure 5:

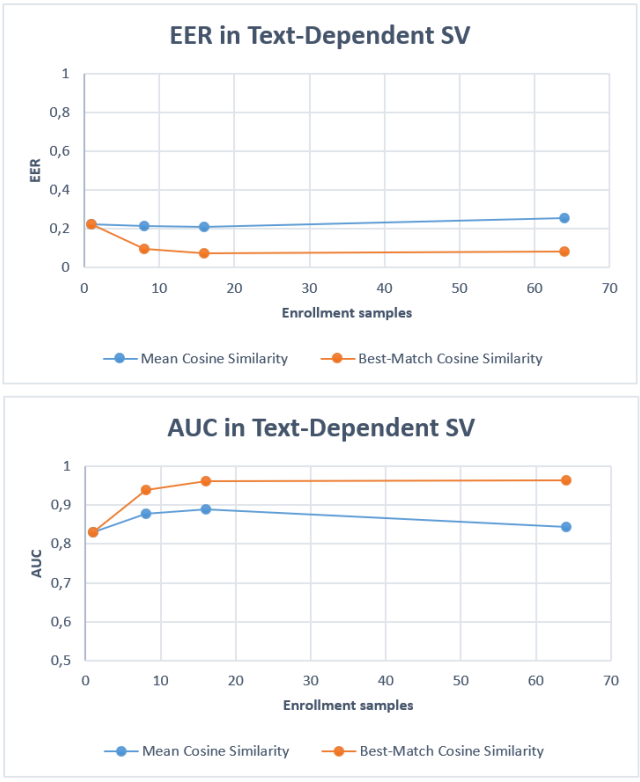

Figure 5: Confrontation of EER and AUC metrics for cosine similarity methods.

In general, preferring the Best-Match cosine similarity method instead of Mean Cosine Similarity allows to obtain a mean improvement of 14.25% in EER and 8.5% in AUC:

Enrollm. samples	EER improv.	AUC improv.
8	0,121	0,0625
16	0,1365	0,0725
64	0,17	0,12

Table 4: Improvements in EER and AUC if best-match cosine similarity is used.

5. On-device implementation

The application as described in section 2 was developed on the Infineon PSoC 62S2 Wi-Fi BT Pioneer Board and allows enrolling an authenticated speaker on-device and subsequently performing verification all through voice interaction, thanks to a cascaded execution of KWS and SV systems. The speaker verification algorithm implemented on the application is the Best-Match Cosine Similarity, chosen for its low computational footprint and great performance. The final application has the following memory requirements in its floating point version:

Final demo requirements	
Flash	354.32 kB
RAM	504.12 kB

Table 5: Final demo memory requirements.

With neural network quantization at 8-bit, flash memory can be reduced to 196.38 kB and RAM to 247.68 kB.

6. Conclusions

This work described an organic approach for combining two fundamental speech processing tasks in Tiny-ML, namely keyword spotting and speaker verification. Despite the rising need of having low-power on-device speaker verification systems, a top-down analysis of the design process for a completely Tiny-ML oriented speaker verification system was still missing in literature. We framed the Tiny-ML speaker verification task in the novel one-class few-shot classification context, highlighting constraints and limitations that have to be faced when developing such a system and analyzing ways for overcoming them. Moreover, a novel similarity system for performing speaker verification has been proposed, in some cases outperforming other state-of-the-art approaches. The KWS demos developed in this Master Thesis have been deemed commercially viable by the company that supported this thesis work, and the SV demo is used as proof-of-concept to showcase that it is possible to perform such tasks with an ultra low-power computer.

References

[1] Poojan Oza and Vishal M. Patel. One-class convolutional neural network. *IEEE Signal Processing Letters*, 26(2):277–281, feb 2019.

[2] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, apr 2015.

[3] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2014.

[4] P. Warden and D. Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-low-power Microcontrollers*. O'Reilly, 2020.

[5] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. 4 2018.