This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

EasyTV descriptive narratives and object-based sound engine implementations

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 33 months

Document. ref.: D2.3

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	**Short name**	**Country**
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES
9	Unione italiana dei ciechi e degli ipovedenti	UICI	IT

PROGRAMME NAME:	**H2020. ICT-19-2017 Media and Content Convergence – IA Innovation Action**
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	UPM
INVOLVED UNITS:	CERTH
DOCUMENT NUMBER:	D2.3
DOCUMENT TITLE:	EasyTV descriptive narratives and object-based sound engine implementations
WORK-PACKAGE:	WP 2
DELIVERABLE TYPE:	Prototype
CONTRACTUAL DATE OF DELIVERY:	31-01-2020
LAST UPDATE:	
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
v. 01	28/11/2019	Draft	David Martín (UPM)	Table of Contents definition and document structure
V.02	07/12/2019	Draft	David Martín (UPM)	Add contributions to Speaker Diarization section.
V.03	15/01/2020	First version	David Martín (UPM)	First solid version of the document.
v.04	16/01/2020	First version	David Martín (UPM), Francisco Moreno (UPM)	Add contributions regarding the semi-automatic sound equalizer system
v.05	22/01/2020	First Review	Nicolamaria Manes (MV)	Reading and minor review
v.06	24/01/2020	Update Review	David Martín Gutiérrez (UPM)	Update the corresponding suggestions and revisions.
v.07	29/01/2020	Review	Giuseppe Vitolo (ENG)	Review
v.08	29/01/2020	Update review suggestions	David Martín (UPM)	Update manuscript with the corresponding comments

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
AI	Artificial Intelligence
SDS	Speaker Diarization System
VAD	Voice Activity Detection
ESS	Environment Sound System
HDFS	Hadoop Distributed File System
CNN	Convolutional Neural Network
DL	Deep Learning
ML	Machine Learning
DB	Database
SSE	Semi-automatic Sound Equalizer

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable is an official document of the EasyTV project funded by the European Union's Horizon 2020 (H2020) research and innovation programme under grant agreement No 761999. It is a report that describes the set of services implemented in the EasyTV platform that are related to audio processing solutions.

This deliverable is named as "EasyTV descriptive narratives and object-based sound engine implementations" and it is included within the framework of WP2.

The document is divided in six main sections: firstly, a brief introduction to the problems that are going to be investigated regarding audio processing tools will be presented.

Section two is devoted to describing and analysing the technology and procedures that were carried out to build an Environment Sound System that will incorporate additional information to the subtitles including a collection of impulsive sounds gathered from a public dataset.

Section three is related to the design and implementation of a Speaker Diarization System based on Deep Learning approaches that will be employed to incorporate information about the characters of all the media content available in the platform. Moreover, this system will fusion its output with the face detector system in order to promote the result when detecting who is speaking.

Section four is devoted to the description of the breakthroughs in the design of a semi-automatic sound equalizer that will be employed to personalize a fixed set of frequency bands based on the profile of the end-user and the media content. This profiling process can be managed by using an audiometry test on the frequency bands that are available.

Section five describes the different end-points that can be used throughout the EasyTV platform in order to access to the aforementioned services.

Finally, the last section summarizes all the conclusions and provides a general overview about future lines and work that will be included in new versions of the aforementioned services to promote the current results.

1. INTRODUCTION

In recent years, the development of multimedia applications has grown exponentially due to the digital transformation that is carried out by many media producers and broadcasters. Moreover, the incorporation of personalized applications to improve the accessibility of media content to people with visual and/or hearing disabilities is also considered as an important challenge to bear in mind when building new services.

Within the accessibility framework, EasyTV seeks to find an integral solution that may help these people by providing them with additional information obtained by employing technologies such as character recognition, sound event recognition, semi-automatic sound equalizer among others.

Furthermore, Artificial Intelligence (AI) is gaining more attention when implementing automatic services in order to improve the quality of life of people with certain disorder or disabilities by supporting them with intuitive tools that may help them to be more independent.

Along this document, a set of audio processing services based on AI will be presented, including the technology that is used as well as the methodology and metrics that were employed to conduct the experiments. Moreover, the document reports the integration of these services in the EasyTV application.

More specifically, the document explains three main services based on audio processing:

a) An Environment Sound System (ESS) that receives and audio recording and yields a collection of environment sound events (if any).
b) A Speaker Diarization System (SDS) which attempts to detect who is the speaker in a certain moment of an audio recording.
c) A Semi-automatic Sound Equalizer (SSE) which seeks to personalize the different frequency bands based on the profile of the end-user and the media content that is consuming.

2. ENVIRONMENTAL SOUND SYSTEM

The scope of this section is to introduce and to describe the methods and experiments conducted to build a system capable of recognizing impulsive sounds such as glass breaks, human screams or door slams. The idea is to provide end-users with this information via subtitles. By using this system, we are improving the accessibility of media content by including additional information to the content subtitles.

The outline of this section is the following: first, a brief introduction to the problem of sound recognition will be presented. Then, the datasets and additional information that were used for conducting the experiments will be described. The next subsection is devoted to presenting the experiments and results of the implementations and finally the integration and the development of the service will be summarized in the last part.

2.1. Background & Related Work

Environmental sound recognition or classification is a growing and opening topic in audio processing due to its huge impact in multiple multimedia applications such as audio surveillance systems, audio scenes classification or media content generation as (Sailor, 2017) suggests.

Previous studies were based on *handcrafted* features including log-Mel features, dictionary learning, wavelet-based features or the so-called cepstral coefficients (MFCC). However, deep learning approaches outperform these previous procedures by using Convolutional Neural Networks (CNN), since a CNN classifier is capable of capturing the energy across the time as it is described in (Salamon, 2017) and (Sailor, 2017).

In this project a CNN model has been used to detect different impulsive or environment sounds that

are included in the Environment Sound Classification (ESC) Dataset (Piczak, 2015). The limitation of using this dataset is that the system can only detect those sounds included in the dataset so that, whether the final application requires different or additional sounds, a large set of sample for these sounds must be included in the training process of the model.

2.2. Environmental Sound Datasets

As it was mentioned before, the dataset that is employed to train and validate the model is the ESC-50 dataset which contains 50 different sounds that are summarized in the following table:

Animals	Natural soundscapes & water sounds	Human, non-speech sounds	Interior/domestic sounds	Exterior/urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

Table 1 Set of sounds available in the ESC-50 Dataset

As one may see, there exist five potential categories: animals, natural soundscapes, human sounds domestic sounds and urban sounds. The total number of audio recordings is 2000 and each sample consists of 5-seconds-long audio recordings. To enrich the dataset, a data augmentation strategy (Cubuk, 2019) has also been employed.

2.3. System Implementation

Along this section, the different steps that are needed to build the Environment Sound System (ESS) are described. Two main phases are considered, including audio pre-processing and both training and validation procedures to evaluate the performance of the system.

The former phase basically consists of representing an audio signal in a more appropriate way using different transformations such as the Short-Fourier transform. To do so, a collection of parameters needs to be defined. The latter phase is a classical approach in Deep Learning (DL) procedures, where the first step is the design of the architecture and the parameters involved in it. Then, the

model needs to be trained and validated using different sets of data and a cross-validation strategy to avoid suffering from the so-called overfitting. In particular, this problem is related to the capability of DL models to generalize well when new data is introduced into them.

The main limitation of these CNN models lies in the necessity of collecting a large number of samples per category with sufficient quality to avoid problems when predicting new samples. Moreover, whether new categories are desired to be detected, the model must be re-trained using the new information, otherwise it will not work as expected.

Figure 1 represents the architecture of the ESS implementation. As it was already mentioned, the system has a set of main phases: audio pre-processing to extract the Mel spectrograms, audio segmentation to synthesize the audio signal into small segments, and then data modelling via CNN models in order to extract relevant features and retrieve the final prediction. Recall that there is a slight modification in the pipeline:

- When training the model, the audio segmentation part is not required since the data is already separated in 5-seconds audio recordings, so this phase is skipped.

- When testing the model using new samples, all the aforementioned phases are included since the audio recording may have different duration and therefore, a window analysis is necessary to split the recording into multiple audio sub-segments. Then, a Mel spectrogram transformation is performed per audio sub-segment. Finally, each audio sub-segment is analysed by the CNN model which will yield a detected sound per sub-segment.

The arrows marked in blue colour correspond to processes that are mandatory during the training whereas the red colour arrows indicate procedures that are only done during the training (such as saving all the spectrograms in a DB for future trainings). Additionally, black arrows indicate that these stages are performed during both processes: training and testing.

Furthermore, sub-segments marked in blue indicate that they are potential candidates to be environmental sounds whereas the red ones may correspond to speech segments or non-activity events. This decision is made by a Voice Activity Detector module that will be explained in the next section.

