Master CORO M1

Master in Control and Robotics

SIP M1

Signal and Image Processing

Project Report

June 25, 2020

Unsupervised Exploration Of Large Scale Acoustic Data

Elice Roy Kanjamala

Tarun Teja Yaram

Supervisor

Mathieu LAGRANGE

Cr2 CNRS Researcher

Abstract

A large set of acoustic sensors were deployed in a city as part of French Project CENSE. This was aimed for continuously gathering information about the acoustic environment of the city. A dataset of 100 acoustic scene recordings were gathered with different classes of environment like restaurant, supermarket, office, street etc. We considered several machine learning tools such as deep embeddings, summary statistics of short-term audio descriptors etc. to build the project. One approach is the MFCC approach that is used to calculate the audio similarity measurement which models a recording using summary statistics of short-term audio descriptors. The other is Look, Listen, and Learn (L^3 -Net) approach which is an embedding approach trained through self-supervised learning of audio-visual correspondence requiring labeled data. Finally, we aim to compare the computational complexity and performances with the different approaches like RbQ-c Approximate, RbQ-c Complete and Early Integration which finds the similarity measurement for the acoustic scenes.

Contents

Abstract i

List of Figures iii

1 Introduction 1

2 Computational Methods 3
2.1 Mel-Frequency Cepstral Coefficients (MFCCs) . . . 3
2.2 L^3-NET . . . 4
2.3 Libraries . . . 5
2.4 Early Integration . . . 6
2.5 RbQ-c Complete . . . 6
2.6 RbQ-c Approximate . . . 7

3 Experiment 9
3.1 Dataset . . . 9
3.2 Loading the dataset and Computation of Distance Matrix . . . 9
3.2.1 Distance Matrix for MFCC . . . 9
3.2.2 Distance Matrix for L^3-Net . . . 10

4 Results 12
4.1 Using Mel-Frequency Cepstral Coefficients (MFCCs) . . . 12
4.2 Using OpenL3 . . . 13
4.2.1 Using an embedding of size 512 . . . 13
4.2.2 Using an embedding of size 6144 . . . 15

5 Conclusion 19

Bibliography 20

List of Figures

2.1 High-Level Architecture of L^3- Net 4

4.1 Early Integration and RbQ-c Complete using MFCC 12
4.2 RbQ-c Approximate using MFCC 13
4.3 Early Integration and RbQ-c Complete using Open L3 14
4.4 RbQ-c Approximate using Open L3 15
4.5 Early Integration and RbQ-c Complete using Open L3 and with an embedding size of 6144 16
4.6 RbQ-c Approximate using Open L3 and an embedding size of 6144 . 16

Chapter 1

Introduction

A large set of acoustic sensors were deployed in the city of Lorient as part of a French Project CENSE. The acoustic monitoring aimed at gathering rich information about the environment by some network of sensors. These networks gather large amounts of data requiring analysis. This continuously evolving data is further analysed to identify the repeating patterns, anomalous and isolated events by finding the similarity measurement for the acoustic scenes.

From the works of Lostanlen et al. [6], we understand the concepts of mel-frequency cepstral coefficients (MFCCs) that is used to model a recording using summary statistics of short-term audio descriptors. Like the works in this paper, we study the similarity between two scenes using 100 acoustic scene recordings. The dataset is divided into 10 acoustic scene classes namely *bus, busy street, office, open air market, park, quiet street, restaurant, supermarket, tube, and tube station* and each class contains 10 audio recordings. Unlike in the paper, we compare the performance of similarity measures using RbQ-c Approximate, RbQ-c Complete and Early integration with this dataset. We calculate precision at rank k (p@k) for each method which is computed by taking a query item and counting the number of items of the same class within the k closest neighbors, and then averaging over all query items. Finally, we find a similarity represenation method that suits best for the MFCC approach.

Also, from the works of Cramer et al. we understand the Look, Listen, and Learn (L^3-Net) concept which is an embedding that is trained through self-supervised learning requiring labeled data [4]. Generally, we train it on a subset of AudioSet using self-supervision by exploiting the correspondence between sound and visual objects in video data. In this paper, we mainly learn how to learn deep audio embedding's from large audio collections and use them to train the shallow classifiers with small labeled datasets. We use Mel-frequency log-magnitude spectrograms as input to deep convolutional networks for training, as we find that Mel

spectrograms are designed to capture perceptually relevant information more efficiently with less frequency bands compared to a linear spectrogram and also because Melspectrograms helps in better generalization. We make use of the OpenL3 Library to implement this. As we tested for MFCC approach, we also carry out the comparison of performance of similarity measures using RbQ-c Approximate, RbQ-c Complete and Early integration using the same dataset. We find the best similarity representation method using OpenL3 approach also.

In our project we finally perform the comparison of the performance with MFCC and L^3-NET. We compare the computation time, best representation of similarity scenes etc. The compilation of different computational methods used in the project is explained in the Chapter 2. The first part of this section focuses on acoustic scene similarity retrieval task using MFCCs while the second part focuses on L^3-NET using OpenL3. The results of the experiments is plotted and explained in Chapter 3.

Chapter 2

Computational Methods

2.1 Mel-Frequency Cepstral Coefficients (MFCCs)

A Mel-frequency cepstrum (MFC) is a representation of a short-term equally spaced power spectrum of sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. MFCCs are computed as coefficients from the Fourier transform along the frequency of the signal's log spectrogram. By extracting MFCCs on every successive short-time window of a signal, it is transformed into a time-series of feature points in an Euclidean space [5].

The MFCC pattern is usually computed within a short time window, where the audio signal stays rather constant. Mel-frequency spectrograms are obtained by averaging spectrogram values over mel-frequency bands. It improves stability to time-warping, but it also removes information. Over time intervals larger than 25 ms, the information loss becomes too important, which is why mel-frequency spectrograms and MFCCs are limited to such short time intervals[1].

