MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

DEEP LEARNING FOR ACOUSTIC SIGNAL PROCESSING

Case Study:
Speaker Recognition through One-Shot Learning and Siamese Neural Networks

Jaime Pérez Sánchez
Course 2019/20

Table of Contents

1. Deep Learning In Acoustic Applications

Acoustics is the branch of physics that studies the transmission and characteristics of mechanical waves, that is, those waves that are transmitted through materials whether they are gases, liquids or solids. It is a very broad field of study and the variety of sub-disciplines and applications is so vast that a whole report (or book) would be needed to describe it correctly. Still, we'll attempt a brief approach that will give us an overview of this scientific discipline. We will also address and discuss the possible applications of Deep Learning in this field of study.

1.1. Map of Acoustic Application Fields

The following is a brief description of the main sub-disciplines within acoustics, based on the PACS (Physics and Astronomy Classification Scheme) list used by the Acoustical Society of America **[1]**. It is worth noting that many of the following areas are interrelated with each other, making the acoustic discipline a vast and complex field.

- **Aeroacoustics**: Studies the noises generated by air movements and the transmission of sound through fluid air. It is also of great interest to understand how wind musical instruments work.
- **Acoustic Signal Processing**: Studies the electronic manipulation of acoustic signals. It has numerous sub-disciplines and applications, motivated by digital development, such as active noise control, music production, echo cancellation, perceptual coding, etc.
- **Electroacoustics**: Studies the recording, manipulation, and reproduction of audio through the use of electronics. This sub-discipline is highly related to that of Acoustic Signal Processing.
- **Architectural Acoustics**: Studies the acoustic quality of buildings and rooms according to their architecture and design. This discipline is essential in the construction of auditoriums and theaters.
- **Bioacoustics**: Studies the hearing and calls of animals, as well as how they are affected by the acoustics and sounds of their habitat.
- **Noise**: The study of noise is a sub-discipline that influences many other branches and has a great variety of applications. The main objective of these studies is the modeling and attenuation (or cancellation) of undesired sounds (or vibrations) in different environments.
- **Musical Acoustics**: Studies, among others, the physics of musical instruments, the processing, and analysis of audio signals in electronic music, musical composition and perception, and cognitive neuroscience of music.
- **Psychoacoustics**: Studies the relationship between acoustics, and human perception and cognition. It also studies the implications, interrelations, and influences of human psychology (and biology) with music and sounds.
- **Speech**: It studies the perception, processing, and reproduction of speech, usually with the use of computers. The most important areas are Speech recognition and speech synthesis. This sub-discipline is also influenced by other branches of science such as psychology, physics, and linguistics.

- **Ultrasonics**: Studies sounds at frequencies too high to be heard by humans. It has numerous applications in medicine, chemistry, material characterization, and underwater acoustics.
- **Underwater Acoustics**: It studies the transmission of natural and artificial sounds in liquid fluids such as water. It includes applications such as SONAR for submarines navigation, marine bioacoustics, and climate change studies.
- **Vibration and Dynamics**: Studies how mechanical systems vibrate and interact with their environment. Applications include the study of the transmission and isolation of mechanical waves in materials and architectural structures such as buildings.

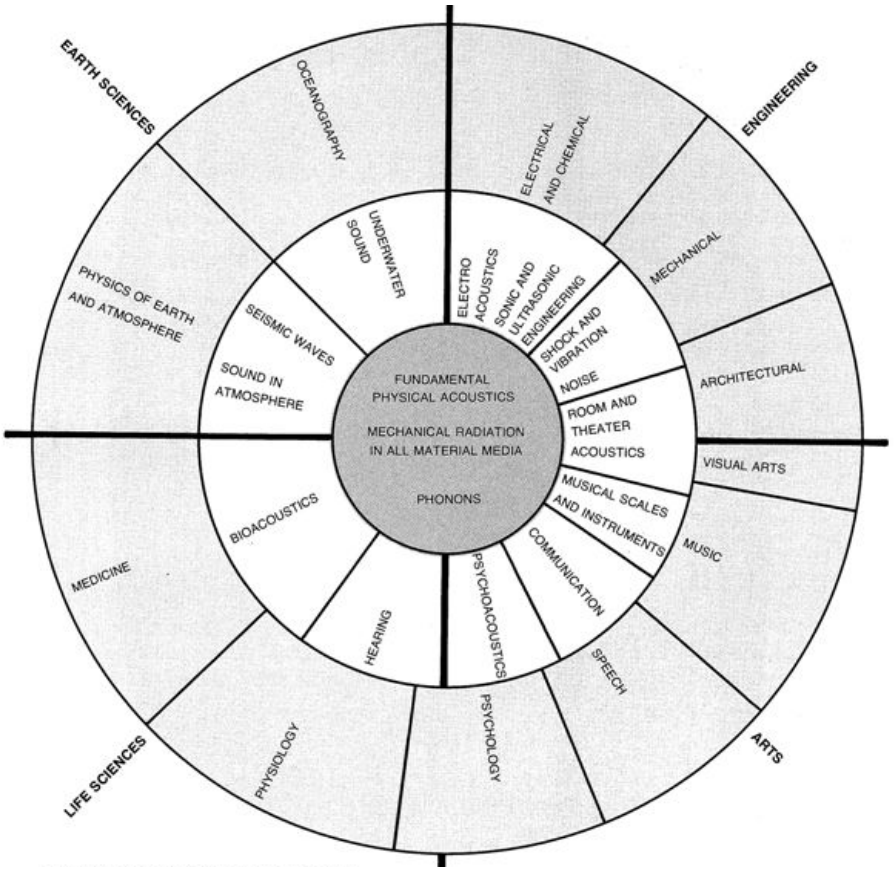

Figure 1. "Lindsay's Wheel of Acoustics"

In Figure 1 we can find the so-called "Lindsay's Wheel of Acoustics", created in 1964 for the Acoustical Society of America journal. This wheel describes the main fields of study of acoustics, dividing it into four major fields (Life sciences, Engineering, Arts, and Earth sciences). It is considered of great historical importance in this field of study.

1.2. Deep Learning for Acoustic Application Fields

Machine Learning is a subset of artificial intelligence that studies computer algorithms that improve automatically through the analysis of data. These techniques have enabled broad advances in automated data processing and pattern recognition capabilities. The measurement and labeling of the data used are often uncertain, thus statistical methods are often involved.

Deep Learning is, in turn, a subset of Machine Learning based on the interconnection of non-linear feature transforms (Artificial Neural Networks) trained through massive amounts of data. The motivation for using these algorithms in acoustics fields is that the physical models are too specialized and not capable of capturing the subtleties of the phenomena underlying data. Often it is beneficial to learn the representation directly from large collections of examples using these high-capacity machine learning models. These Deep Neural Network architectures are capable of learning complicated non-linear patterns from the data. In recent years they have proven to be state-of-the-art algorithms for the automatic execution of complex tasks, even in some cases exceeding human performance.

Research in acoustics has traditionally focused on developing high-level physical models and using these models for inferring properties of the environment and the objects in it **[2]**. However, these models are often highly complex to develop and require major restructuring if environmental conditions change slightly.

The use of Machine Learning and Deep Learning techniques in the field of acoustics has many applications that have been observed in recent years, finding them to achieve better results than conventional signal processing methods. The traditional use of Machine Learning algorithms does not explicitly require specific domain knowledge to obtain satisfactory results. However, machine learning-based methods have significant limitations: they are data-driven and therefore require huge amounts of data to train and test the models; the obtained results are less interpretable than traditional physical methods (particularly deep learning models can be considered black-boxes, meaning that the intervening operations between the inputs and outputs of the system are not necessarily physically intuitive).

Despite all this, there is a still-young field of study that aims to combine the domain-specific knowledge of physical methods, with the automatic learning capacity of machine learning algorithms. This field has enormous potential and promising results never seen before in the area of acoustics. In the following sections, we will explain and develop the use case chosen for this project.

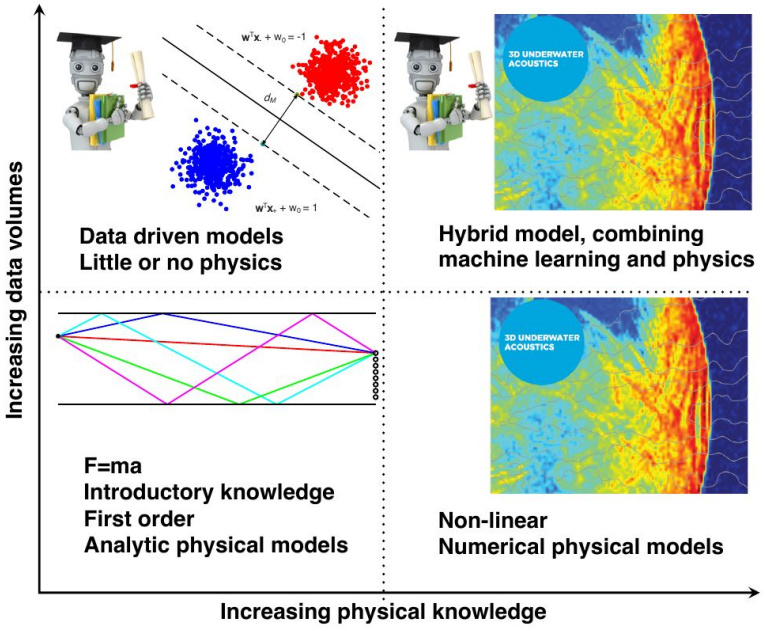

Figure 2. Approaches to the field of acoustics using Machine Learning, physical models, and both

2. Case Study: Speaker Recognition through One-Shot Learning and Siamese Neural Networks

2.1. Motivation and Uses Cases

Speaker recognition is the identification of a person and distinguishes from others, based on its voice characteristics. These specific voice attributes reflect patterns from both anatomy and learned behavior. The two main branches in speaker recognition (verification and identification) have many potential use cases such as: (i) simplify the task of translating speech in systems that have been trained on specific voices; (ii) improved and personalized service in technologies whose only interface is voice, such as Alexa or Google Home; (iii) complement biometric verification methods in security systems, such as smartphones; (iv) uses in criminal investigations.

It is worth noting the difference between **speaker verification and identification**, even though many of their use cases and methodologies are very similar. When a speaker claims to be a certain identity and the voice is used to verify this claim, this is called speaker verification. On the other hand, identification is the task of determining an unknown speaker's identity. In a sense, speaker verification is a 1:1 match where one speaker's voice is matched to a particular template, whereas speaker identification is a 1:N match where the voice is compared against multiple templates. **[3]**

Neural networks are the state-of-the-art algorithms in many classification problems, particularly those on perceptual data such as images, video, and audio. However, this is true only when we have huge amounts of labeled examples to train on. **One-shot learning** aims to address this problem of learning useful information from only one (or a few) training samples. In the use cases of the speaker recognition field mentioned earlier, the problem of having many categories and very few data from each one is very present. As in the case of speaker identification, where we must differentiate among speakers with only a few seconds of data. In recent years there have been many interesting papers about one-shot learning with neural networks, where quite satisfactory results have been achieved. This new field of research has a huge potential to transform the way we conceive the problems of Machine and Deep Learning.

2.2. LibriSpeech Dataset

For the development of this project, it has been decided to use the "LibriSpeech" Dataset **[4]**. This dataset is a corpus of approximately 1000 hours from more than 200 speakers of reading English speech, prepared by Vassil Panayotov with the assistance of Daniel Povey. The data was recorded under a controlled environment with just one speaker at a time and little variance of background noise between recordings. This database was chosen since it is free and text-independent, which will make the model robust against any phrasing.

As this project is closer to a proof of concept than a real application, we will only use a part of this dataset to facilitate data handling and to speed up training times. For the training phase, we will use data samples from 80 different readers; and from 40 different unseen readers for the testing and validation phases.

2.3. Methodology

A structure that has obtained great results in one-shot (or few-shot) learning tasks are the **Siamese neural networks**. Unlike typical deep learning structures, Siamese networks have two separate inputs, each of them is mapped from a high-dimensional space into a low-dimensional one by an encoder network. The "siamese" nomenclature comes from the fact that the two encoder networks are "twins", as they share the same structure and weights, and learn the same function.

These two networks are then joined at the top by a layer that calculates a measure of distance (e.g. Euclidean distance, cosine distance, etc) between the two samples in the embedding space. Instead of matching inputs with certain fixed classes, this network is trained to make the distance small for similar samples, and large for dissimilar samples. Notice that this network is not learning a classification accuracy function, by mapping the inputs to one of the fixed output classes. Rather, this network is learning a **similarity function**, by optimizing directly the properties of the embeddings. In Figure 3 we can observe a simplified diagram of a siamese neural network structure.

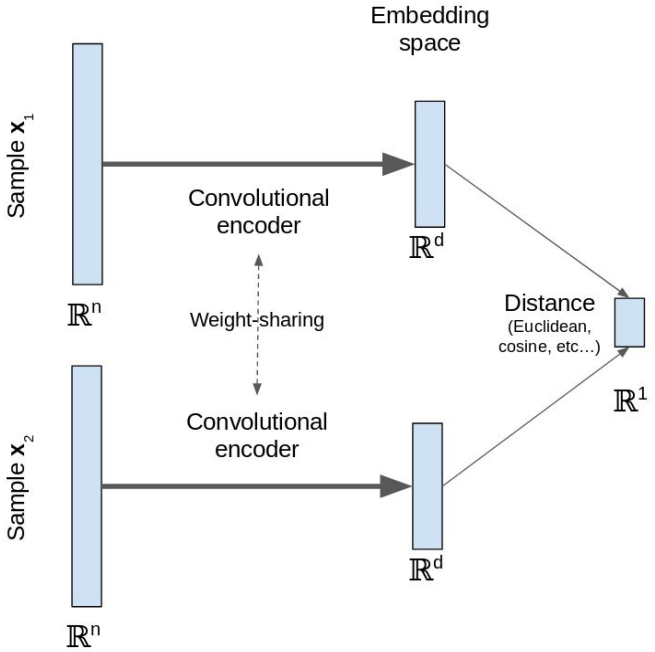

Figure 3. Diagram of a Siamese Neural Network structure

Siamese networks were introduced in 1994 by Bromley et al. **[5]** to verify matches between handwritten signatures. However, they were repurposed by Koch et al. **[6]** in 2015 for one-shot learning tasks.

We will **validate** our speaker recognition system using a methodology usually known as an ***n-shot k-way classification task***. For this project, we will only use a one-shot learning approach, thus n=1. To do so, we will apply the following procedure:

I. A new query sample from an unseen class is given to the model.
II. A "support set" of n samples from k unseen classes is also given to the model.
III. The model then has to identify which sample, in the support set (II) belongs to the same class of the first given query sample (I). To this end, the model produces k different similarity scores, where the maximum is the predicted class.

An example of one-shot 9-way learning is shown in Figure 4, where images of hand-drawn characters from an alphabet are used.