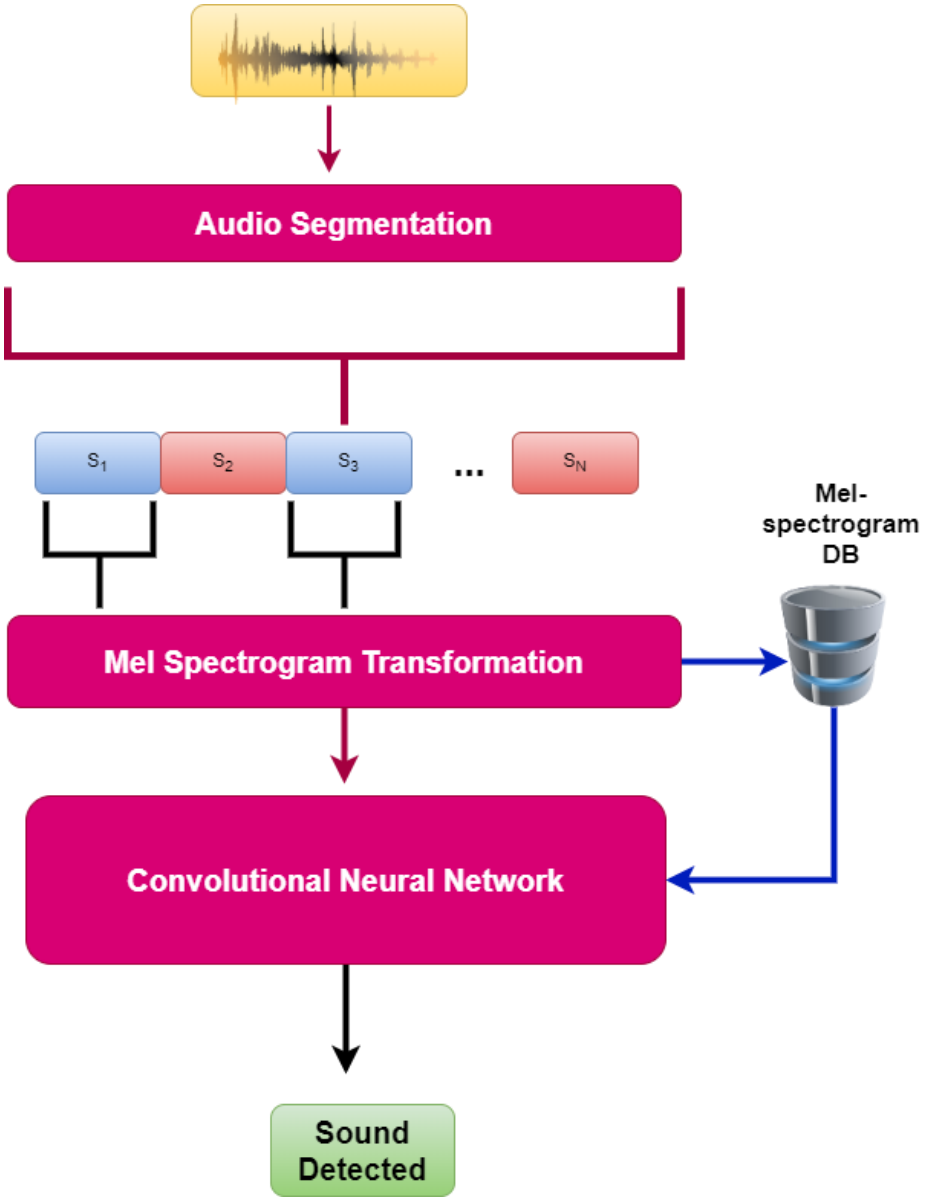

Figure 1 ESS block diagram

2.3.1. Audio Preprocessing

Many recent applications and investigations in audio processing are based on well-known image processing techniques. The reason behind this lies in the use of Deep Leaning (DL) approaches to efficiently obtain relevant audio features that may promote the results of a classifier. It was demonstrated in many studies that DL models outperforms previous statistical modelling and Machine Learning (ML) approaches when performing object recognition or any other topic within image processing.

To address the problem from an audio processing perspective, a previous transformation of the audio signal is required in order to obtain a 2D representation of the raw audio signal: the well-known Mel spectrogram.

The most important parameters to be considered are: the sampling rate, the analysis window length which depends on the sampling rate and the number of seconds of the analysis, the Hop Length which is the overlapped length of the window measured in samples, the number of Discrete Fourier

Transform (DFT) points, the number of bands.

According to the problem that needs to be solved, the sampling rate is the standard 22500Hz, the minimum frequency of the analysis is 50Hz whereas the maximum frequency is provided by the Nyquist criteria, so that, 11250 Hz (half of the sampling rate).

Moreover, the number of bands is 128, the duration of the analysis is fixed to 5 seconds, the number of DFT points is 2048 and the Hop length is fixed to 512 (number of samples between successive frames). The following figure shows examples of two different sound events in order to remark the differences that one may observe when representing both signals as spectrograms.

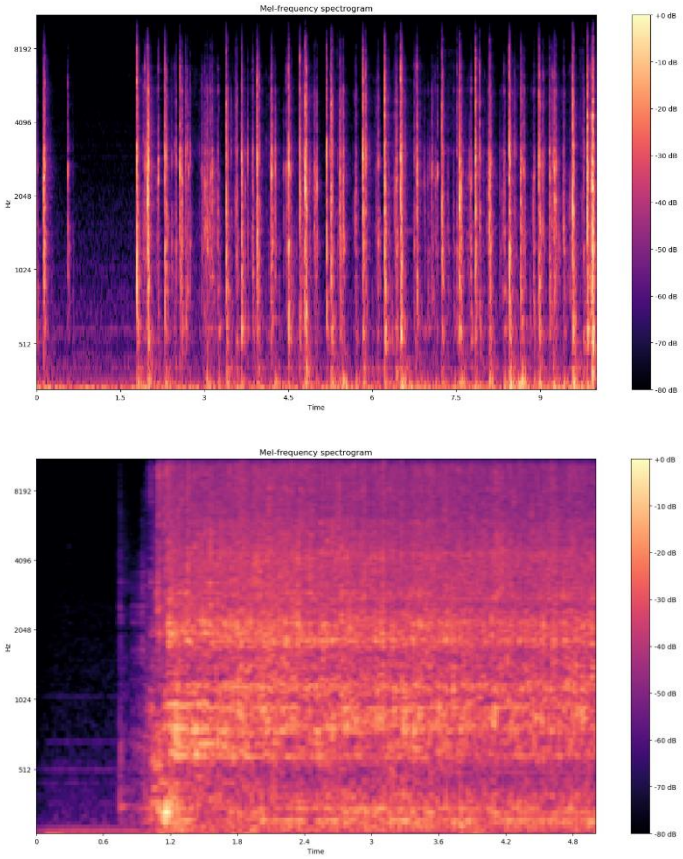

Figure 2 Examples of Mel spectrograms from Keyboard typing and toilet flush sounds

Therefore, for each audio recording on the dataset, a Mel spectrogram with the aforementioned parameters is calculated and stored in a new database (DB).

Moreover, the original signal has been modified using different transformations such as white noise addition, or pitch modification in order to collect more samples from the same audio recording. This technique is named as Data Augmentation and it is widely used in multiple DL applications where the dataset is not enough large to train a complex model such as CNN's models.

2.3.2. **Convolutional Neural Network Model**

When all the spectrograms are stored in the new database, the CNN model can be designed. More specifically, the input of the model is a 4D vector with dimensions *(N x m x L x 1)*. In this case, N corresponds to the total number of samples, *m* indicates the number of bands (128), and *L* the number of audio frames. Finally, the last dimension indicates the number of channels which is 1 since the audio recordings were converted from stereo to mono signals.

In order to obtain the number of audio frames, the following operation is performed:

$$L = \frac{t_s sr}{h}$$

Where t_s represents the number of seconds employed in the analysis, *sr* is the sample rate and *h* corresponds to the Hop length. Thus, the number of audio frames in the analysis is 220.

As it was introduced in previous sections, the model that was designed to solve this problem is a general CNN model whose architecture is presented below. The implementation of the model has been carried out using Python, Keras (Keras, n.d.) and Tensorflow (Tensorflow, n.d.).

