4. Deployment of KWS and SV Neural Network

1. View the source development TinySV: <https://dl.acm.org/doi/full/10.1145/3703412.3703415>

Github repository code: <https://github.com/AI-Tech-Research-Lab/TinySV>

I saw in-depth and this project implemented the MFCC generation and D-Vector extraction, so I will only take from this project the extraction for test purposes.

Approach used is multi-model, but in our case will MFE. The author of the paper tries to compromise Keyword Spotting and Adaptive Speaker Verification Module. Many applications that require on-device adaptation capabilities are consequently still not viable in TinyML. Speaker Verification is used in recognizing the identity of a user by analyzing audio captions provided by user as a reference and comparing them to newly connect audio data. TinySV, is a task specifically tailored to on-device learning context and introduce TinyML algorithm supporting SV Applications. There are two models:

* Instance-based: Solution rely on a Convolutional Neural Network to perform feature extraction and dimensionality reduction on input data. The advantages of this approach relies in training with a dimesionality-reduced version of data and provide acceptable results even with a small amount of data available
* Model-based: Optimized version of backpropagation for the adaptation of NN directly on-device. This approach is limited in their ability to learn complex patterns and exploit batches of data to avoid overfitting. Is important to emphasize that all the model-based assume a large availability of labeled data to perform training

Speaker Verification can be of two types:

* Text-dependent approach consisting in the user who is expected to pronounce a pre-determined word to be recognized
* Text-independent approach is expected to recognize the enrolled user independently from what they are saying

Available solutions for SV are for example:  
- Gaussian Mixture Model Universal Background Models (GMM-UBM)  
- Gaussian Mixture Model Support Vector Machines (GMM-SVM)  
- Joint Factor Analysis (JFA)  
- i-vector  
Method proposed by paper relies on NN able to extract voice-dependent low-dimensional vector, d-vector, from input speech that can be used by an instance-based solution for recognizing the identities of the speakers. However, the proposed approaches are not suitable for TinyML devices.  
  
The deployment of a model requires that SV algorithm:

* Must be adapted directly on device, meaning that a new user should be able to enroll in SV application by providing examples of their voice directly through the target device
* The algorithm must operate in a one-class manner, it should be able to learn to distinguish between the enrolled user and any other users, only from data coming from the enrolled one
* System requires a small amount of memory and computation during both inference and learning phase

Idea is that given a pre-defined keyword k, the task of TinySV algorithm is to assign a label

to the most recent segment of the stream I:

Immagine che contiene testo, Carattere, bianco, linea

Descrizione generata automaticamente

SE is enrolled user, instead SNE is any other speaker.  
The solution uses 2 modules: the first one for Keyword Spotting and the second for Adaptive Speaker Verification.  
Keyword spotting try finding a word k for each frame of a sample, if the word doesn’t correspond to activation word, then the result will be 0, if 1 the system try to identify if it is the user owner via ASV module.

Immagine che contiene diagramma, testo, schermata, linea

Descrizione generata automaticamente

Keyword spotting was already implemented in the previous phase, so now I focus on ASV module.  
Composed by 2 components:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, numero, Carattere, schermata

Descrizione generata automaticamente

* D-vector extractor: Generated spectrograms Pt are used as inputs for convolutional neural network. Following a transfer learning approach is developed by training a neural network to perform a speaker classification task in a supervised manner and then removing its final classification layer.

𝜔Φ𝑓= ∑︁𝜔𝑙, for each 𝑙∈Φ𝑓 is made a sum  
𝛼Φ𝑓= ∑︁ 𝛼𝑙 for each 𝑙∈Φ𝑓 is made a sum  
being 𝜔𝑙 and 𝛼𝑙 the number of weights and activations. for l layers.

* Instance-based model that operates in two different phases:

Immagine che contiene testo, diagramma, schermata, Piano

Descrizione generata automaticamente

* + Learning phase: Training phase of algorithm consists just in collection of a pre-determined number n of enrollment D-vectors De, collected from enrolled Speaker Se. This set is Δ𝐸 ={𝐷1𝐸,...,𝐷𝑖𝐸...,𝐷𝑛𝐸}
  + Inference phase: Cosine similarity between the newly collected D-vecotr Dt extracted from the extractor and all other vectors in Δ𝐸 and best-match cosine similary. The cosine similary is equal to:

𝜎(𝐷𝑡,Δ𝐸)= max(𝐷𝑡·𝐷𝑖/||𝐷𝑡||·||𝐷𝑖||)  
{𝐷𝑖∈Δ𝐸}

The result of the inference phase depends on a particular threshold taht will determine the user recognition.  
Technically, KS module acts as a filter, almost halving the amount of computation that would be performed at each inference cycle if the two algorithms were being executed in parallel.  
  
Proposed algorithm implementation:

Immagine che contiene testo, ricevuta, Carattere

Descrizione generata automaticamente  
The memory requirements for each component, like the intermediate computations I, P, D and models k, f, c and can be estimated with formulas and we highlight that this estimation is system-agnostic. We emphasize that memory of all components can be computed as product of number of parameters required by component. For estimation b=4B for all components, except for I, which is store 2B precision.

Immagine che contiene testo, Carattere, numero, schermata

Descrizione generata automaticamente

The size of the neural network with this implementation and combination of the models is:

Immagine che contiene testo, Carattere, numero, schermata

Descrizione generata automaticamente

This is a problem, because we know that the Neural Network can host only 589k parameters of 4-bit, so the total memory is 294,5 kB, so this model seems to not fit. Instead if we separate them we have for keyword spotting 200kB, instead for speaking verification is about 200kB. This probably indicates that for Syntiant NDP101 is recommended to use 2 different ones.

The environment on which these observations were done are on hardware Infineon PSoC 62S2 Wi-Fi BT Pioneer Board, which is programmable embedded system-on-chip, integrating a 150-MHz ARM Cortx-M4 as primary application process and a 100-MHz Arm Cortex-M’+ that supports low-power applications, up to 2MB Flash and 1MB SRAM. Implementation details were done using W of 1 s and a frequency of 16 kHz. Ach I is a 16000-element long vector. Considering that the windows are partly overlapped that corresponds to 0.75 s, computed as W-inference time of KWS.

1. What is the difference between a I-vector and a d-vector?

The first one is a feature that represents the characteristics of frame-level features distributive pattern. The extraction is a dimensionality reduction of GMM supervector. It’s extracted per sentence

The d-vector instead is extracted via a DNN model that takes stacked filterbank features and generates the one-hot speaker label on the output is trained. D-vector is averaged activation from the last hidden layer of DNN. This doesn’t have any assumption about feature’s distribution, instead i-vector assumes as default a Gaussian distribution.

1. Creation of a custom neural network focusing on KWS and SV:
   1. Creating a custom neural network focusing on KWS
   2. Creating a custom neural network combining KWS with a text-dependent SV model