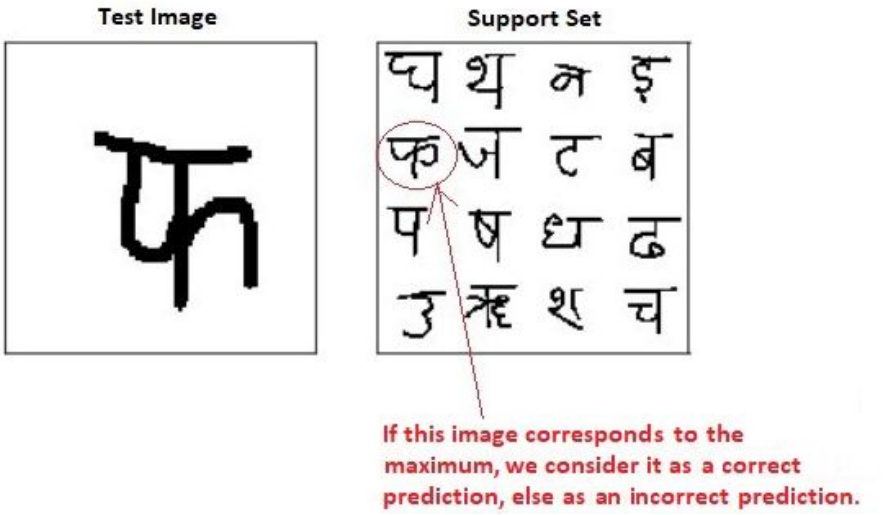

Figure 4. Example of one-shot 9-way learning

In case we would like to extend this *n-shot k-way* validation method for n > 1, several approaches exist. One way is to obtain directly the representation in the embedding space of the validation data, and then perform a nearest-neighbor classification. Another slightly different method is to calculate the mean position of the embeddings belonging to each class, and then select as the prediction the class with the mean embedding at a shorter distance from the query sample embedding. Figure 5 shows an example diagram of what a 4-shot 3-way classification task would look like. Nevertheless, this will not be implemented in this project and is proposed as a future line of work.

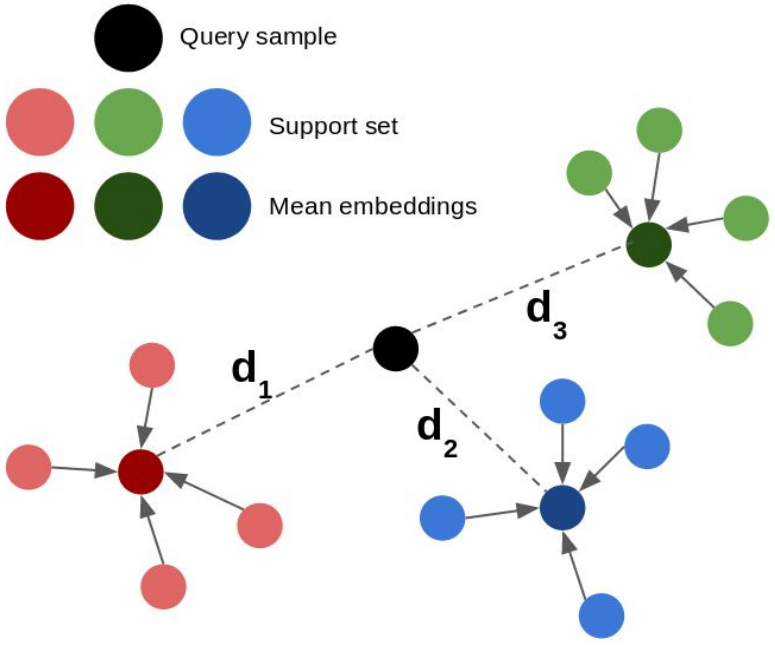

Figure 5. Example diagram of a 4-shot 3-way classification task

It is important to note that, since the neural model requires **pairs of samples at the input**, it will be empirically impractical for us to train the model for all **combinations** of the data. In my implementation, I have decided to create arrays containing the indices of the data pairs that can be used during the training, with half of the pairs corresponding to an alike class and the other half (approximately) to different classes. Since the number of combinations between different classes is much higher and the siamese network requires the classes to be balanced, I randomly sample some of them to match the number of pairs of each type. If we have C samples of each E class, having a total of $C \cdot E$ samples, the total number of possible pairs is given by:

$$N_{pairs} = \binom{C \cdot E}{2} = \frac{(C \cdot E)!}{2!(C \cdot E - 2)!}$$

If we apply this expression to the data used during the training phase, split into 3-second fragments, we obtain a total of 555,961,185 possible combinations in pairs.

Regarding the **data manipulation**, all the audios have been split into 3-second-long fragments. Intuitively, the longer the fragments are, the more accurate the results will be, but it also requires greater temporal and computational resources. **Three different representations** of the audios will be used to compare the results:

A. **Raw Audio**

This is the original data representation used in the dataset, audio sampled at 16 kHz. To speed up the training times and reduce complexity, it has been decided to subsample the audio at 4 kHz when using this representation.

B. **Mel Frequency Cepstral Coefficients (MFCCs)**

The Mel-Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of acoustic signals, based on a linear cosine transform of a logarithmic power spectrum on a non-linear Mel scale of frequency. This representation is of great importance in the representation of audio signals since the frequency bands are equally spaced in the Mel scale, which is closer to the human auditory system response than the linear scale used in other typical spectral representations.

From this representation, we can obtain two types of data to feed the models: The spectrogram on the non-linear Mel scale itself; or the Mel Frequency Cepstral Coefficients (MFCCs) that collectively compound an MFC. Figures 6 to 8 show respectively the spectrogram of an audio signal, the filter bank on a Mel-scale and the MFCCs.

Eventually, what we obtain are bi-dimensional arrays of values that can be treated as images (although with some limitations). For a better understanding of these concepts, I recommend the following *medium* articles **[7] [8]** and to explore the limitations of audio spectrogram analysis with image processing methods (CNNs) I recommend the following article **[9]**.

C. **VGGish Embeddings**

VGGish **[10]** is a pre-trained model shared by Google, that was trained on over 2 million human-labeled YouTube video segments, a preliminary version of what later became YouTube-8M dataset **[11]**. We will use transfer learning from this pre-trained model, in order to extract fundamental features from our dataset. The model extracts 128-dimensional embedding arrays, which helps greatly to reduce the dimensionality of the data.

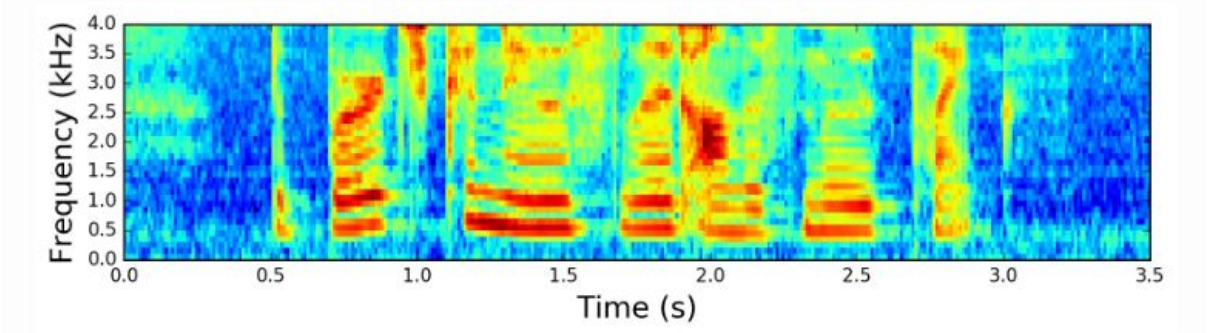

Figure 6. Spectrogram of an audio signal

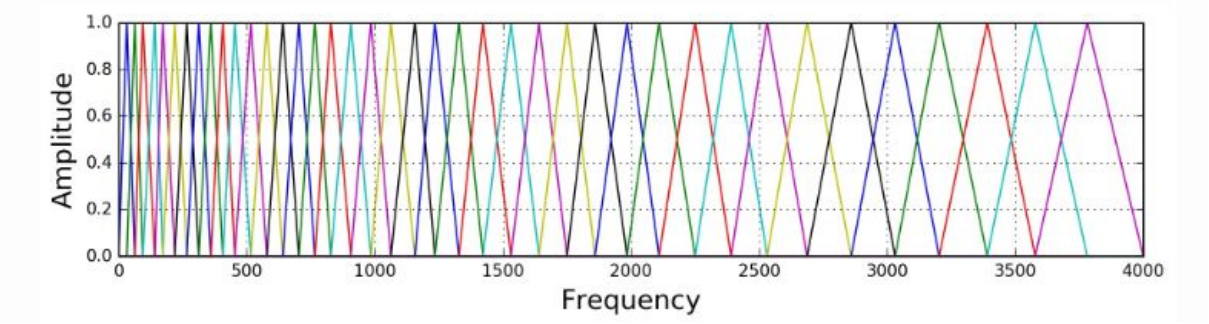

Figure 7. Filter bank on a Mel-scale

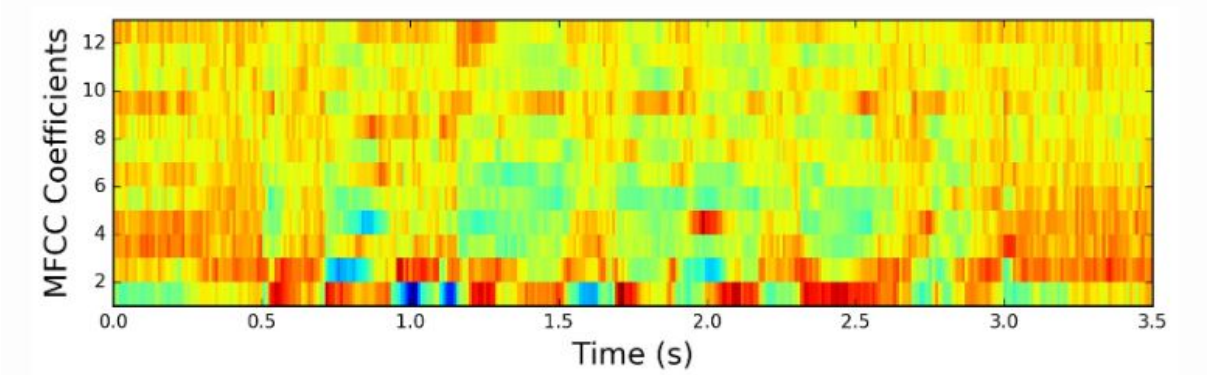

Figure 8. Mel Frequency Cepstral Coefficients (MFCCs) of an audio signal

3. Experiments

In this section, we will explain the conducted experiments and discuss the obtained results. The project has been implemented in the Google Colab tool using Python programming language (version 3.7). The notebooks can be found in the following Google Colab links **[12] [13] [14] [15]** and also in my GitHub repository **[16]**.

3.1. Raw Audio

In this collection of experiments **[12]**, raw audio was used directly from the dataset, although to speed up training times and reduce complexity, the data were subsampled from 16 kHz to 4 kHz. The general neural network structure used is the following:

Layer		Arguments	Activation Function
2 x Input		Shape (None, 12000, 1)	-
4 x CNN Block	1D Convolution	128 filters // 3 stride // L2 regularizer	ReLU
	Batch Normalization	-	-
	1D Convolution	128 filters // 3 stride // L2 regularizer	ReLU
	Batch Normalization	-	-
	1D Max Pool	-	-
	1D Spatial Dropout	0.2 rate	-
1D Global Max Pool		-	-
Fully Connected		1024 neurons // L2 regularizer	Sigmoid
Dropout		0.2 rate	-
Lambda (Distance measure)		Euclidean distance	-
Fully Connected		1 neuron	Sigmoid

In the following, the experiments conducted with the selected hyperparameters and the results obtained are shown. In the training phase of all the experiments, the RAdam optimizer and the Binary Cross-Entropy loss function were used. The validation phase has been carried out with 1-shot 4-way classification tasks, every 25 training iterations, over 100 tasks. In the embedding space visualization, the t-SNE algorithm has been used to reduce the total dimensions of this space to 2.

3.1.1. Experiment 1

- **Hyperparameters**
  - CNN Layers: 64 filters are used in the first block, 128 filters in the second block, 192 filters in the third block and 256 filters in the fourth block
  - Batch Size: 48
  - Embeddings Dimension: 1024

- **Training Phase**

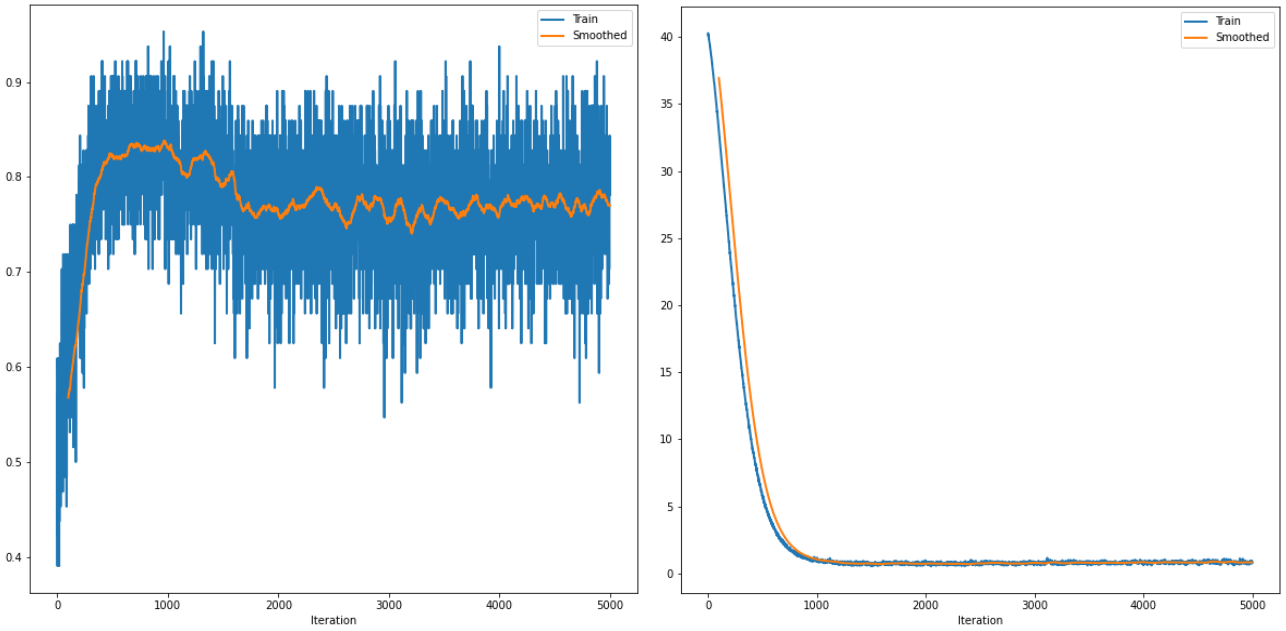

Figure 9. Raw Audio Experiment 1: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