Model: "ESCNet"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 220, 64)	640
activation_1 (Activation)	(None, 128, 220, 64)	0
conv2d_2 (Conv2D)	(None, 128, 220, 64)	36928
activation_2 (Activation)	(None, 128, 220, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 64, 110, 64)	0
dropout_1 (Dropout)	(None, 64, 110, 64)	0
conv2d_3 (Conv2D)	(None, 64, 110, 128)	73856
activation_3 (Activation)	(None, 64, 110, 128)	0
conv2d_4 (Conv2D)	(None, 64, 110, 128)	147584
activation_4 (Activation)	(None, 64, 110, 128)	0
max_pooling2d_2 (MaxPooling2	(None, 32, 55, 128)	0
dropout_2 (Dropout)	(None, 32, 55, 128)	0
conv2d_5 (Conv2D)	(None, 32, 55, 256)	295168
activation_5 (Activation)	(None, 32, 55, 256)	0
conv2d_6 (Conv2D)	(None, 32, 55, 256)	590080

```
_________________________________________________________________
activation_6 (Activation)    (None, 32, 55, 256)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 28, 256)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 16, 28, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 114688)            0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              117441536
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 50)                25650
=================================================================
Total params: 119,136,242
Trainable params: 119,136,242
Non-trainable params: 0
_________________________________________________________________
```

As expected, the input of the model must have the aforementioned dimensions whereas the last layer of the model is equal to 50 since the dataset contains such number of categories. Moreover, the loss function that is minimized during the training process is the categorial cross-entropy loss and the optimizer selected was the Adam method.

2.4. Experiments and Results

To conduct the experiments, the ESC50 Dataset was divided in different subsets according to the documentation provided by the authors. More specifically, they provide a pre-defined subsets of samples to perform a Cross-validation procedure (Fu, 2019) using k-fold strategy with k=10.

Then, all the spectrograms are computed for each 5-seconds audio recording and stored in a Hadoop Distributed File System (HDFS) file.

The next step consists in training the model throughout the Cross-validation strategy and evaluate the performance at each iteration in terms of the following metrics: accuracy, precision, recall and f1-score. These set of metrics are the classical ones for evaluating a classification system. The loss function (categorical cross-entropy loss) can be also be useful when measuring the performance of the system.

In the following tables, the first results using the aforementioned model are presented. As expected, some classes or categories are better predicted by the model than others. This is a normal behaviour in DL models and the solution to address this issue lies in gathering more data with high-quality from these categories that are worse predicted.

Category	Accuracy	Precision	Recall
Siren	98.56	98.95	97.96
DoorKnock	98.02	98.87	97.56
Clapping	96.85	97.62	96.36
Helicopter	96.23	97.05	95.12

Table 2 Classification metrics for a set of categories that are very well predicted by the CNN model

Category	Accuracy	Precision	Recall
VacuumCleaner	75.26	76.32	75.10
Dog	75.29	76.55	74.94
Train	62.50	63.41	62.08
CarHorn	61.97	62.53	60.86

Table 3 Classification metrics for a set of categories that were considerable well-predicted by the CNN model

Category	Accuracy	Precision	Recall
GlassBreaking	49.56	50.14	49.73
Coughing	48.75	49.69	48.63
Thunderstorm	47.52	48.94	47.29
ToiletFlush	47.25	47.32	46.11

Table 4 Classification metrics of a set of categories that are not predicted with high precision by the CNN model

Once the model was trained and validated, it was tested using real data from the available media content in the EasyTV platform. However, the results were not as expected since the data that was introduced into the network were very poor in terms of quality and noise.

On the one hand, in some cases there were multiple events in the same audio sub-segment so that, the model is not capable of separating the audio signal to retrieve only one of its components. On the other hand, many false positive raised due to the lack of a sound event detector that will only allow audio segments which contain possible events to be introduced into the network. In this experiments, instead of using a Sound Event Detector (SED), we have employed a Voice Activity Detector (VAD) which separate the audio recording into speech and non-speech segments. This approach is not as solid as the one that employs a SED module before predicting the audio sub-segment.

Furthermore, a sound annotation and verification tool was implemented in order to annotate specific data corresponding to a fix set of environmental sound events instead of using the data from the public datasets. By doing so, we mitigate the aforementioned problems since we will employ directly data from the media content which is more realistic to train the final model that will be deployed in the production environment.

2.5. Service Implementation

The ESS has been totally implemented in Python using *Flask* (Flask, s.f.) in order to deploy the different phases of the system as a service. The code to deploy this service as an standalone project can be found at https://gitlab.com/davidGATV/environmental-sound-classification-50 by firstly requesting access to the repository.

Moreover, all the implementations are included in the EasyTV platform and the documentation with both data models and end-points that are needed to use the service can be found in the general Swagger documentation of the platform.

Basically, there is one main functionality that can be performed using the platform:

a) Predict a new audio recording using the VAD model to detect non-speech segments and then the CNN model yields the sounds detected for each non-speech segment.

The rest of the functionalities including dataset generation and training has not been included in the current version of the platform but will be incorporated once the new custom dataset, built by the sound annotation tool, is available.

3. SPEAKER DIARIZATION SYSTEM

Along this section, a complete description of the development regarding the Speaker Diarization System will be presented. Firstly, a brief introduction to the current state of art and the problem itself are exposed. Then, the different datasets that have been used to conduct the experiments are described to understand the kind of data employed in the proposed system. This section will end by analysing the limitations of using Public Datasets in the final application of the project and the solution and challenges that were considered to address such issues.

The next sub-section is devoted to describing the different experiments that were carried out when implementing the system. Finally, the last part of this section is related to the implementation of the system within the EASYTV platform.

After analysing the state of the art and the problem itself regarding EASYTV, it was decided to follow a strategy of supervised learning models which receives raw audio signals and extracts a set of features based on such signals that lead them to determine the class or speaker associated to such input.

3.1. Background & Related Work

Speaker Diarization systems (SDS) have been widely studied and investigated to solve the problem of “who spoke when”. In particular, most of the existing studies are based on three main stages as authors remarked in (Zhang, 2019):

1) A *Speech Segmentation* module which attempts to separate the speech parts from the non-speech ones.
2) A *Feature Extractor* module to collect representation information from the different speakers as a set of vectors or embeddings.
3) A *Clustering module* to extract the number of clusters and to classify the aforementioned embeddings based on distance metrics.
4) A *Re-segmentation* module that seeks to reinforce the segmentation of the speech parts based on the clustering results.

Many authors have proposed procedures to solve the Feature Extractor module using neural networks to promote the acquisition of the speaker embeddings such as (Garcia-Romero, 2017), (Wang, 2018), (Zajíc, 2017). Moreover, regarding the clustering phase, most of the well-known methods are based on unsupervised techniques such as Gaussian Mixtures, spectral clustering or agglomerative hierarchical clustering. However, authors in (Zhang, 2019) suggests a supervised approached that outperforms previous work in this area.

3.2. Speaker Diarization Datasets

To perform the experiments, the well-known *LibriSpeech ASR corpus* (Open SLR, n.d.) is used. The library contains several parts to train and validate the proposed models. More specifically, the dataset contains a large amount of corpus of read English speech. Therefore, the segmentation phase is not necessary to train the models due to the fact that each utterance contains only speech regarding one specific speaker.

The main limitation of using this or any other Public Dataset lies in the lack of noisy signals and environment distortions of the sound because the data is very clear so that the model may suffer from overfitting and therefore, it might not generalize as well as expected in other scenarios where noise and/or background sounds may provoke occlusions in the speaker signal.

To address this limitation, a sound annotation tool was proposed to annotate the information of some media content that will be used within the EASYTV platform. By doing this, the model will have a considerable set of more realistic data to mitigate the amount of errors when predicting noisy signals.

3.3. System Implementation

In Figure 3, a description of the proposed system to solve the Speaker Diarization problem is presented.

As it was introduced in previous sections, the first phase consists in extracting speech and non-speech segments from the audio raw signal. To do so, a Voice Activity Detection (VAD) module is needed. Then, all the segments are pre-processed, and they pass through a supervised Deep Learning (DL) model based on Convolutional Neural Networks (CNN). This network is used to perform the feature extraction phase where the speaker embeddings are gathered. Finally, an algorithm is used to match and associated the embedding to a specific speaker.

The system implementation is slightly different when training and predicting. When training the model, the VAD module is not required since the data is already separated in speech segments and thus, this module is omitted. On the other hand, in a real situation, a new audio signal is introduced into the system and the VAD module is employed to extract such segments that corresponds to speech information that needs to be both pre-processed and classified.

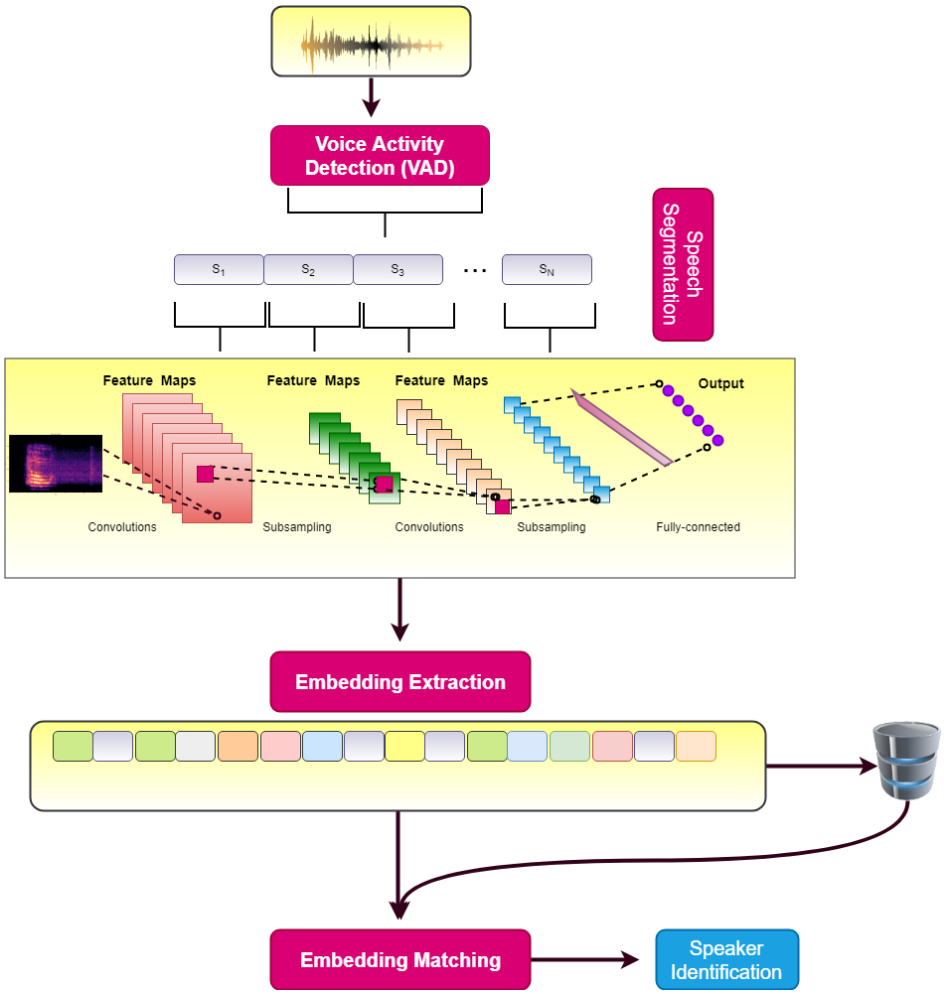

Figure 3 A block diagram regarding the general schema of the Speaker Diarization System

3.3.1. Audio Pre-processing

In this first stage, it is necessary to compute a more appropriate representation of the audio recording as it was already mentioned in previous sections, in order to employ these representations as inputs in the CNN model.

Once again, to address the problem from an audio processing perspective, a preliminary transformation of the audio signal is required in order to obtain a 2D representation of the raw audio signal: the Mel spectrogram.

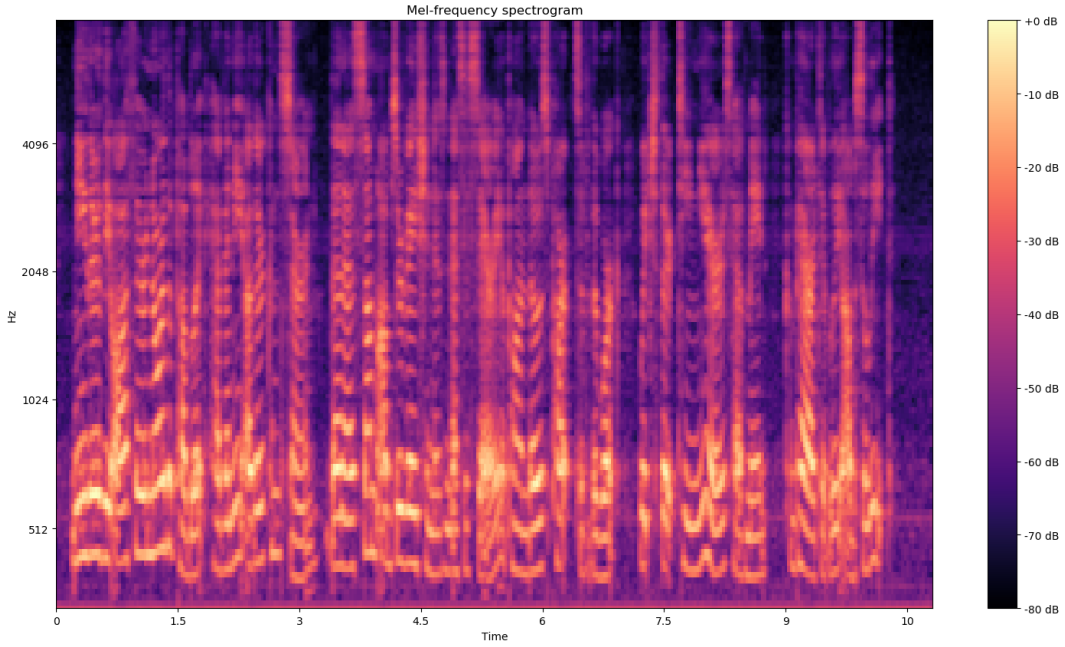

Figure 4 A representation of a Mel spectrogram from a speech signal.

In Figure 4, an example of a Mel spectrogram calculated from a speech audio signal is shown. As one may observe, the spectrogram represents the different frequencies, in a certain range, along the time.

The calculation of the spectrogram requires a set of parameters such as the sampling rate, the number of filters, the window employed, or the number of points needed in the Short Fourier transform among others.

Since the objective of the model is to recognize speakers, the range of the frequency must be bounded by the human frequency range which is the following:

- In the male voice, the frequency range lies between 100Hz to 8kHz approximately.
- In the female voice, the frequency range lies between 350Hz and 17kHz.

Therefore, the minimum frequency required for the analysis is 100Hz. The maximum frequency depends directly on the sampling rate. According to the Nyquist criteria as it is redefined in (Romanov, 2019), the maximum frequency must be as much half of the sampling frequency. Therefore, fixing a sampling of 32000Hz, the maximum frequency must be approximately of 16kHz.

3.3.2. **Voice Activity Detector**

The main goal of the VAD system consists in efficiently separate speech from non-speech audio segments. During these months, several state-of-art models have been implemented to achieve the best performance in this first stage of the SDS pipeline. There exist different families of approaches: ones related to statistical modelling and others based on DL approaches. During these months, both families were analysed and developed to compare the performance in different scenarios.

The first approach is based on changes in the energy within the human voice frequency band using median filters as in (Pang, 2017).

The second approach consists of a DL model that is trained to detect both gender and activity/non-activity events (Doukhan, 2018). The architecture is based on CNN layers to extract relevant features from the audio signal and fully connected layers to perform the classification task.

The third implemented approach is based on DL models as well but in this case, the output of the model is a binary decision regarding speech and non-speech using the Google WebRTC tool (Google, s.f.).

In Figure 5, a simple block diagram explaining the VAD module is depicted. The input of the module is the raw audio signal and the output consists of a set of annotated segments whose values will depend on the selected VAD model. In such image, the colour of the segments just represents whether the model detect it as speech (grey) or non-speech (red).

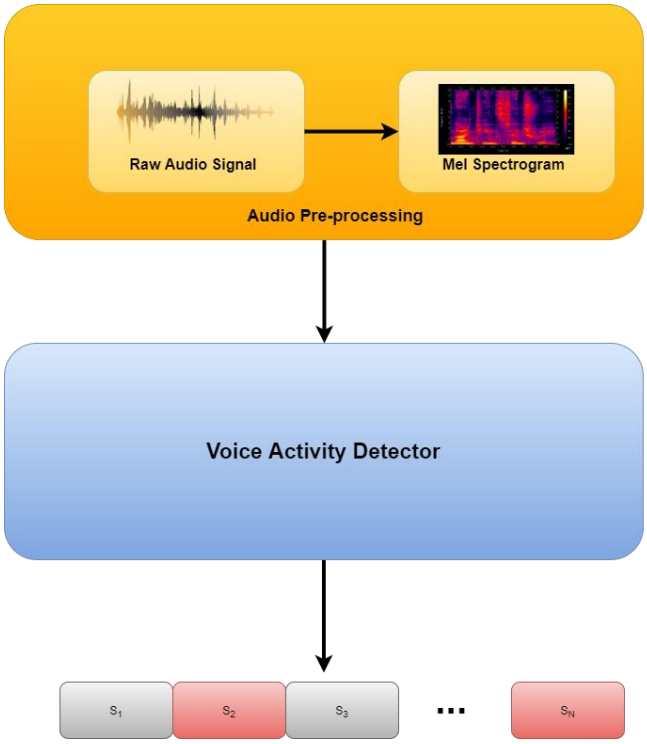

Figure 5 Simple Block Diagram for a Voice Activity Detector module

3.3.3. Convolutional Neural Network Model

The third phase of the system is the feature extraction stage, where a CNN is employed to train a model capable of distinguishing among the different speakers of the dataset based on the distribution of the annotated Mel spectrograms that are introduced in the network.