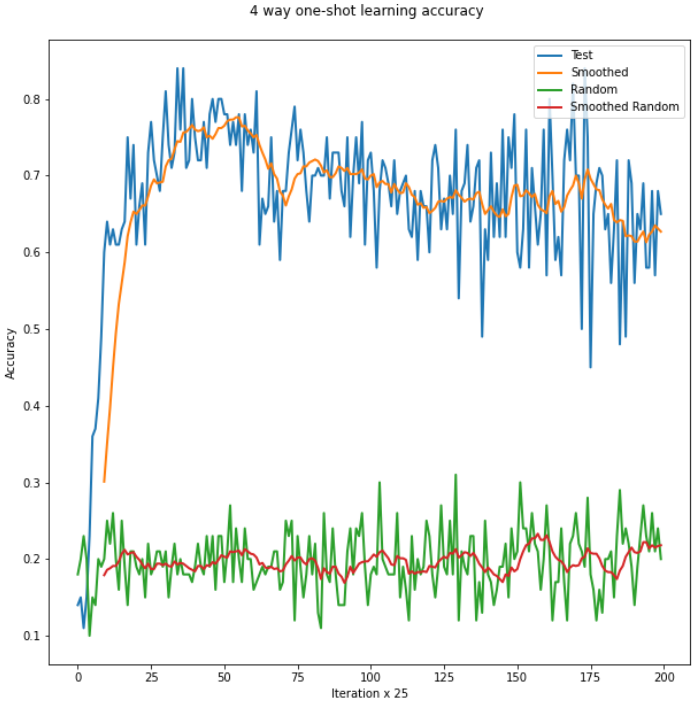

Figure 10. Raw Audio Experiment 1: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

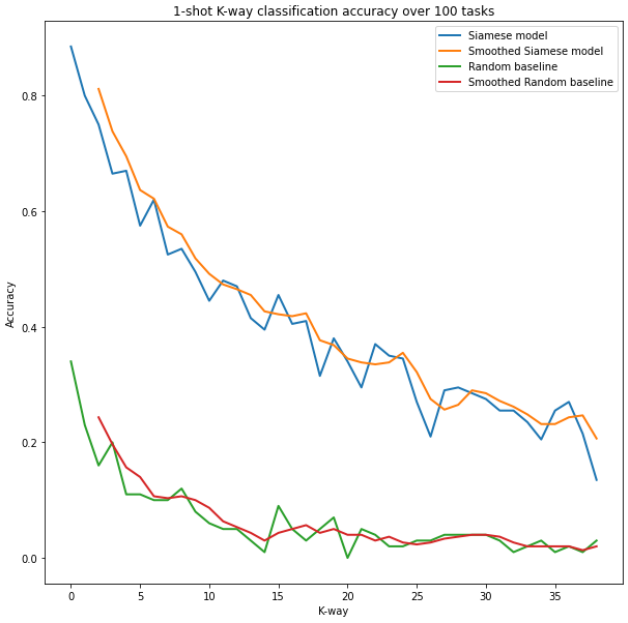

Figure 11. Raw Audio Experiment 1: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

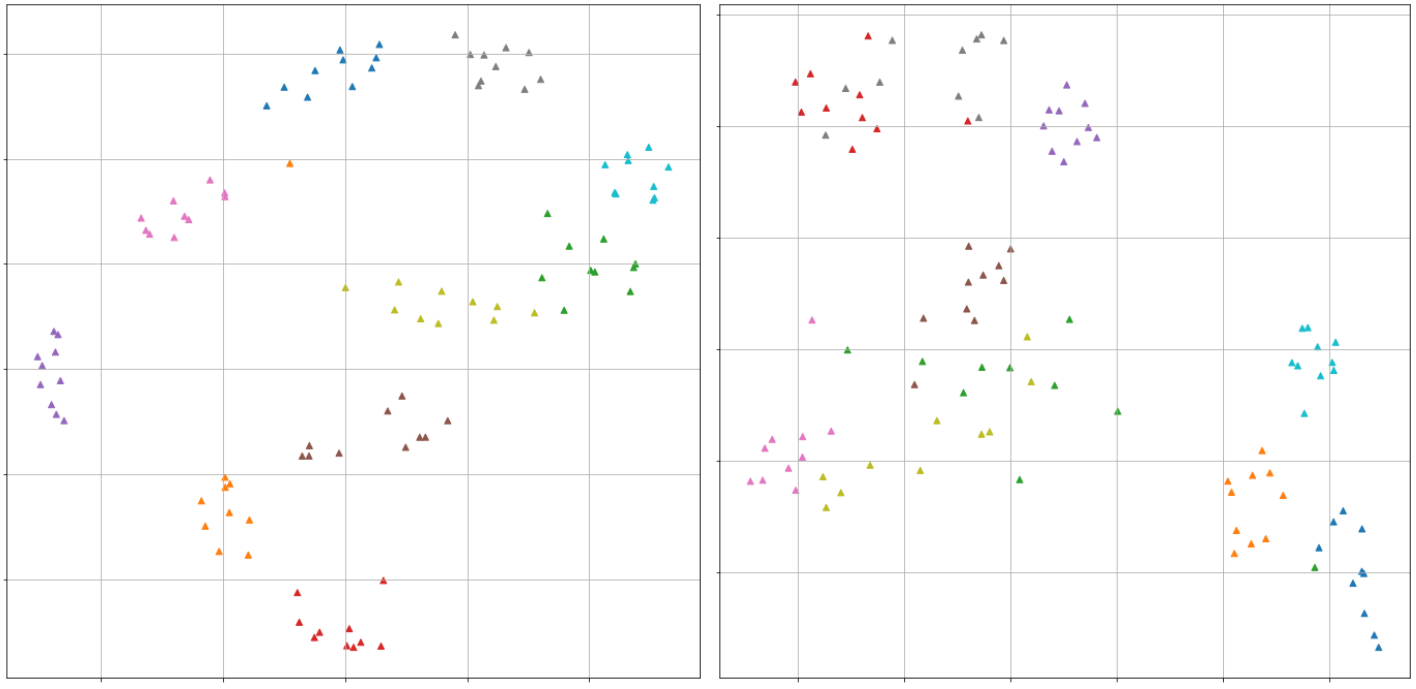

Figure 12. Raw Audio Experiment 1: Embedding Space Visualization

3.1.2. Experiment 2

- **Hyperparameters**
  - CNN Layers: 64 filters are used in the first block, 128 filters in the second block, 192 filters in the third block and 256 filters in the fourth block
  - Batch Size: 128
  - Embeddings Dimension: 64

- **Training Phase**

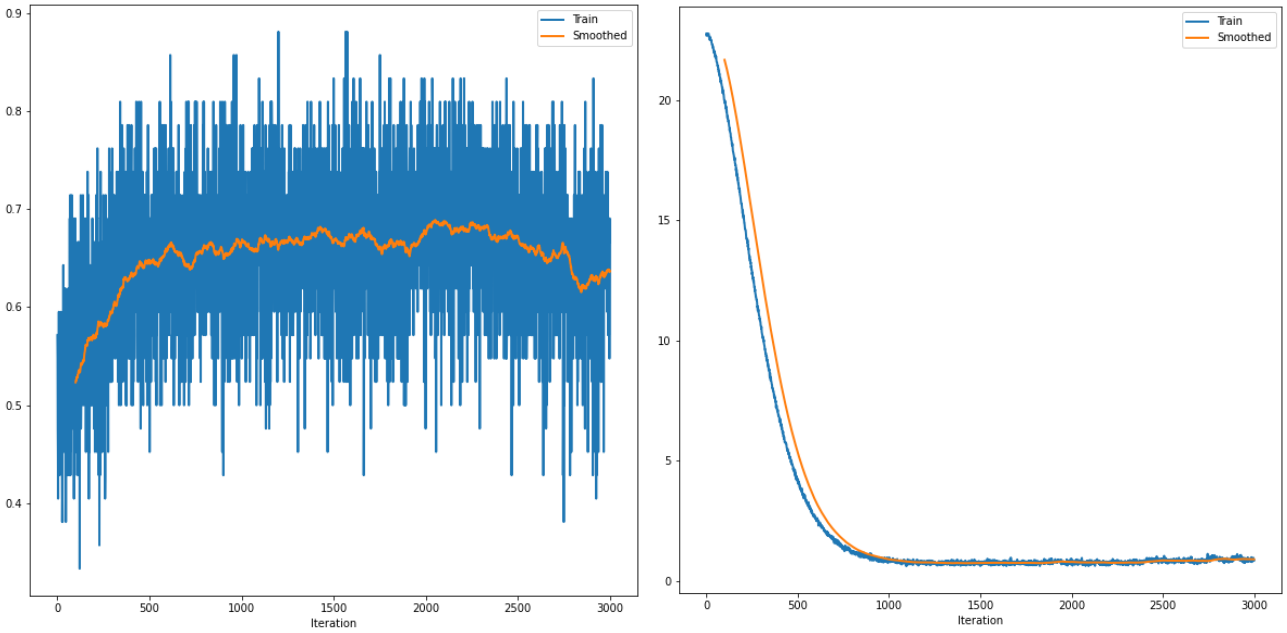

Figure 13. Raw Audio Experiment 2: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

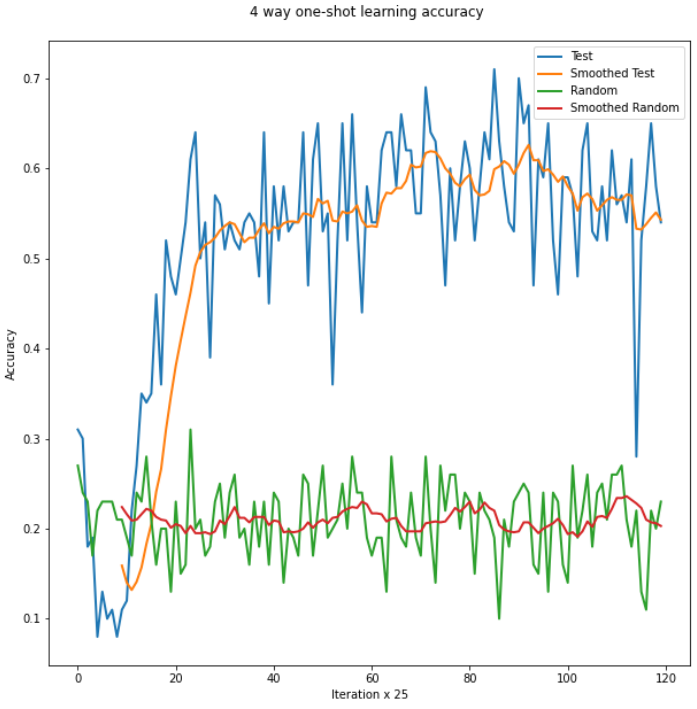

Figure 14. Raw Audio Experiment 2: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

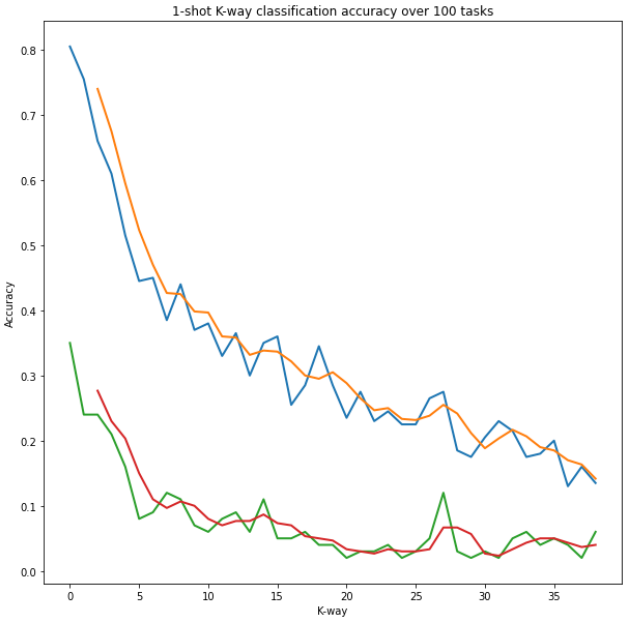

Figure 15. Raw Audio Experiment 2: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

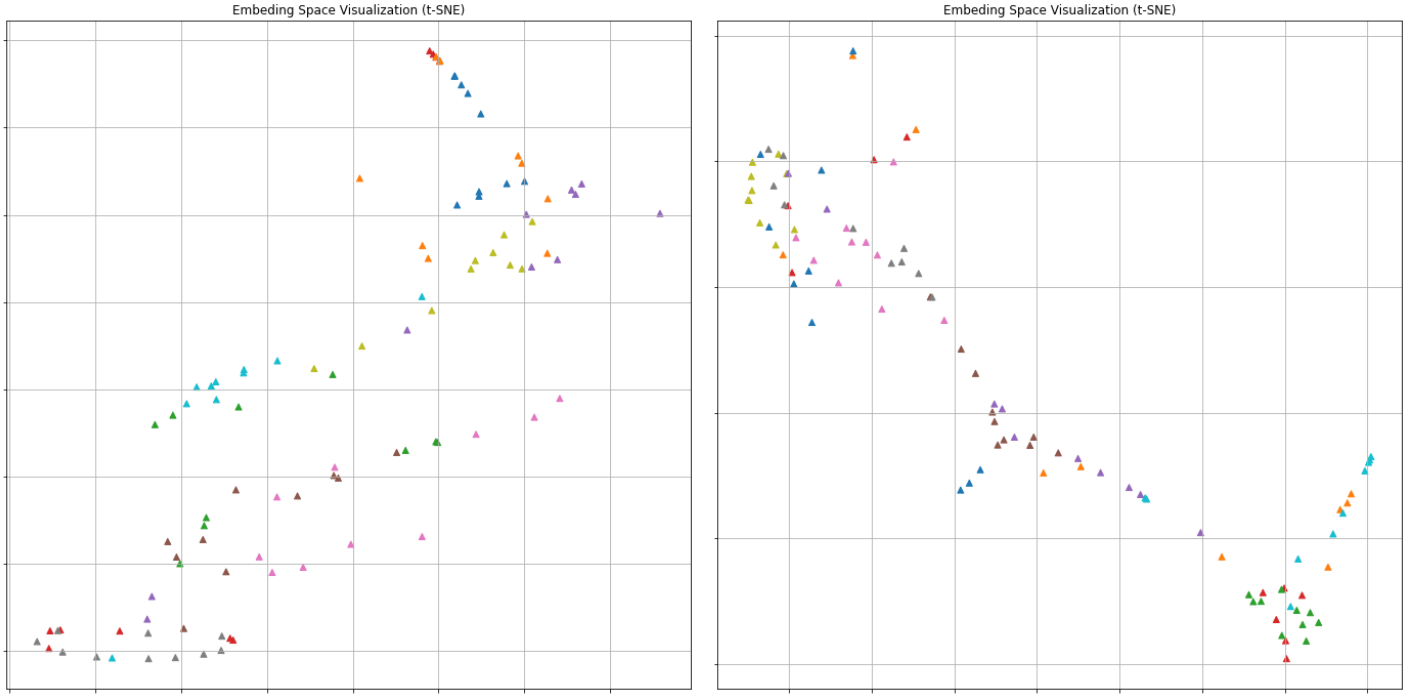

Figure 16. Raw Audio Experiment 2: Embedding Space Visualization

3.1.3. Experiment 3

- **Hyperparameters**
  - CNN Layers: 128 filters are used in the first block, 256 filters in the second block, 384 filters in the third block and 512 filters in the fourth block
  - Batch Size: 48
  - Embeddings Dimension: 2048