The architecture of the model is developed in *Keras* (Keras, n.d.) using *Tensorflow* (Tensorflow, n.d.) as backend and its composition is presented in the following table:

_____

Model: "speechFeatExtractor"

_____

Layer (type) Output Shape Param #

==========

conv2d_2 (Conv2D) (None, 32, 517, 64) 640

_____

activation_1 (Activation) (None, 32, 517, 64) 0 _____

conv2d_3 (Conv2D) (None, 32, 517, 64) 36928

_____

activation_2 (Activation) (None, 32, 517, 64) 0

_____

max_pooling2d_1 (MaxPooling2 (None, 16, 259, 64) 0

_____

dropout_1 (Dropout) (None, 16, 259, 64) 0

_____

conv2d_4 (Conv2D) (None, 16, 259, 128) 73856

_____

activation_3 (Activation) (None, 16, 259, 128) 0

_____

conv2d_5 (Conv2D) (None, 16, 259, 128) 147584

_____

activation_4 (Activation) (None, 16, 259, 128) 0

_____

max_pooling2d_2 (MaxPooling2 (None, 8, 130, 128) 0

_____

dropout_2 (Dropout) (None, 8, 130, 128) 0

_____

conv2d_6 (Conv2D) (None, 8, 130, 256) 295168

_____

activation_5 (Activation) (None, 8, 130, 256) 0

_____

conv2d_7 (Conv2D) (None, 8, 130, 256) 590080

_____

activation_6 (Activation) (None, 8, 130, 256) 0

_____

max_pooling2d_3 (MaxPooling2 (None, 4, 65, 256) 0

_____

dropout_3 (Dropout) (None, 4, 65, 256) 0

_____

flatten_1 (Flatten) (None, 66560) 0

_____

embedding (Dense) (None, 512) 34079232

==========

Total params: 35,223,488

Trainable params: 35,223,488

Non-trainable params: 0

_____

The crucial layer is the one named as “embedding” which consists in a 512 vector that will represent a (*n x m)* Mel spectrogram introduced into a one-dimensional vector of 512 values. This branch will continue with a typical set of fully-connected layers that provide the final classification.

Moreover, during the training process, the so-called categorical cross-entropy loss function is minimized using the Adam optimizer technique.

3.3.4. **Matching Speaker Embedding Algorithm**

In the previous section, an explanation of the feature extraction procedure to gather the speaker embeddings is presented. As a recall, after training the model the model can be used either predict the available speakers in the dataset or extract the speaker embeddings using its last layer.

All the embeddings for each audio segment are stored in an *embedding* database. The main issue when storing all the embeddings in the database lies in the fact that many of them may not be very representative and may lead to errors when predicting the final speaker. To address this problem, a clustering method is used to remove those embeddings associated to a certain speaker that may not represent it as expected.

The proposed algorithm for this purposed is named as Density-Based Clustering Based on Hierarchical Density Estimates (HDBSCAN) (McInnes, 2017) which basically is an improvement of the density-based, hierarchical clustering method proposed in (Campello, 2013). By using this approach for every set of embeddings associated to each speaker, the embedding dataset is reduced by only those embeddings that are detected to belong indeed to a specific speaker class.

Additionally, the centroids of each of the classes (speakers) are also stored for being used when predicting a new audio signal based on the remaining embedding.

Moreover, an algorithm based on a distance metric is needed as well in order to perform the final classification based on the centroid embeddings when a new audio signal is introduced into the system. When such signal feeds the mode, it generates its speaker embedding which must be associated to any of the speakers available in the dataset if and only if, the distance between such speaker embedding and the centroid embedding is lower than a particular threshold. If all the distances between pairs of embedding centroids and the new speaker embedding do not satisfy the criteria, then, the new audio signal will be assigned to “New Speaker”, otherwise it will be associated to the speaker whose centroid is closer to the speaker embedding.

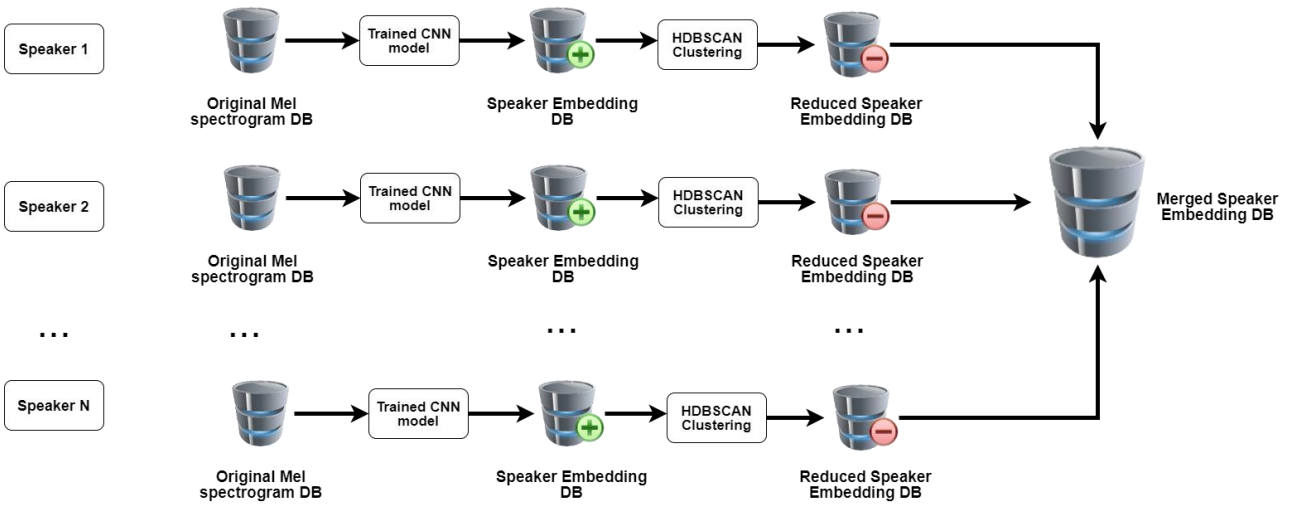

Figure 6 Block diagram regarding the speaker embedding computation.

In Figure 6, an illustration of the process involved in the calculation of the speaker embeddings is presented. As one may observe, once the process ends, the database contains the more

representative embedding for each speaker. Then, the centroid of a particular speaker is computed by taking the average of the set of embeddings associated to it.

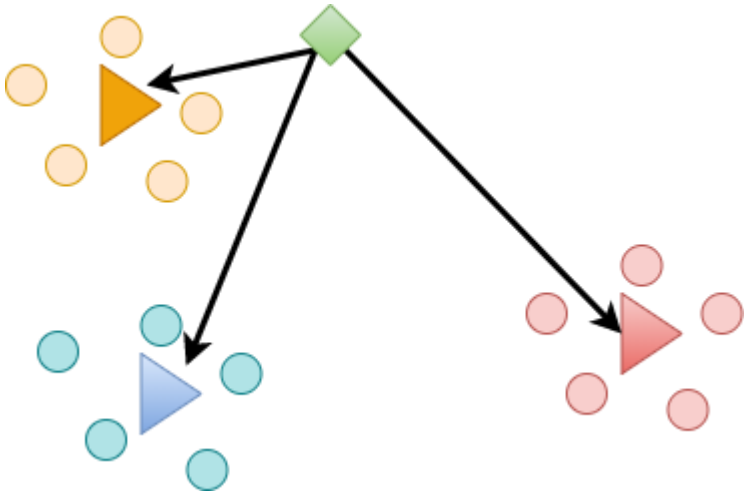

Figure 7 Illustration of the embedding matching algorithm based on the centroids of the embeddings

In Figure 7, an illustration of the process regarding the embedding matching is presented. In this case, a new audio signal as introduced into the system, the model generated its embedding, which is depicted in the figure as green diamond. Finally, the embedding is associated to the orange cluster (which represents a speaker) since the distance satisfy the criteria regarding the threshold. The distance used for this purpose is the cosine similarity which is 1 if the two vectors have the same orientation and 0 whether the two vectors are totally different. The threshold is a parameter to be found based on the data but and must be adjusted depending on the problem to be solved. In this case, a threshold between 0.8 and 0.9 was initially proposed.

3.4. Experiments and Results

To conduct the experiments, the dataset is split in two different sets: training (80% of the whole data) and test (remaining 20%) sets. The former is used to train the model using a k-Fold Cross-validation method (Fu, 2019) in order to validate the model with different subsets of the training set.

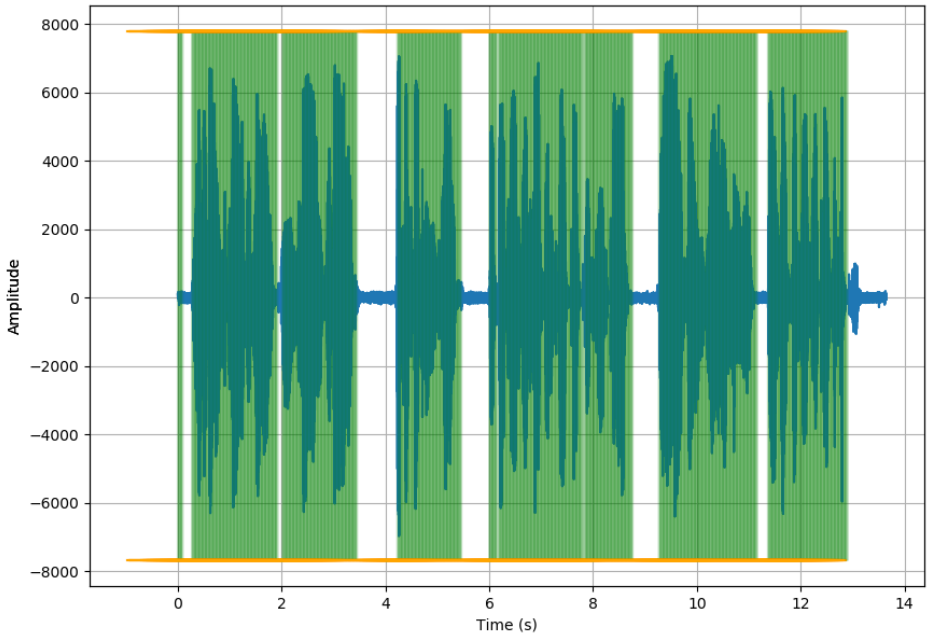

Figure 8 An Example of the output provided by the Voice Activity Detector

In the figure above, an illustrative example of the output provided by the VAD for a certain speech audio signal is presented. As one may observe, the VAD model is able to extract the sub-segments where there were speech fragments (vertical green lines).

In Table 5, the results regarding the training procedure of the model are presented. More specifically, the classical classification metrics such as accuracy, precision, recall and f1-score are reported. As the table shows, the results are very promising in terms of precision. Since the process is performed using 10-fold validation, the average of the values per iteration is calculated.

# of speakers	Accuracy (train)	Precision (train)	Recall (train)	F1-Score (train)	Accuracy (validation)	Precision (validation)	Recall (validation)	F1-score (validation)
7	96.38	96.97	95.85	96.40	98.82	99.06	98.45	98.76
20	86.84	89.72	84.58	83.96	84.05	87.68	80.74	84.06

Table 5 Classification results of the CNN model

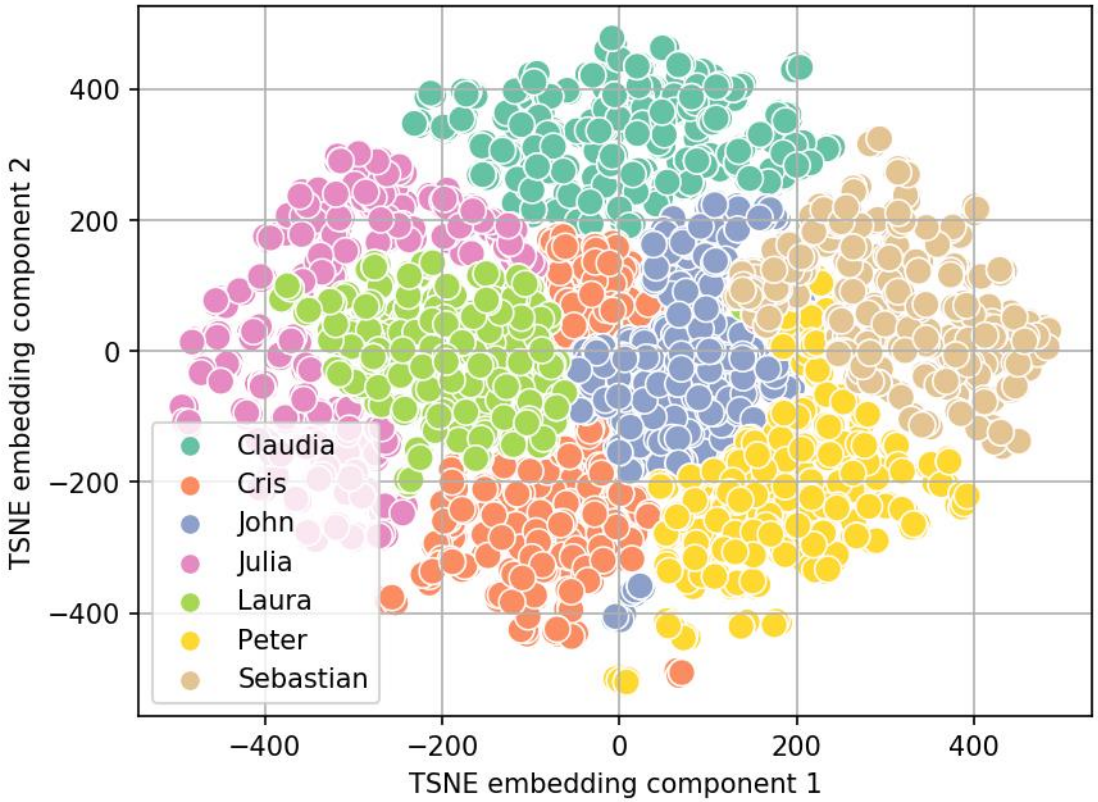

Figure 9 T-SNE representation of the embeddings of a subset of speakers

Once the model is properly trained, the speaker embeddings can be computed using the fully-connected layer of the model named as *embedding*. As a reminder, an embedding is a vector that represents the input data in a compressed way.

To visualize the results of these experiments, the so-called TSNE (Van Der Maaten, 2014) algorithm is performed. Basically, this algorithm projects the high-dimensional embedding (512-dimensions) into a 2D or 3D vector. Moreover, Figure 9 shows the result of the analysis using a subset of speakers (seven indeed). The speakers are anonymous in the dataset but during the experiments they were given names to be more intuitive when presenting the results.

Furthermore, the same model was used to be trained using specific data from some media content that will be used in the EASYTV platform. However, since such data was collected using the annotation tool and the amount of information per speaker was not enough to optimize all the parameters of the model, the results were poorer than in the case of using the public datasets. The main reason behind this is based on two main points:

- The amount of the data needed to train and validation in a proper way a CNN model must be large.
- The quality of the training and validation samples must be as good as possible to avoid errors when predicting new samples.

To mitigate this issue, more media content will be properly annotated by end-users in order to carried-out more experiments using CNN models.

3.5. Service Implementation

The SDS has been totally implemented in Python using *Flask* (Flask, s.f.) in order to deploy the different phases of the system as a service. The code to deploy this service as an standalone project can be found at https://gitlab.com/davidGATV/speaker-diarization by requesting access to the code.

Moreover, all the implementations are included in the EasyTV platform and the documentation with

both data models and end-points that are needed to use the service can be found in the general Swagger of the platform.

As in the previous audio service, there are similar functionalities that can be performed via EasyTV platform:

a) Generate a new dataset for a specific media content, which retrieves all the data available in the DB and extracts the Mel spectrograms needed for training the model.
b) Train the CNN model, which triggers the training procedure and starts reading the Mel spectrogram DB and performing the Cross-validation strategy.
c) Extract the set of speaker embeddings and their centroids using the last layer of the CNN model and stored them into a file.
d) Predict a new audio recording using the VAD model to detect non-speech segments and then the CNN model yields the sounds detected for each non-speech segment.

4. SEMI-AUTOMATIC SOUND EQUALIZER

The scope of this section consists in describing the first approach of a Semi-automatic Sound Equalizer (SSE) in order to improve the quality of experience of users that may suffer from hearing disabilities. Moreover, this solution can also be used to personalize the sound depending on the media content that is consumed.

4.1. Frequency bands selection

The first phase in this case is related to analyse which frequency bands are the most problematic ones and therefore the most interested to be considered in the solution. To bound the problem, five frequency bands have been initially considered: Lows, Low-Mids, Mids, High-Mids and Highs. As an example, the human voice will be between the Low-Mids and the Mids, whereas most of the energy of a keyboard tipping noise will be included in the Lows frequency band.

The following figure shows the five frequency bands that have been mentioned before where the x-axis represents the frequency in Hz and the y-axis represents the level in dB.