- **Training Phase**

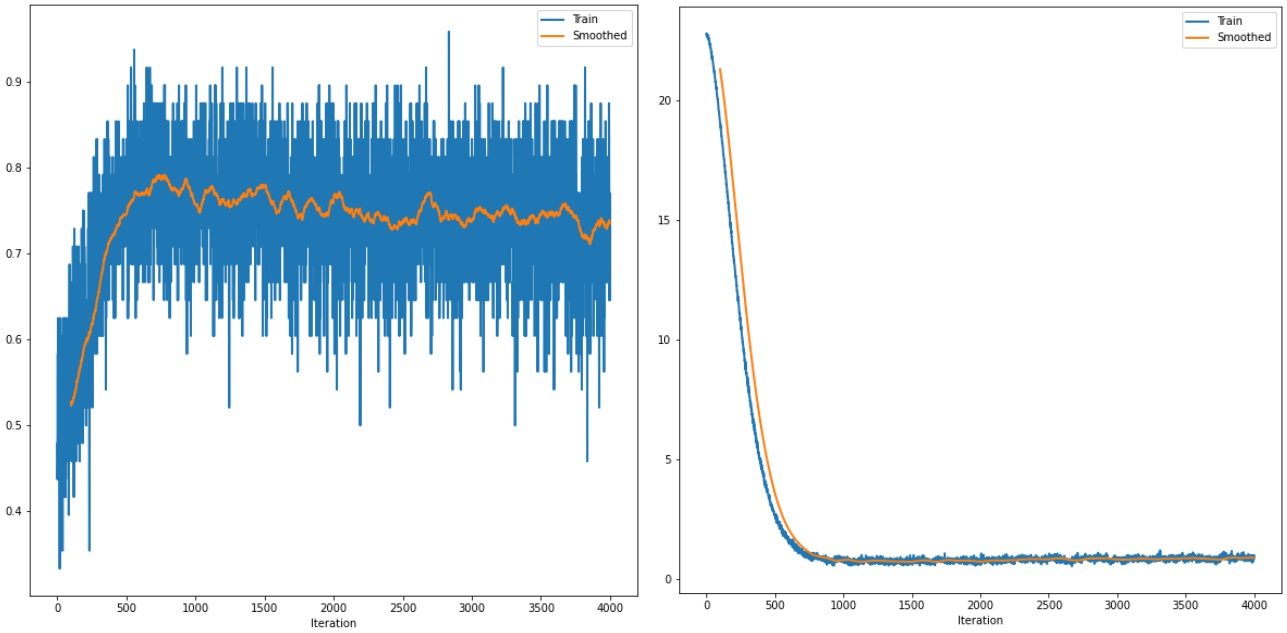

Figure 17. Raw Audio Experiment 3: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

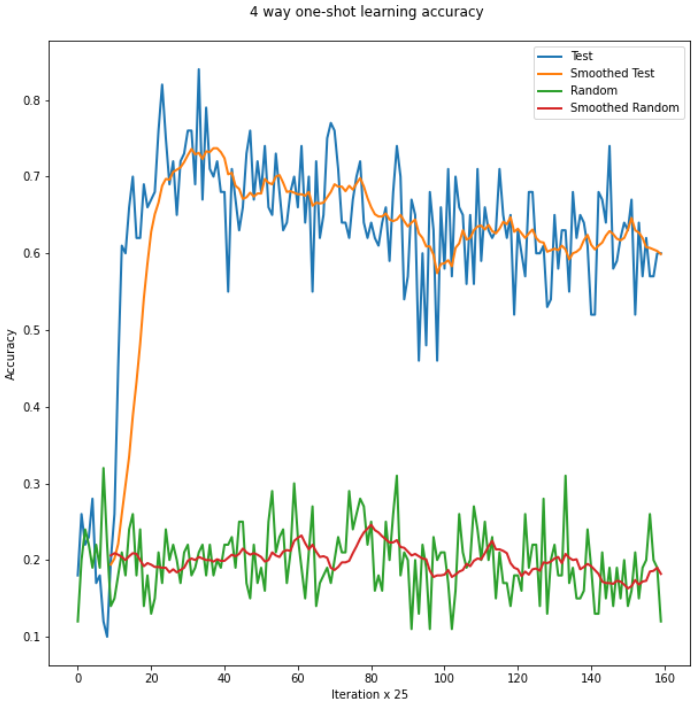

Figure 18. Raw Audio Experiment 3: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

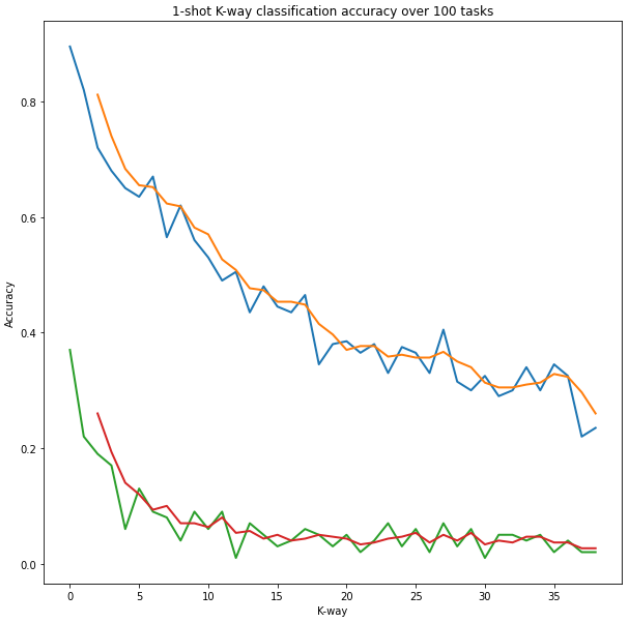

Figure 19. Raw Audio Experiment 3: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

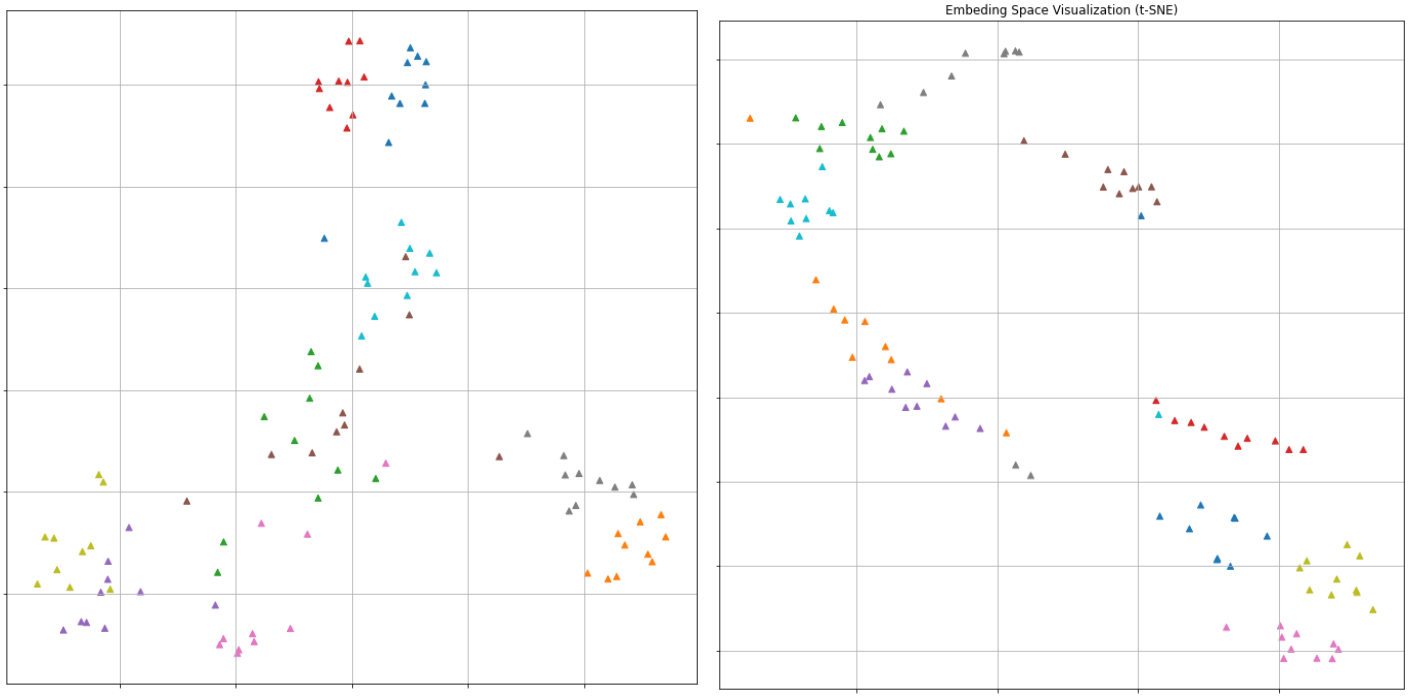

Figure 20. Raw Audio Experiment 3: Embedding Space Visualization

3.2. Mel Frequency Cepstral Coefficients (MFCCs)

In this collection of experiments **[13]**, the MFCC characteristics of the audio segments have been used as inputs to the models. It has been decided to use **32 components of MFCCs** and to obtain them, *Librosa* Python library has been used. By extracting the MFCC features from the 3-second audio fragments, we finally get two-dimensional 32x94 component arrays. The general neural network structure used is the following:

Layer		Arguments	Activation Function
2 x Input		Shape (None, 32, 94, 1)	-
3 x CNN Block	2D Convolution	64 filters // 3x3 stride // L2 regularizer	ReLU
	Batch Normalization	-	-
	2D Convolution	64 filters // 3x3 stride // L2 regularizer	ReLU
	Batch Normalization	-	-
	2D Max Pool	-	-
	2D Spatial Dropout	0.2 rate	-
2D Global Max Pool		-	-
Fully Connected		1024 neurons // L2 regularizer	Sigmoid
Dropout		0.2 rate	-
Lambda (Distance measure)		Euclidean distance	-
Fully Connected		1 neuron	Sigmoid

In the following, the experiments conducted with the selected hyperparameters and the results obtained are shown. In the training phase of all the experiments, the RAdam optimizer and the Binary Cross-Entropy loss function were used. The validation phase has been carried out with 1-shot 4-way classification tasks, every 25 training iterations, over 100 tasks. In the embedding space visualization, the t-SNE algorithm has been used to reduce the total dimensions of this space to 2.

3.2.1. Experiment 1

- **Hyperparameters**
  - CNN Layers: 128 filters are used in the first block, 256 filters in the second block and 384 filters in the third block
  - Batch Size: 64
  - Embeddings Dimension: 1024

- **Training Phase**

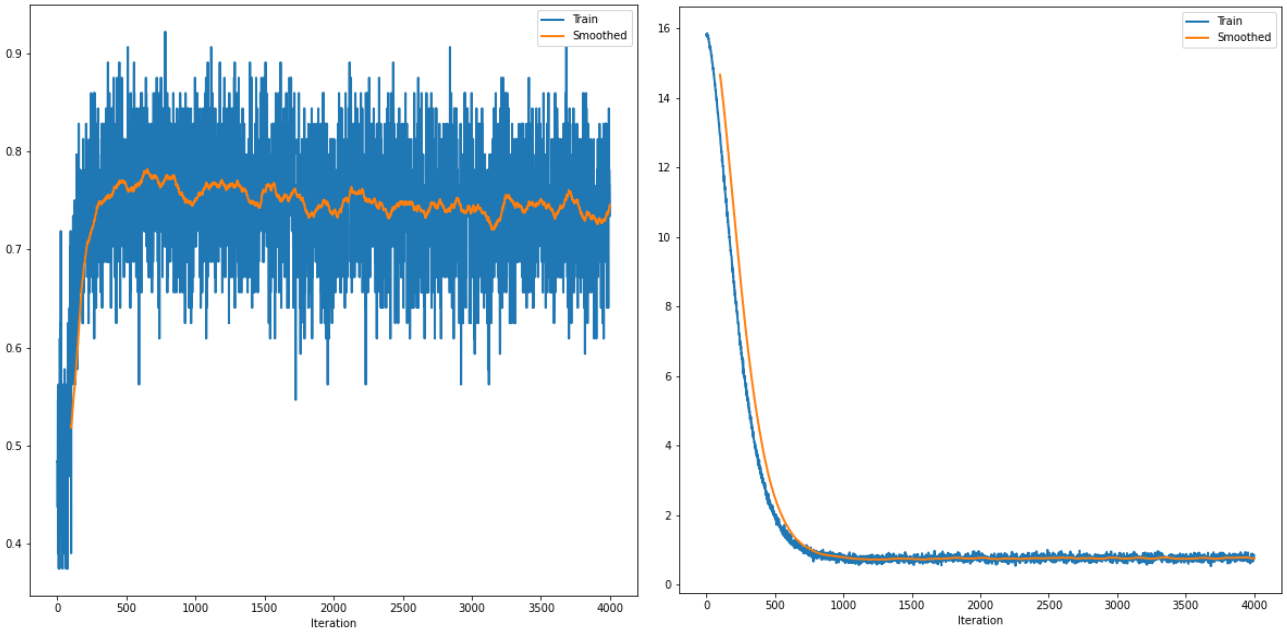

Figure 21. MFCC Audio Experiment 1: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

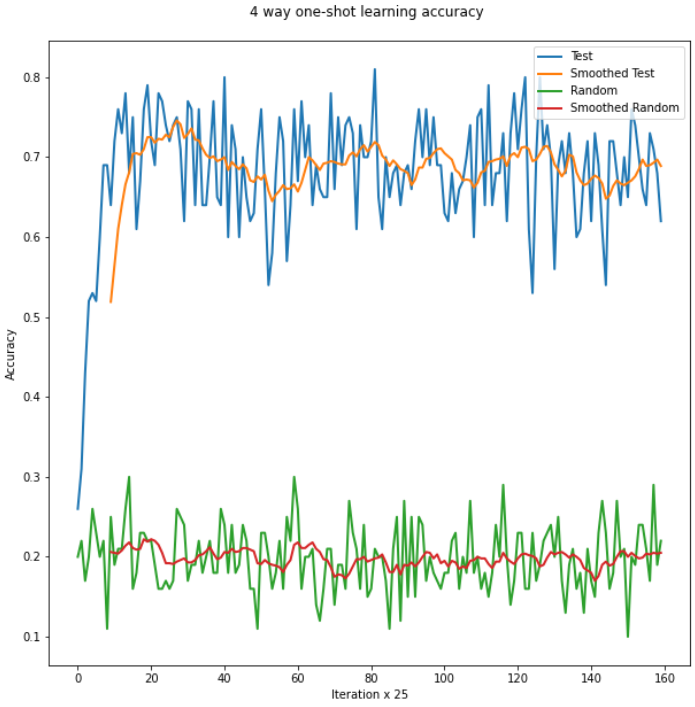

Figure 22. MFCC Audio Experiment 1: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

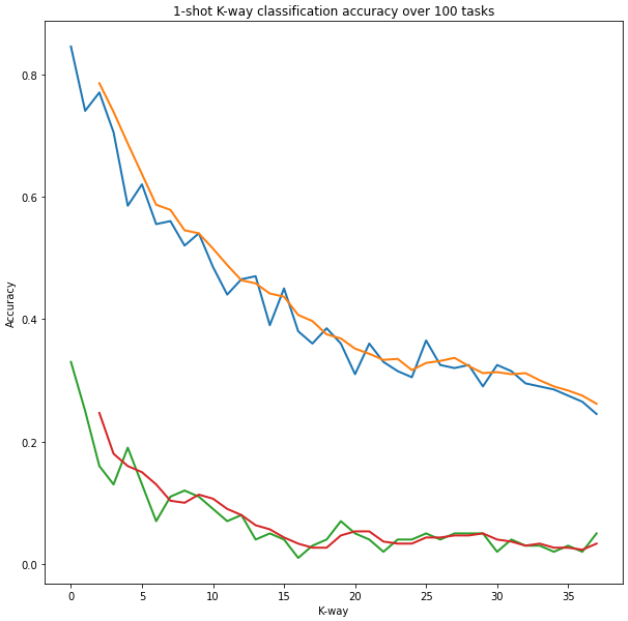

Figure 23. MFCC Audio Experiment 1: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

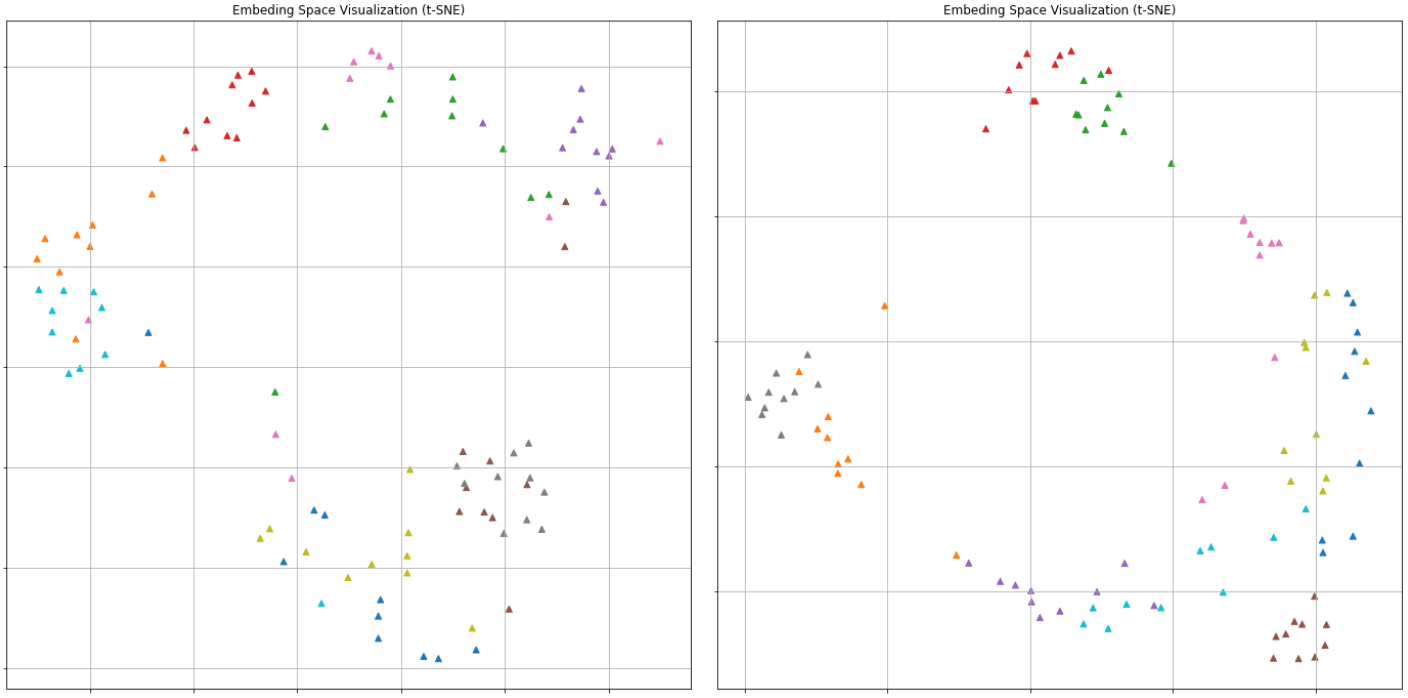

Figure 24. MFCC Audio Experiment 1: Embedding Space Visualization

3.2.2. Experiment 2

- **Hyperparameters**
  - CNN Layers: 64 filters are used in the first block, 128 filters in the second block and 192 filters in the third block
  - Batch Size: 64
  - Embeddings Dimension: 1024
  - Added an extra Fully Connected layer of 1024 neurons

- **Training Phase**

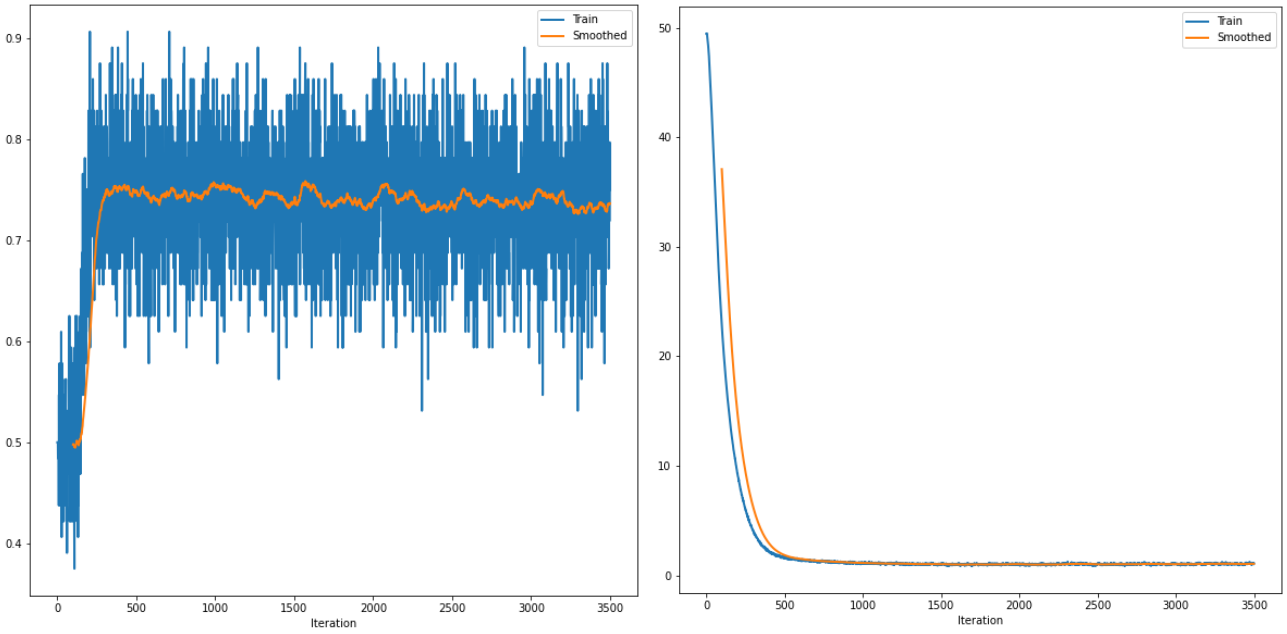

Figure 25. MFCC Audio Experiment 2: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

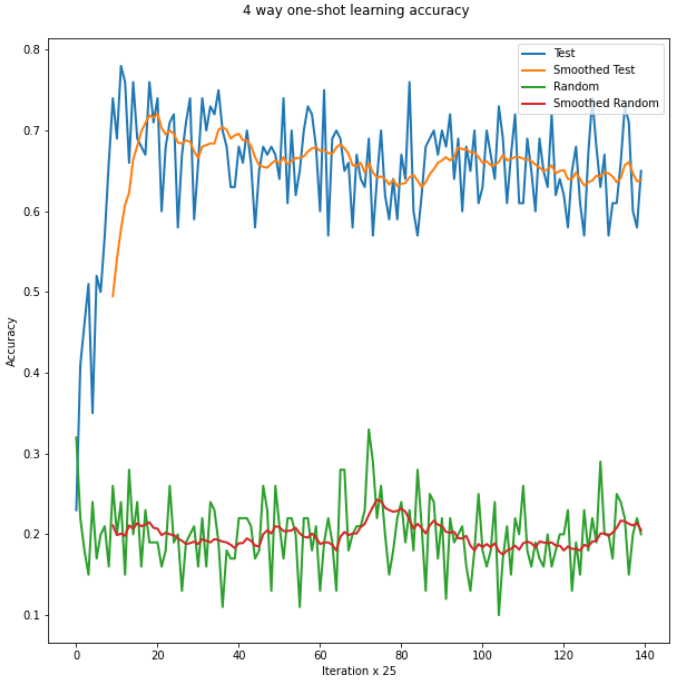

Figure 26. MFCC Audio Experiment 2: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

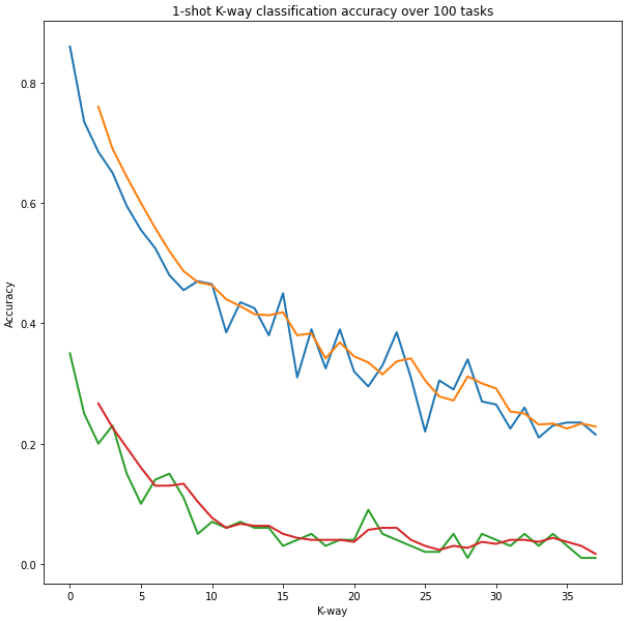

Figure 27. MFCC Audio Experiment 2: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

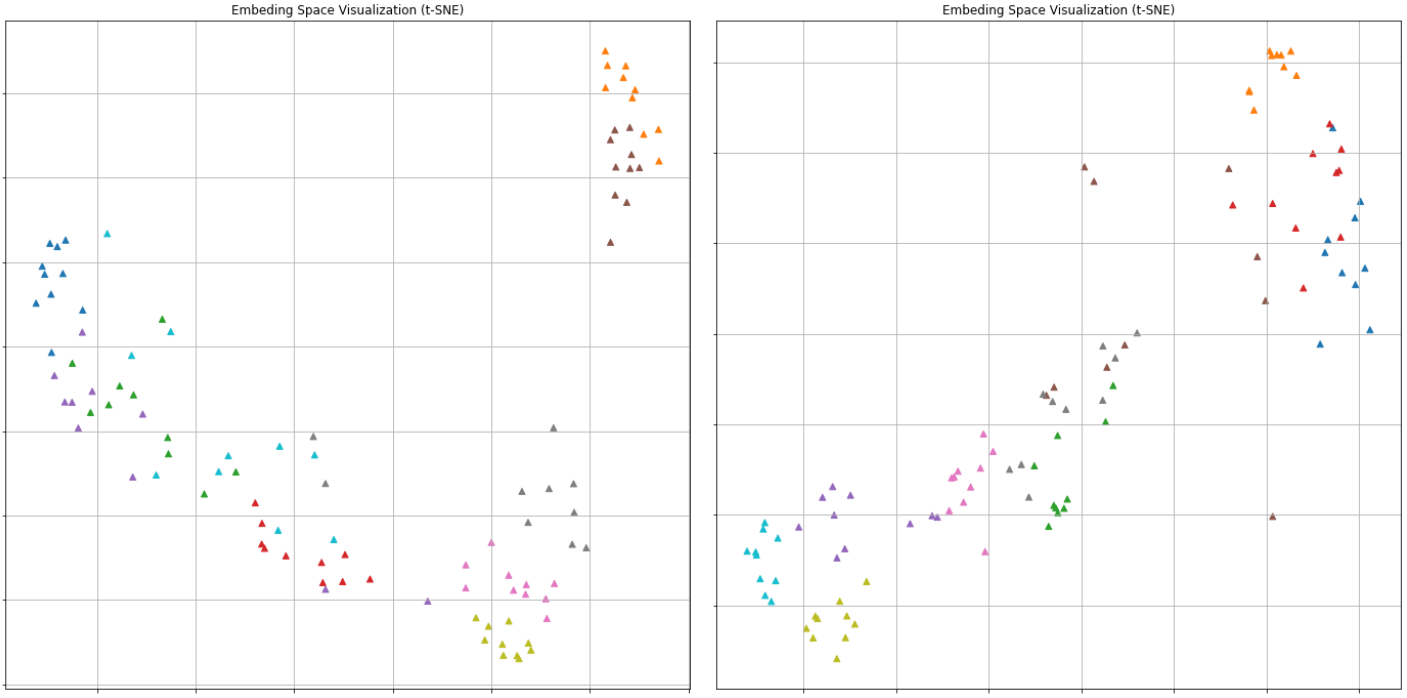

Figure 28. MFCC Audio Experiment 2: Embedding Space Visualization

3.2.3. Experiment 3

- **Hyperparameters**
  - CNN Layers: 64 filters are used in the first block, 128 filters in the second block and 192 filters in the third block
  - Batch Size: 64
  - Embeddings Dimension: 1400
- **Training Phase**