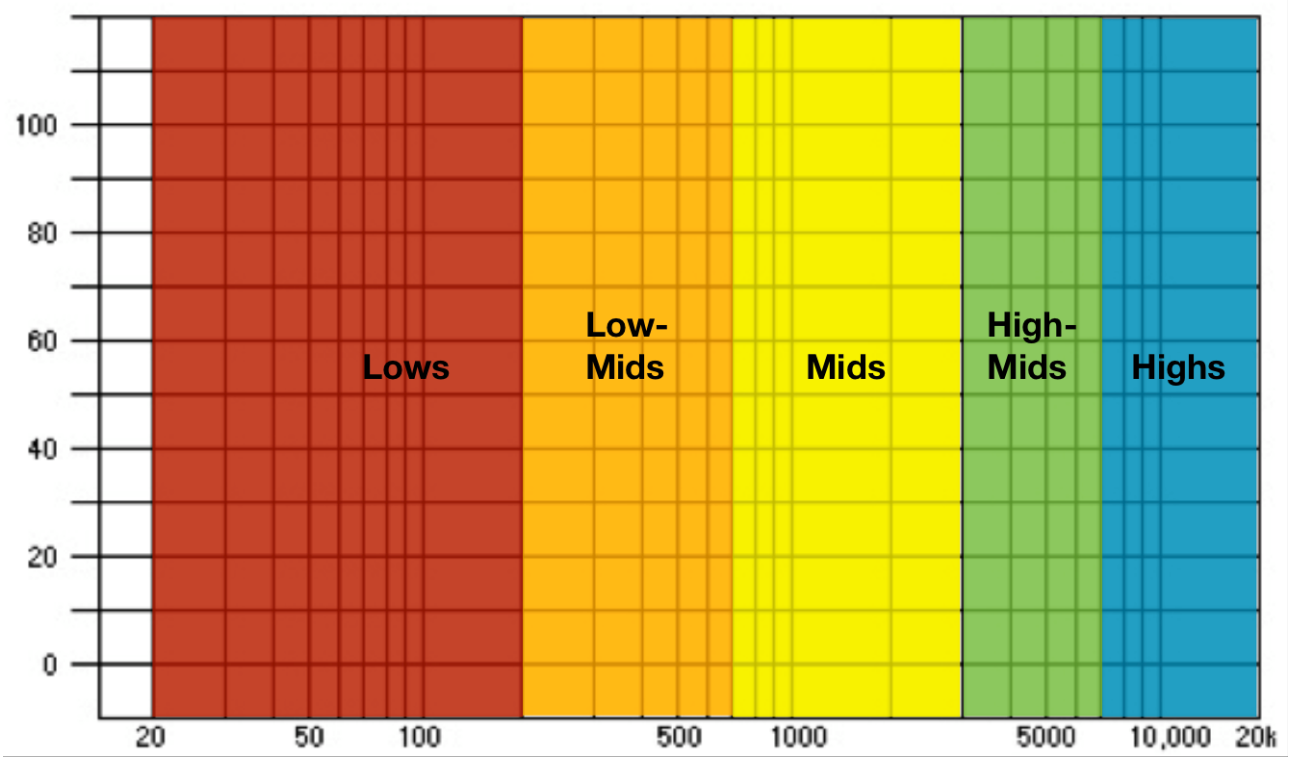

Figure 10 Descriptive representation of the typical frequency ranges (Original Source: MacProvideo)

4.2. User profiling: Audiometry test

As a brief definition, an audiometry test consists in a painless evaluation that seeks to measure the person's ability to hear different kind of sounds, pitches or frequencies.

The idea of this first stage relies on providing a free and simple audiometry test to end-users by playing different sounds (female voice, male voice, background noise, glass crashing and so on) at different levels and in different media contents (sports, news, cartoons). Depending on their hearing necessities, they may modify the level of the sounds that are listening to in order to automatically adapt the frequency bands.

Once this process ends, the system stores the data provided by the end-users including: the levels of the frequency bands that they selected for each sound, how many times the play the same sound and additionally, some personal information.

In Figure 11, a representation of the pipeline that is followed is presented. Two main phases can distinguished: retrieve the information of the end-users via audiometry and perform the profile analysis which is used to obtain clusters based on the user similarities.

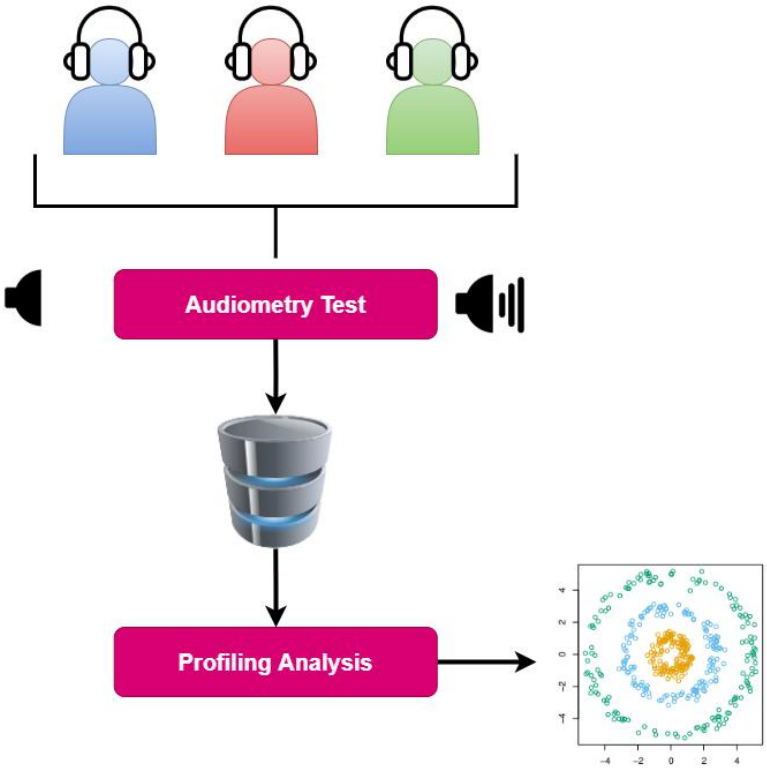

Figure 11 A schema representing the audiometry and the profile analysis stages

Furthermore, a first approach regarding the audiometry test has been implemented but it was not included in the current production version of the platform since the task is under continuous development. Figure 12 shows an illustrative representation of how the audiometry test will be presented to end-users via EasyTV platform. In this initial approach, one may observe a set of bars that represents different filters: High-pass filter (HF), Low-pass filter (LF) a High-shelf filter (HMF) and finally a Low-shelf filter (LMF). Moreover, different profiles are available in order to create customize equalizers by modifying properly the different bars of the filters. Additional documentation regarding the filters can be found at (Web Audio API, n.d.).

The first bar for each band corresponds to the centred frequency of the filters whereas the second bar corresponds to the level that needs to be adopted depending on the needs o the user. On the other hand, the last two components corresponds to other aspects related to the equalizer such as the Q-factor which is related to the bandwidth of the filter and therefore, with the quality of the frequency band.

More bands can be included as well in the interface in order to have the full spectrum completer and more accurate. Moreover, a simple and intuitive documentation will be provided to end-users in order to simplify the process of the audiometry as well as to support them with any help needed.