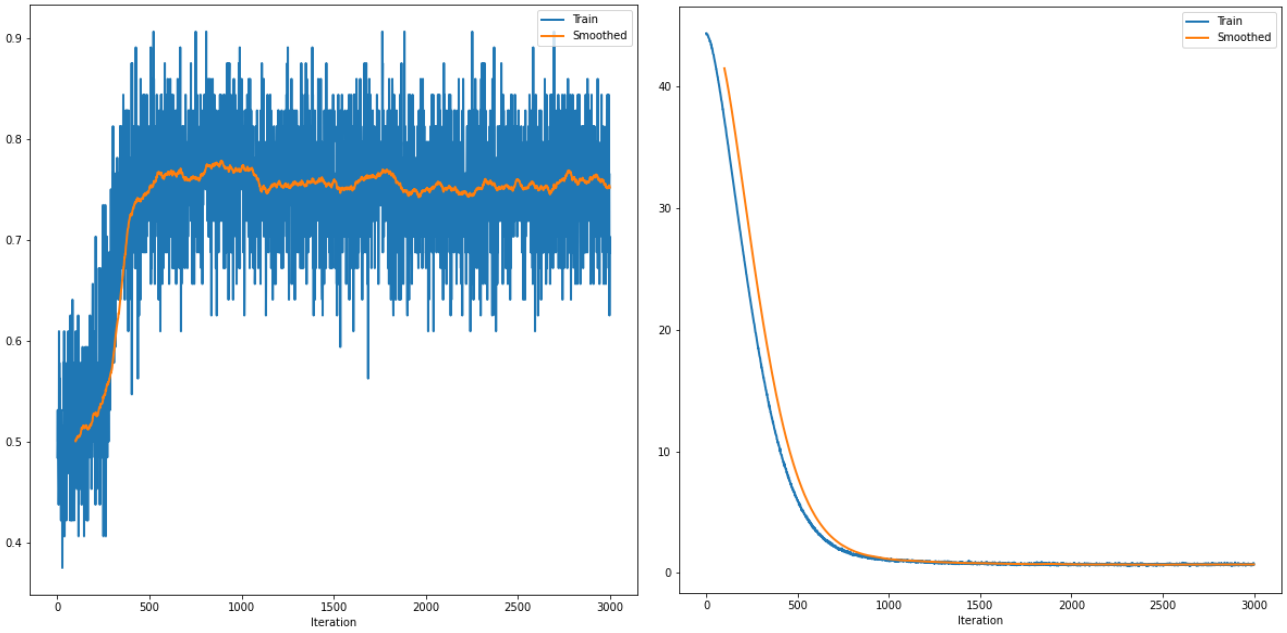

Figure 29. MFCC Audio Experiment 3: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

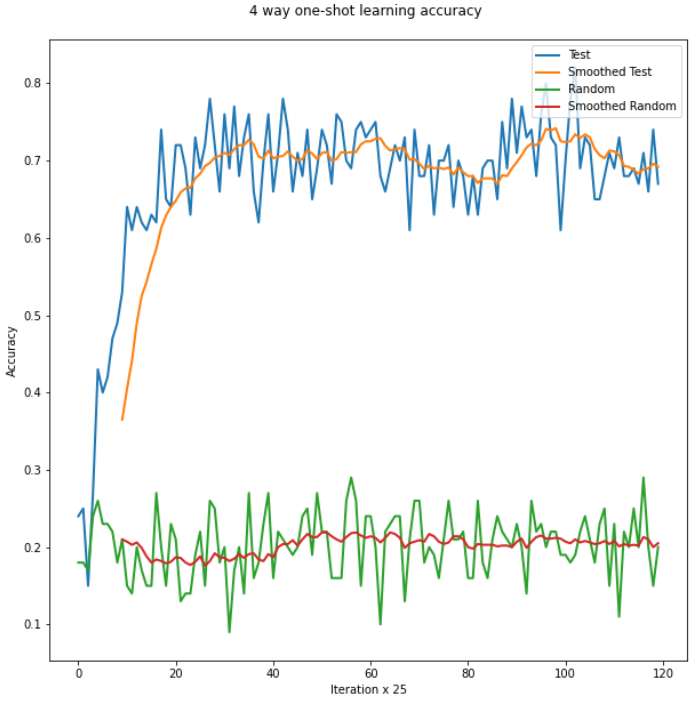

Figure 30. MFCC Audio Experiment 3: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

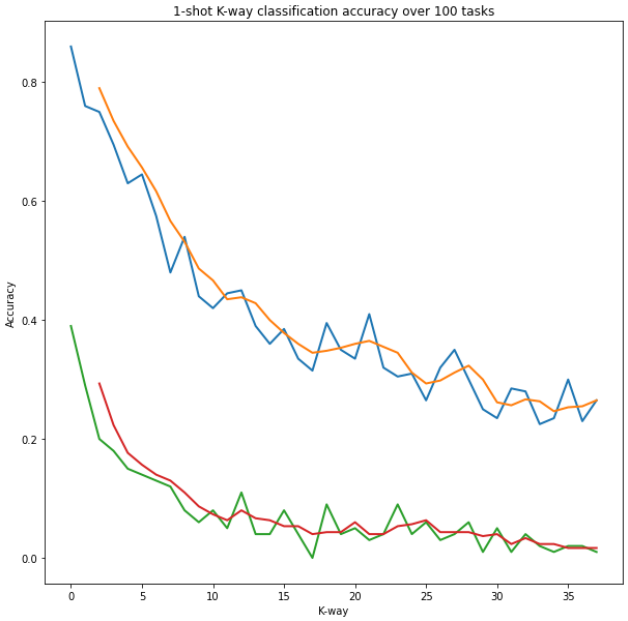

Figure 31. MFCC Audio Experiment 3: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

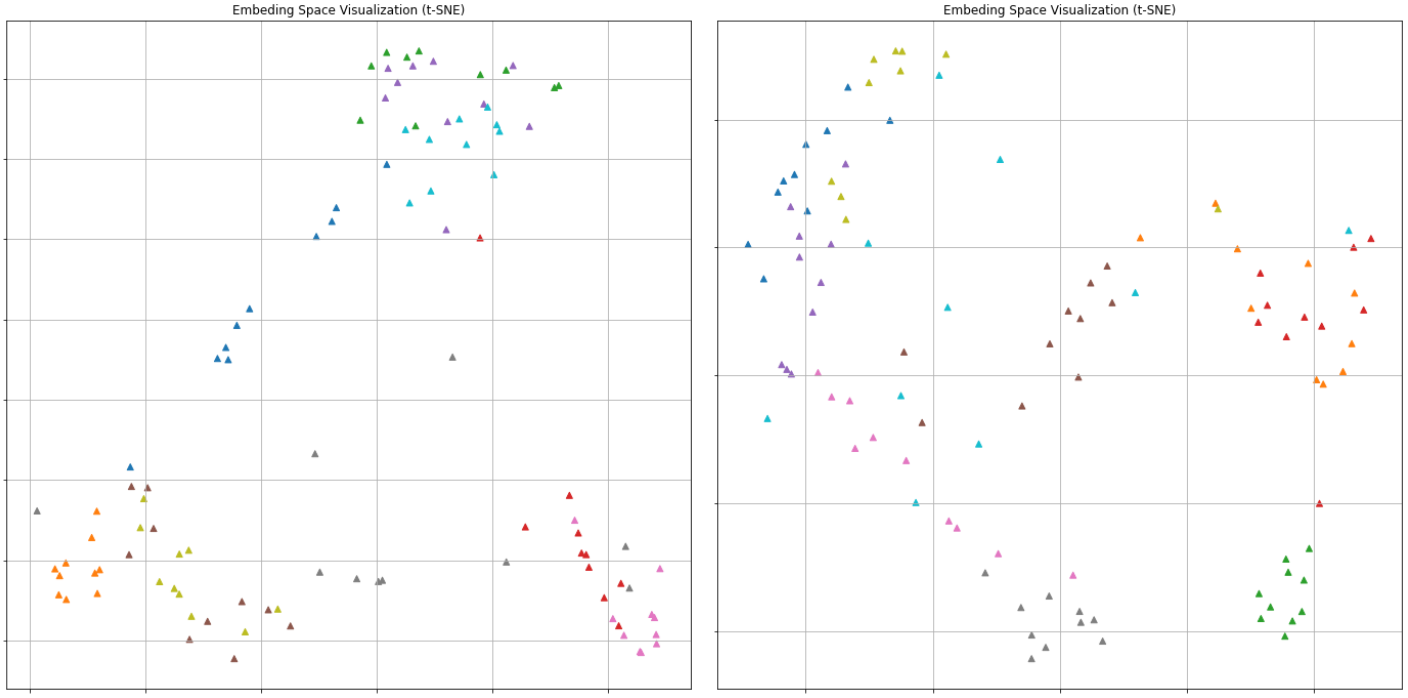

Figure 32. MFCC Audio Experiment 3: Embedding Space Visualization

3.3. VGGish Embeddings

In this collection of experiments **[15]**, the vector embeddings from the VGGish pre-trained network have been used as inputs to our models. To extract the array embeddings I have based on the examples provided by Google and applied it to the 3-second audio fragments **[14]**. I have selected a hop size of 0.1 seconds for the VGGish network inputs, obtaining eventually two-dimensional arrays embeddings of 21x128 components. Figure 33 shows the visualization of an embedding array extracted from the VGGish network, using a hop size of 0.01 seconds (for a better visualization) and an example audio fragment from our dataset as input.

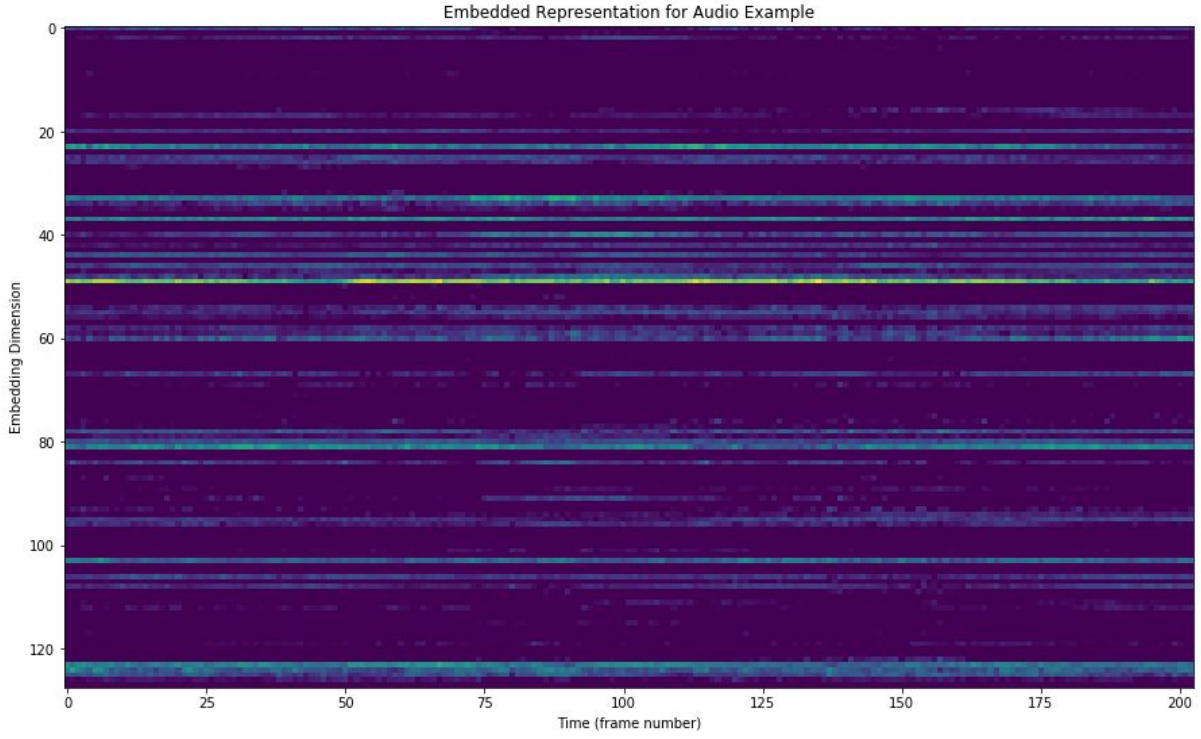

Figure 33. Embedding representation for an audio example (VGGish model)

The general structure of the neural network used is identical to that used in the previous section of experiments with MFCC (Section 3.2). In the following, the experiments conducted with the selected hyperparameters and the results obtained are shown. In the training phase of all the experiments, the RAdam optimizer and the Binary Cross-Entropy loss function were used. The validation phase has been carried out with 1-shot 4-way classification tasks, every 25 training iterations, over 100 tasks. In the embedding space visualization, the t-SNE algorithm has been used to reduce the total dimensions of this space to 2.

3.3.1. Experiment 1

- **Hyperparameters**
  - CNN Layers: 64 filters are used in the first block, 128 filters in the second block and 192 filters in the third block
  - Batch Size: 32
  - Embeddings Dimension: 1024

- **Training Phase**

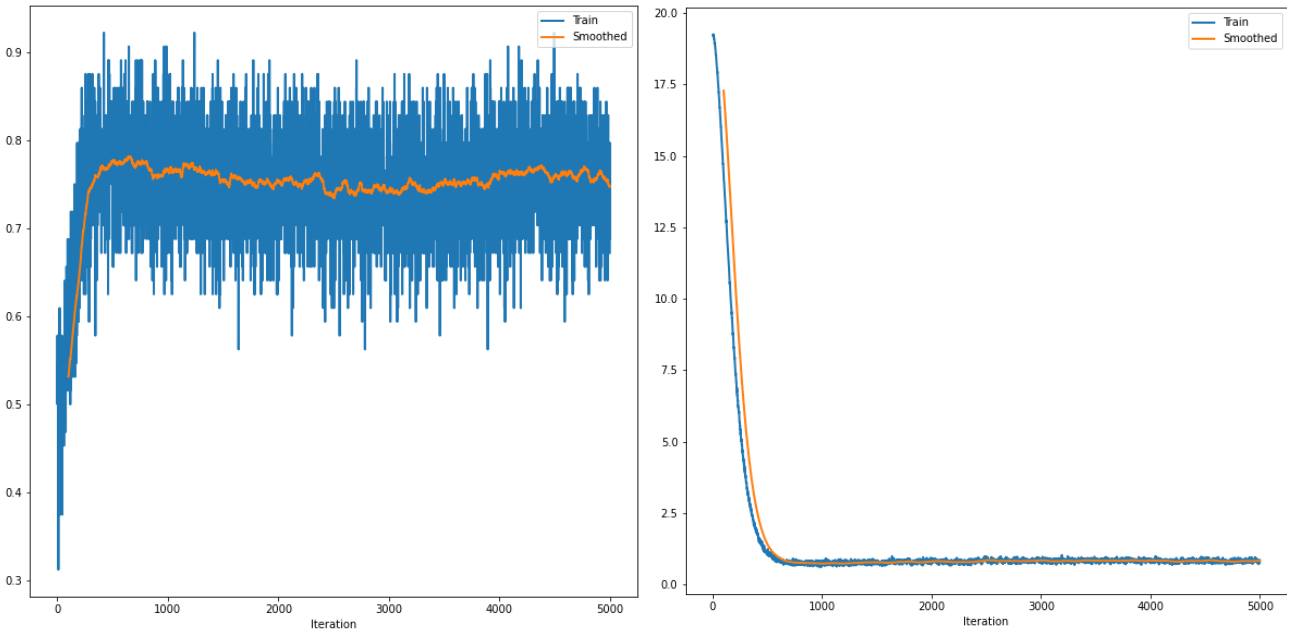

Figure 34. VGGish Embeddings Experiment 1: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