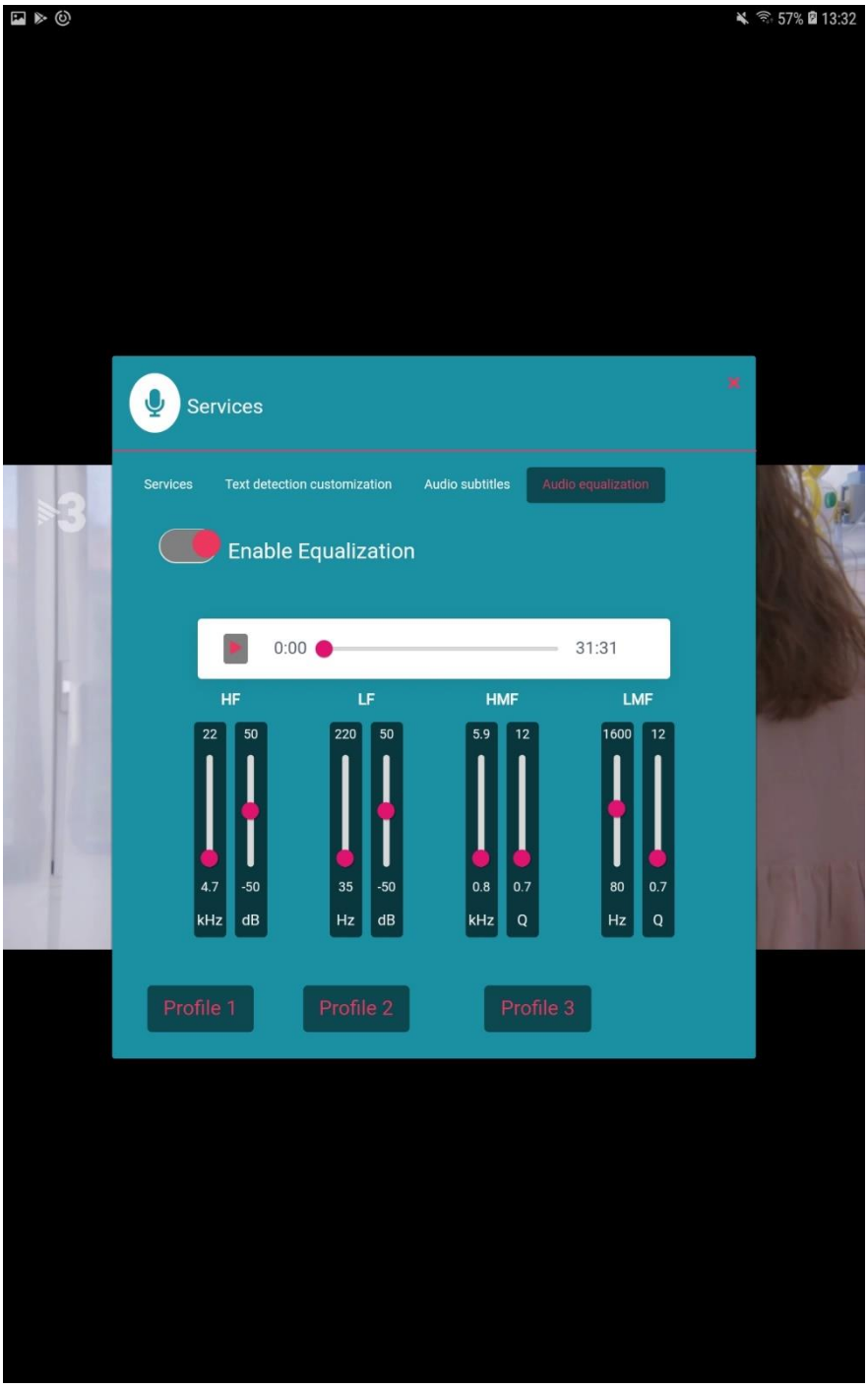

Figure 12 Initial approach for the audiometry test via EasyTV app

4.3. Semi-automatic Sound Equalizer

Once the data has been stored, a model is used to generate a profiling of the different users that are registered into the platform in order to find some clusters that may have similar patterns in terms of auditive response to the input sounds that were played during the audiometry test. Finally, after training the model, each registered end-user will have a cluster associated to it. Moreover, end-users

from the same cluster will have the same distribution of frequency bands depending on the content.

For instance, people from cluster A will have the same level (in dB) at each frequency band for Sports or for soap opera. Thus, when a certain registered end-user may uses the service, it will yield the corresponding level of dBs for the aforementioned frequency bands and he/she will be able to play the media-content with a personalized sound equalization based on his/her needs.

4.4. Service Integration

The semi-automatic sound equalizer service is not available in the current version of the platform, but it will be in the next release of the EasyTV platform in order to be tested by a considerable collection of end-users of different age ranges and auditive limitations.

5. SWAGGER DOCUMENTATION

As it was already mentioned in the previous sections, a general Swagger of the platform can be found at http://138.4.47.33:8080/docs/ where all the available end-points can be consulted.

Moreover, the aforementioned services are also included in this documentation. Figure 13 is a snapshot of the sound analysis section of the API.

More specifically, the first three end-points are POST and are associated to the following functionalities:

- Create the speaker dataset,
- Train the CNN model
- Extract the embeddings

On the other hand, the last two end-points are GET and are associated to:

- Retrieve the speakers of a new episode or media content
- Retrieve the environment sounds detected in a new episode or media content

To see more details, visit http://138.4.47.33:8080/docs/. Moreover, the sound annotation or verification tool that was developed to create the datasets regarding the different media content can be found at http://138.4.47.33:8080/series/verification.

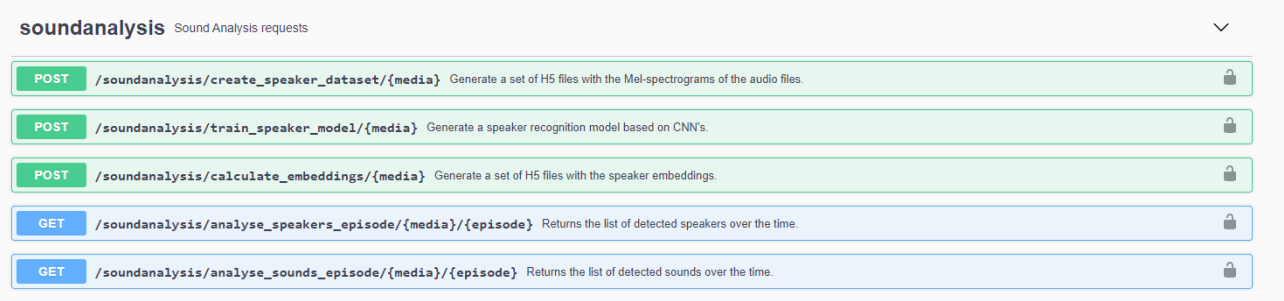

Figure 13 Sound Analysis end-points included in the general EasyTV platform.

6. CONCLUSIONS AND FUTURE LINES

In this document, the three main services that are related to audio engineering were presented and fully described. It was stated that their main goal consists of providing end-users with additional

information about the speakers that are talking in a certain scene as well as the environment sounds that may appear in such scene. All this information is then stored in the subtitles of the media content so that, it reinforces the quality of experience of users with certain hearing disabilities by offering additional information about such content.

Furthermore, a first approach of the Environment Sound System was implemented and tested in the platform and since the results in real scenarios were not as good as the ones in a research stage, some solutions arose including the generation of a specific dataset using the available media content in the platform instead of using public datasets where the data is very clean and without multiple events at the same time. Moreover, more experiments using Siamese Neural Networks (De Baets, 2019) will be carried out to promote the classification task of sound events.

Regarding the Speaker Diarization System, a CNN model was presented to solve the problem and some experiments using public datasets were conducted. Additionally, a service was integrated in the platform and some tests were performed using available real data. As before, the results were not as good as in the research phase since the data used to train the models were not enough to characterize all the speakers available in a specific media-content. To mitigate this problem, more data will be annotated via the annotation tool. As future work, more models will be tested in the production to optimize the performance of the service.

Finally, a first approach of the semi-automatic sound equalizer was developed and reported in Section six. This first approach will be soon available as well in the EasyTV platform. Moreover, more experiments will be conducted in order to validate the profiling of the end-users.

7. REFERENCES

Campello, R. J. (2013). Density-based clustering based on hierarchical density estimates. *Pacific-Asia conference on knowledge discovery and data mining*.

Cubuk, E. D. (2019). AutoAugment: Learning Augmentation Strategies From Data. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* .

De Baets, L. D. (2019). Detection of unidentified appliances in non-intrusive load monitoring using siamese neural networks. *International Journal of Electrical Power & Energy Systems*.

Doukhan, D. C. (2018). An open-source speaker gender detection framework for monitoring gender equality. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Flask. (s.f.). Obtenido de Pallet Projects: http://flask.palletsprojects.com/en/1.1.x/

Fu, W. &. (2019). Estimating the number of clusters using cross-validation. *Journal of Computational and Graphical Statistics*.

Garcia-Romero, D. S. (2017). Speaker diarization using deep neural network embeddings. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Google. (s.f.). *WebRTC*. Obtenido de WebRTC: https://webrtc.org/

Keras. (n.d.). Retrieved from Keras: https://keras.io/

McInnes, L. H. (2017). hdbscan: Hierarchical density based clustering. *J. Open Source Software*.

Open SLR. (n.d.). Retrieved from LibriSpeech ASR corpus: http://www.openslr.org/12/

Pang, J. (2017). Spectrum energy based voice activity detection. *IEEE 7th Annual Computing and Communication Workshop and Conference* .

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. En K. J. Piczak, *Proceedings of the 23rd Annual ACM Conference on Multimedia.* Brisbane, Australia: ACM Press. Obtenido de https://github.com/karolpiczak/ESC-50

Romanov, E. &. (2019). Above the Nyquist rate, modulo folding does not hurt. *IEEE Signal Processing* .

Sailor, H. B. (2017). Unsupervised Filterbank Learning Using Convolutional Restricted Boltzmann Machine for Environmental Sound Classification. *INTERSPEECH*.

Salamon, J. &. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing*.

Tensorflow. (n.d.). Retrieved from Tensorflow: https://www.tensorflow.org/

Van Der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*.

Wang, Q. D. (2018). Speaker diarization with lstm. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* .

Web Audio API. (n.d.). Retrieved from Web Audio API: https://webaudio.github.io/web-audio-api/#biquadfilternode

Zajíc, Z. H. (2017). Speaker Diarization Using Convolutional Neural Network for Statistics Accumulation Refinement. *INTERSPEECH* .

Zhang, A. W. (2019). Fully supervised speaker diarization. *IEEE International Conference on Acoustics, Speech and Signal Processing*.