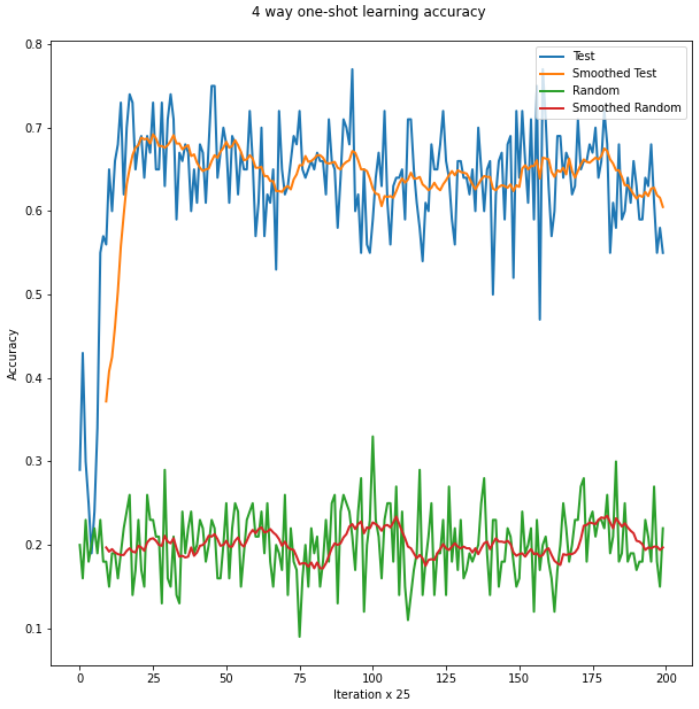

Figure 35. VGGish Embeddings Experiment 1: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

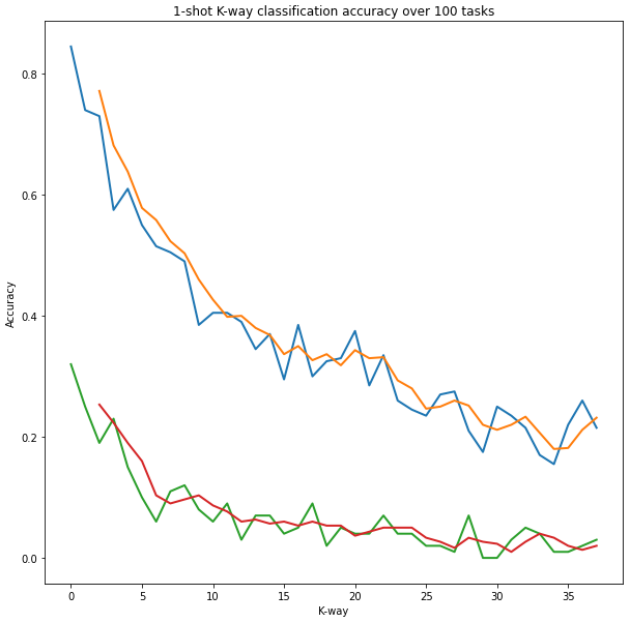

Figure 36. VGGish Embeddings Experiment 1: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

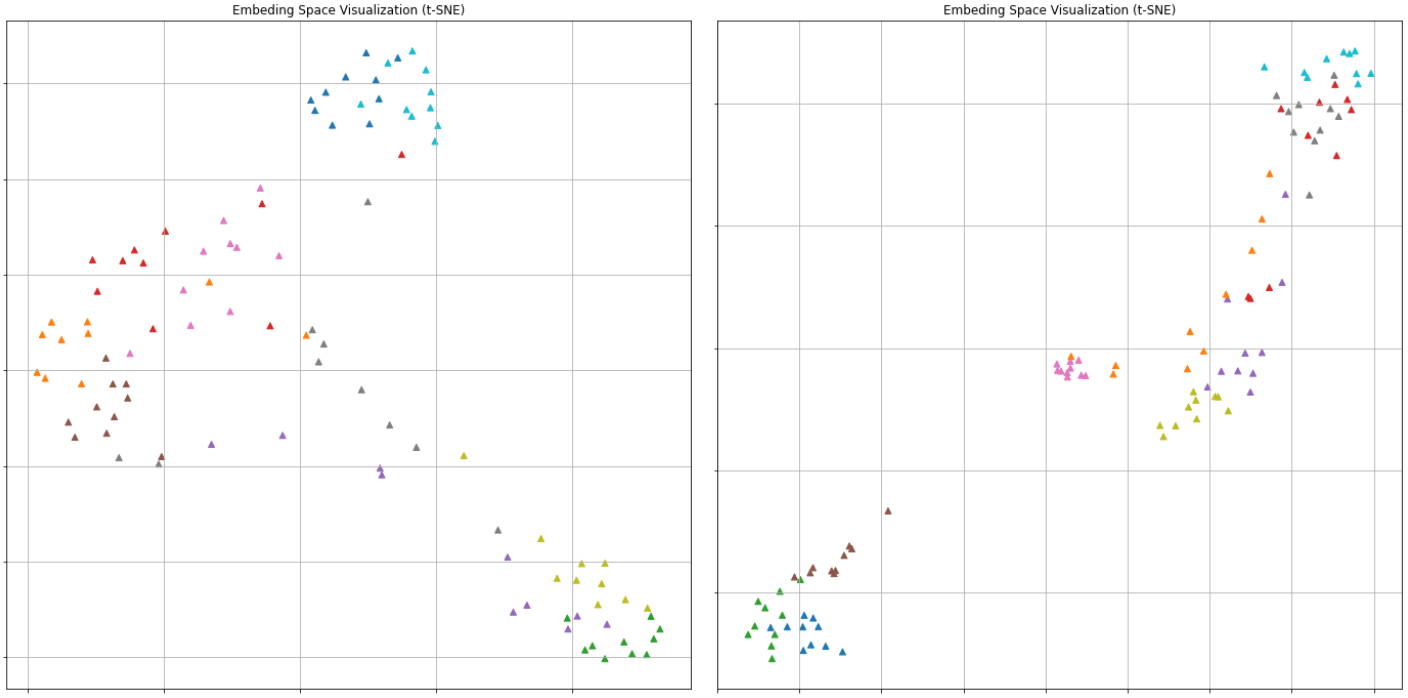

Figure 37. VGGish Embeddings Experiment 1: Embedding Space Visualization

3.3.2. Experiment 2

- **Hyperparameters**
  - CNN Layers: 96 filters are used in the first block, 192 filters in the second block and 288 filters in the third block
  - Batch Size: 96
  - Embeddings Dimension: 1800

- **Training Phase**

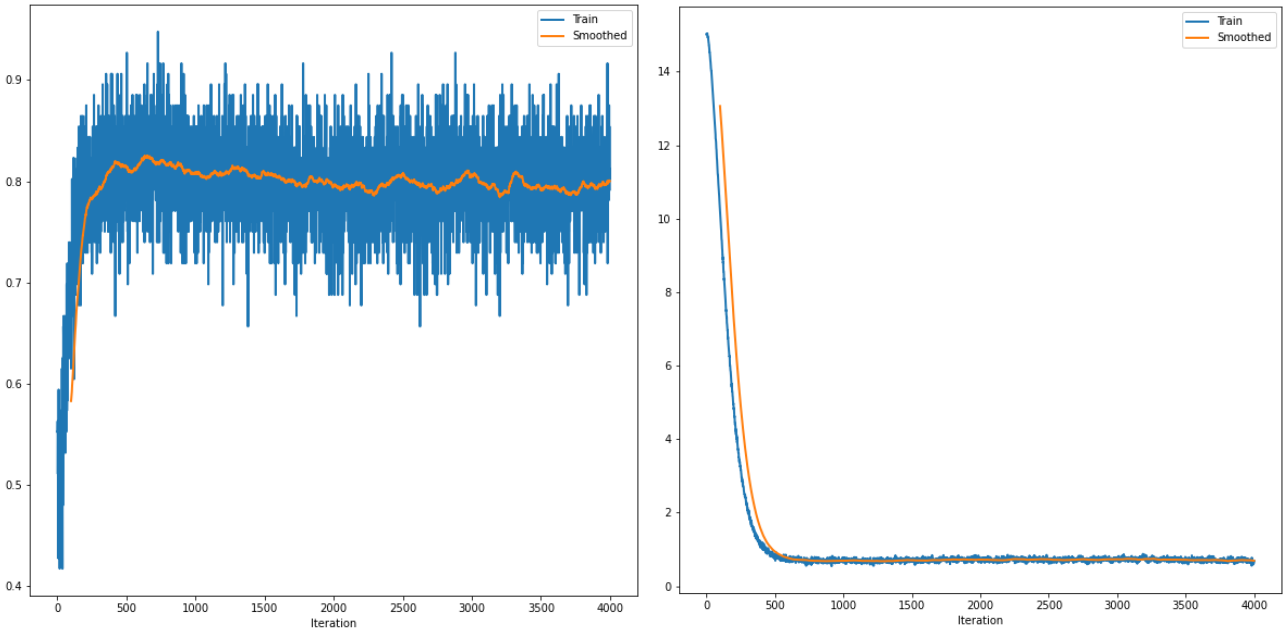

Figure 38. VGGish Embeddings Experiment 2: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

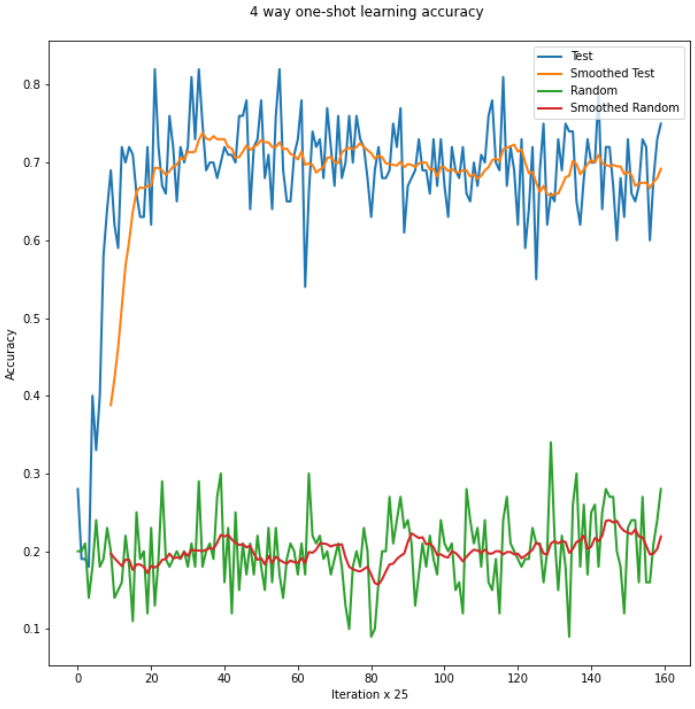

Figure 39. VGGish Embeddings Experiment 2: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

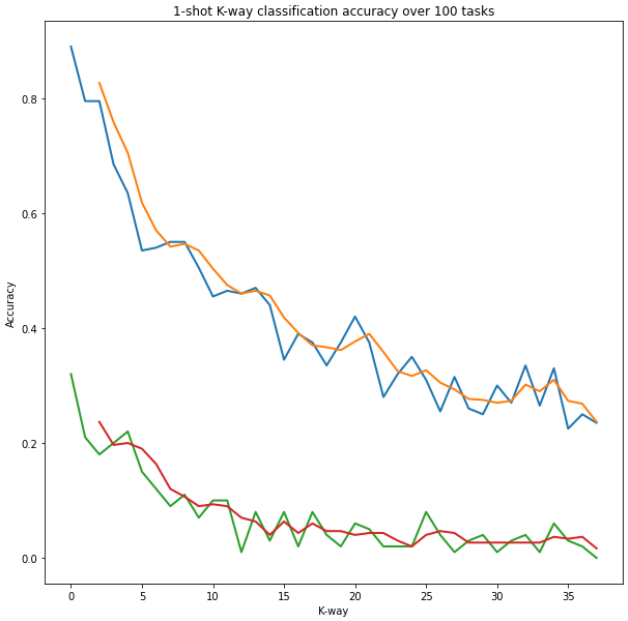

Figure 40. VGGish Embeddings Experiment 2: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

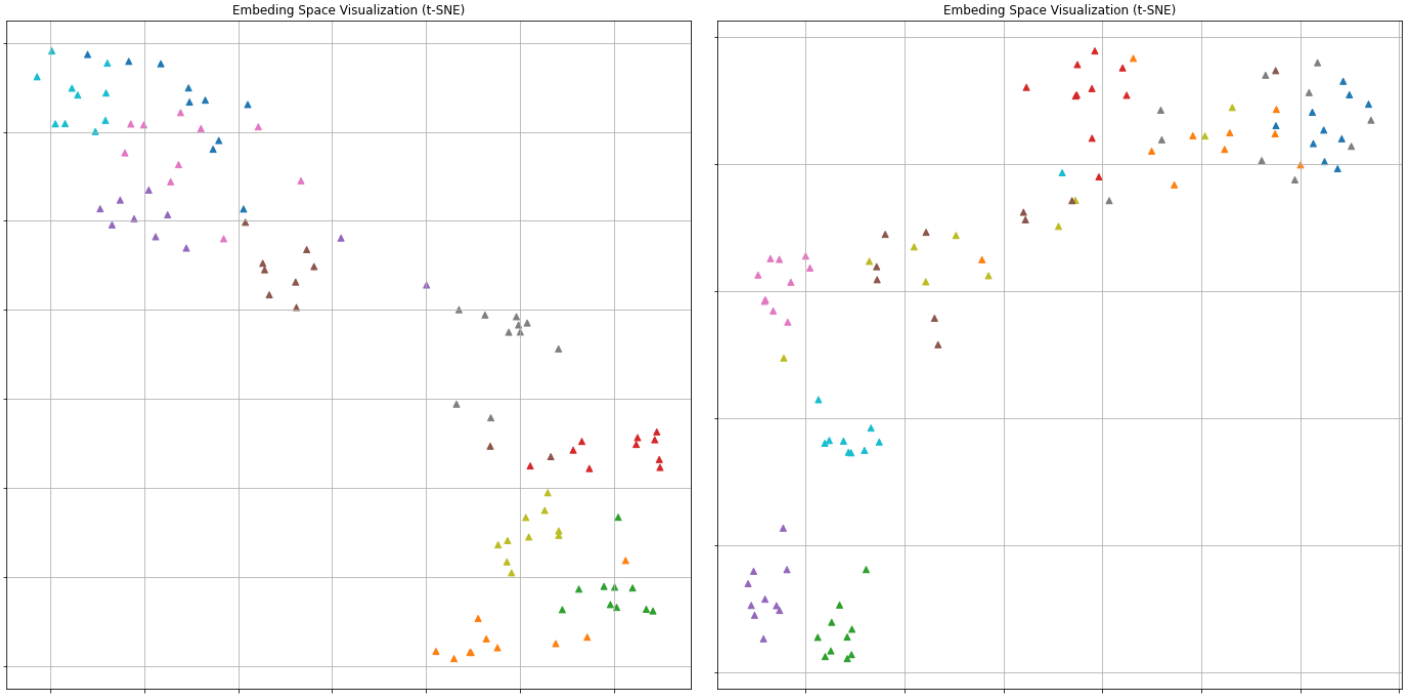

Figure 41. VGGish Embeddings Experiment 2: Embedding Space Visualization

3.3.3. Experiment 3

- **Hyperparameters**
  - Added an extra CNN layer
  - CNN Layers: 64 filters are used in the first block, 128 filters in the second block, 192 filters in the third block and 256 in the fourth block
  - Batch Size: 124
  - Embeddings Dimension: 1500

- **Training Phase**

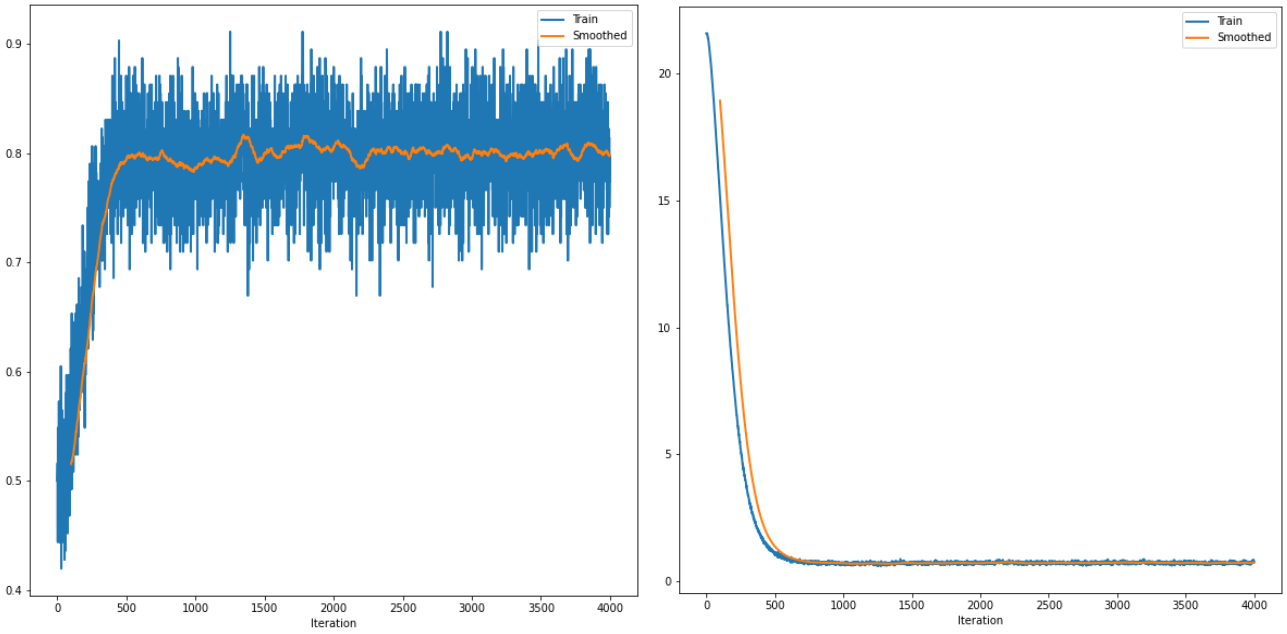

Figure 42. VGGish Embeddings Experiment 3: Train Accuracy and Loss

- **Validation Phase:** 1-shot 4-way classification (100 tasks)

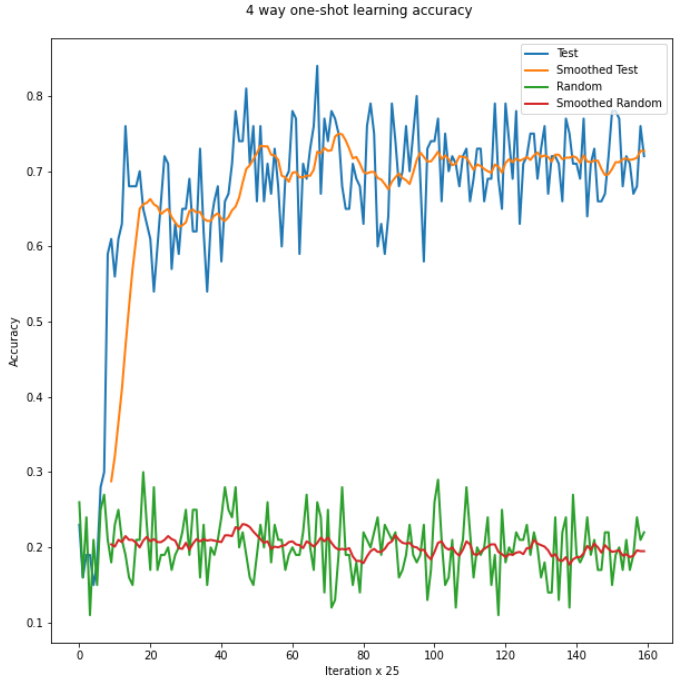

Figure 43. VGGish Embeddings Experiment 3: Validation Accuracy

- **Testing Phase:** 1-shot *k*-way classification (100 tasks per *k*)

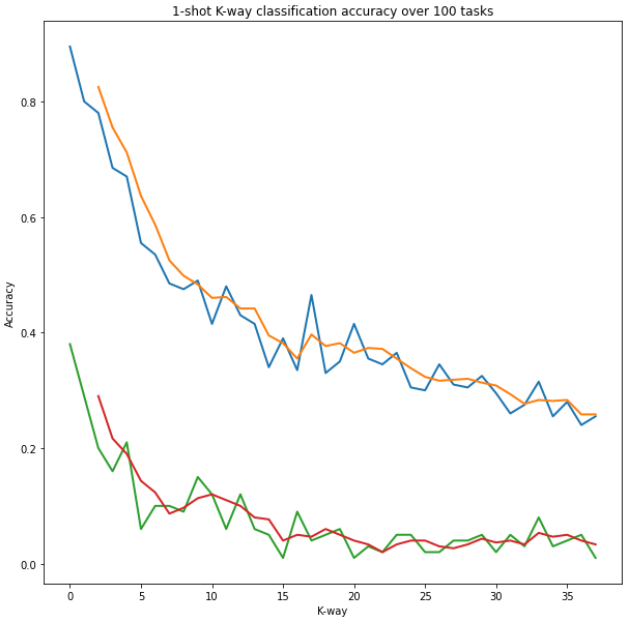

Figure 44. VGGish Embeddings Experiment 3: Testing Phase

- **Embedding Space Visualization:** It is shown the vector embeddings (through t-SNE) obtained from 2 sets of 10 speakers (with 10 random audios each) from the training set.

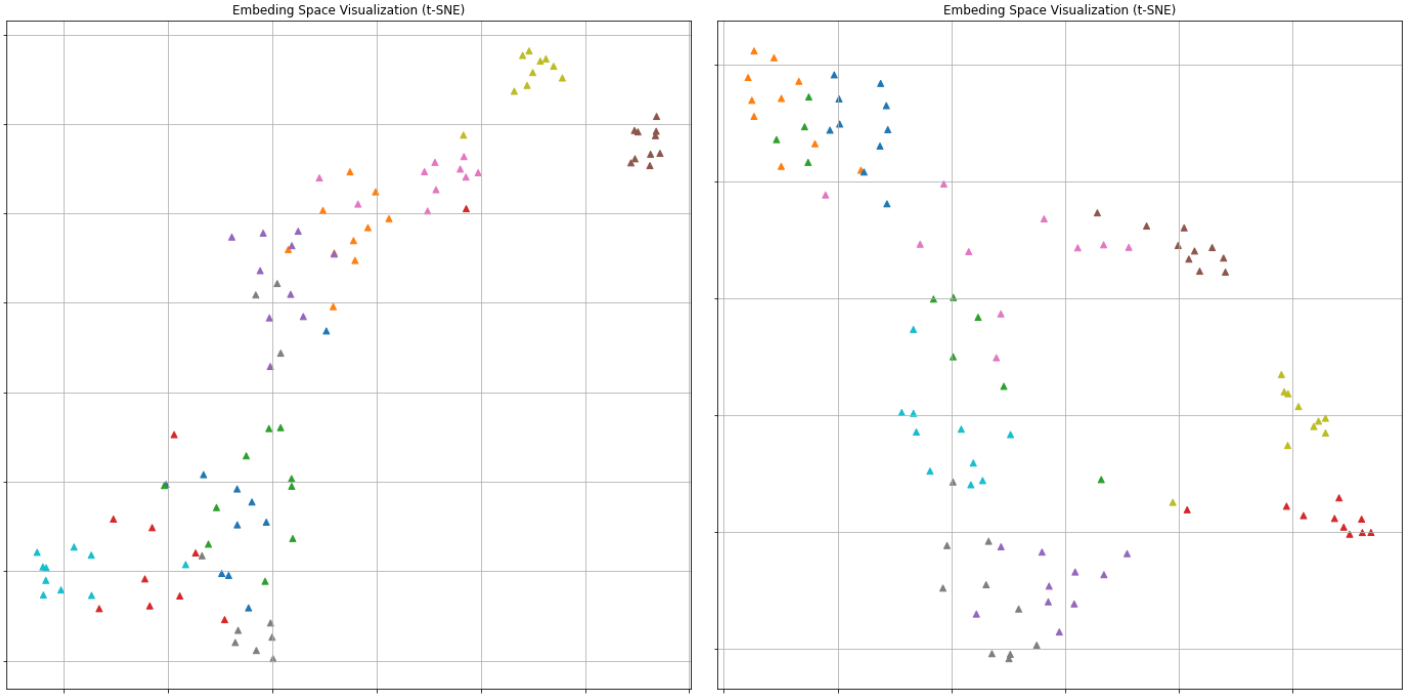

Figure 45. VGGish Embeddings Experiment 3: Embedding Space Visualization

4. Results Discussion and Future Directions

From the experiments carried out in the previous sections, we can draw multiple conclusions. Firstly, we can highlight that **the three types of data representation have obtained quite satisfactory results**, taking into account the limited computational and temporal resources, and having used a relatively small dataset. Bearing this in mind and the **great difference in dimensionality**, especially between raw audio and the other two representations, it indicates that raw audio is indeed a very inefficient way of transmitting information due to its great amount of autocorrelation and redundancy.

The best results have been obtained using MFCC features, ratifying the initial premise of the promising results that can be obtained if we join Deep Learning or Machine Learning techniques that "learn" from real data, with specific and physical knowledge of the problems to be solved. However, the difference between the best results obtained from each data representation used is so small that it could be achieved with an intensive hyperparameter exploration. Therefore, based on the results obtained, a real-world implementation should use that representation whose dimensionality or computing requirements are best suited to the needs of the entire system. Although the raw audio does not seem to indicate a great margin of improvement, and its dimensionality is several orders of magnitude greater than the rest of the representations.

Another aspect worth mentioning is that **the Siamese Network architecture has been able to perform the proposed task quite successfully**, managing to differentiate speakers without taking into account the specific speech. The major drawback in using this architecture is that (as explained in Section 2.3) the network requires paired data, making the number of possible combinations to feed it grow exponentially with the amount of data available. This has a serious impact on the time required to train these models.

In general terms, this project has been very enriching and has helped me to approach the field of audio treatment with machine learning and specific knowledge techniques. In addition, it has helped me to consolidate my knowledge in Python, due to the particular requirements of the architecture used, both in the training pipeline, data management, and model testing.

Regarding the **future directions** that can be taken to continue this project, the first would be an **intensive search for hyperparameters** to obtain the best results that the model can offer (length of audio fragments, subsampling rates, neural structures, extended training times, etc). At the implementation level, it would be necessary to add **different distance metrics used by the Siamese network**, since only the Euclidean distance has been considered. It would also be necessary to implement a solution for the validation or testing phase with *n-shot k-way* tasks, for *n*'s greater than 1.

Also, with regard to the testing phase, it would be very interesting to analyze the performance of the model when, instead of facing a test sample against k samples of unseen classes, the test sample is confronted with the rest of the classes used during the training (or during a fine-tuning phase). Presumably, the performance would be much higher and would be closer to some real case scenarios proposed in the introduction. It would also be of great interest to implement a **better baseline** than the proposed random model, such as a classification based on the k-NN algorithm.

5. References

[1] Acoustical Society of America, 2010, «Physics and Astronomy Classification Scheme,» Available: https://web.archive.org/web/20130514111126/http://www.aip.org/pacs/pacs2010/individuals/pacs2010_regular_edition/reg_acoustics_appendix.htm
[Accessed 10 January 2020]

[2] The Journal of the Acoustical Society of America, 2019, «Machine Learning in Acoustics: Theory and Applications,» Available: https://asa.scitation.org/doi/full/10.1121/1.5133944
[Accessed 17 April 2020]

[3] Wikipedia, «Speaker Recognition,» Available: https://en.wikipedia.org/wiki/Speaker_recognition
[Accessed 17 April 2020]

[4] LibriSpeech Dataset, Available: https://www.openslr.org/12/
[Accessed 5 April 2020]

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah, 1994 «Signature Verification using a "Siamese" Time Delay Neural Network,» Available: https://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf
[Accessed 17 April 2020]

[6] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov, 2015, «Siamese Neural Networks for One-shot Image Recognition,» Available: http://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
[Accessed 17 April 2020]

[7] Pratheeksha Nair, 2018, «The dummy's guide to MFCC,» Available: https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd
[Accessed 28 April 2020]

[8] Tanveer Singh, 2019, «MFCC's Made Easy,» Available: https://medium.com/@tanveer9812/mfccs-made-easy-7ef383006040
[Accessed 28 April 2020]

[9] Daniel Rothmann, 2018, «What's wrong with CNNs and spectrograms for audio processing?,» Available:https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd
[Accessed 28 April 2020]

[10] TensorFlow GitHub, 2019, «VGGish,» Available: https://github.com/tensorflow/models/tree/master/research/audioset/vggish
[Accessed 28 April 2020]

[11] Google Research, 2019, «YouTube-8M Segments Dataset,» Available: https://research.google.com/youtube8m/
[Accessed 28 April 2020]

[12] Google Colab Notebook, «LibriSpeech_Raw_Audio.ipynb,» Available: https://colab.research.google.com/drive/1gPgvsjFQpvx_1GLjGW4C9ZvWJt7VjFKK
[Accessed 29 April 2020]

[13] Google Colab Notebook, «LibriSpeech_MFCCs.ipynb,» Available: https://drive.google.com/open?id=17dUnLY_IS5603FcoJyKL6SpKz_83uQCy [Accessed 29 April 2020]

[14] Google Colab Notebook, «VGGish_Embeddings_LibriSpeech.ipynb,» Available: https://drive.google.com/open?id=12H96lLjt-mX4RT_sCDR0o4i4GYjYVtEv [Accessed 29 April 2020]

[15] Google Colab Notebook, «LibriSpeech_VGGish.ipynb,» Available: https://drive.google.com/open?id=1ed6MOBfcMvUaRb9sD6gZynpOvND7Tx3_ [Accessed 29 April 2020]

[16] GitHub Profile, Jaime Pérez Sánchez, «Speaker_Recognition_Siamese_Networks,» Available: https://github.com/jaimeperezsanchez/Speaker_Recognition_Siamese_Networks [Accessed 29 April 2020]