MFCCs are commonly used as features in speech recognition systems, such as the systems which can automatically recognize numbers spoken into a telephone. They have recently found wider use in music information retrieval [3] and environmental audio processing [5]. MFCC values are not very robust in the presence of additive noise, and so it is common to normalise their values in speech recognition systems to lessen the influence of noise. Some researchers propose modifications to the basic MFCC algorithm to improve robustness, such as by raising the log-mel-amplitudes to a suitable power (around 2 or 3) before taking the DCT (Discrete Cosine Transform), which reduces the influence of low-energy components.

2.2 L^3-NET

Look, Listen, and Learn (L^3-Net) is an open-source deep audio embedding which trained through self-supervised learning of audio-visual correspondence in videos as opposed to other embeddings requiring labeled data [4]. This framework has the potential to produce powerful embeddings for downstream audio classification of tasks, but has a number of unexplained design choices that may impact the embeddings' behavior. We try to implement this using OpenL3 which is an open-source Python library for computing deep audio and image embeddings.

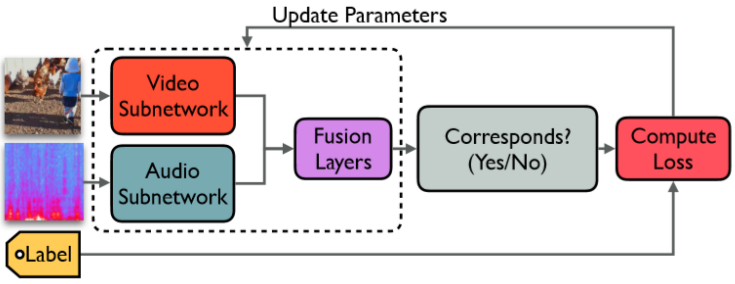

FIGURE 2.1: High-Level Architecture of L^3- Net

The L3-Net approach [2] proposes a means of learning embeddings via the auxiliary task of audio-visual correspondence (AVC) which aims to determine whether a video image frame and a 1 s audio segment come from the same video and overlap in time [4]. Figure 2.1 shows the architecture of the L^3-Net. We study that the L^3-Net is divided into 3 main distinctive parts namely, the $video\ subnetwork$, the $audio\ subnetworks$ and the $fusion\ layers$.

The video and the audio subnetworks helps to extract the visual and audio features respectively, and the fusion layers helps to predict correspondence. Since both matched and mismatched image-audio pairs can be generated automatically from the training data, it requires no manual labeling to train the model on this binary classification task. The audio and vision subnetworks use four blocks of convolutional and max-pooling layers, the outputs of which are flattened, concatenated, and passed to the fully-connected fusion layers to produce the correspondence probability. The audio embedding is obtained from the final output layer of the audio subnetwork, replacing the layer's non-linear activation with a max-pooling layer, the output of which is flattened leading to a final embedding dimensionality of 6144.

2.3 Libraries

The most important libraries to be imported for the first part, the MFCC part are the *os* module, the *numpy* , the *matplotlib.pyplot, glob* and the *librosa* library. In the second part, the OpenL3 part, we import two additional libraries *openl3* library and *soundfile* package.

The *OS* module in python provides functions for interacting with the operating system. This module provides a portable way of using operating system dependent functionalities.

$NumPy$ is an array-processing package in python. It aims to provide an array object that is up to 50x faster that traditional Python lists. The array object in NumPy is called $ndarray$, which helps to provide a lot of supporting functions that make working with $ndarray$ very easily.

$Matplotlib.pyplot$ is a collection of command style functions that make matplotlib work like MATLAB. Each $pyplot$ function makes some change to a figure like it can create a figure, create a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes.

The $glob$ module finds all the pathnames matching a specified pattern. It is used to access the audio data from its files.It is useful in any situation where the program needs to look for a list of files on the filesystem with names matching a specific pattern i.e. If we need a list of filenames that all have a certain extension, prefix, or any common string in the middle, we use glob module to do it.

$Librosa$ is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. Since we are dealing with audio data, this is an important package. Librosa helps to visualize the audio signals and also do the feature extractions in it using different signal processing techniques. The MFCC implemented in the code was with help of librosa package.

$SoundFile$ is an audio library based on libsndfile, CFFI and NumPy. SoundFile

can read and write sound files. File reading/writing is supported through libsnd-file, which is a free, cross-platform, open-source (LGPL) library for reading and writing many different sampled sound file formats that runs on many platforms including Windows, OS X, and Unix. In our project SoundFile is used to represent audio data as NumPy arrays. The 100 audio data files is read using this library. We read each audio file using the $soundfile.read()$ and it returns a two-dimensional (frames x channels) NumPy array.

2.4 Early Integration

In early integration, we average over time the feature set of each scene, resulting in one feature vector per scene. The distance on this average feature vector is then used to determine the metric $p@k$ which is computed by taking a query item and counting the number of items of the same class within the k closest neighbors, and then averaging over all query items. In simple words, it is the euclidean distance between the averaging of all the features. It can be computed using the formula:

$$Distance = X_i - X_j \tag{2.1}$$

where X_i , X_j $\in$X$_d$ and $X_d = \frac{\Sigma X_d}{n}$

The computational complexity of Early Integration is $\mathcal{O}(n)$.

2.5 RbQ-c Complete

Relevance-based Quantization Closest Similarity (RbQ-c) Complete is defined as:

$$Distance(n) = Complete(X_i(n), X_j(n), Size\ of\ Window) \tag{2.2}$$

where $\forall n \in [1, number\ of\ windows]$
and $i \in [1, Size\ of\ Audio\ Data]$
and $j \in [i, Size\ of\ Audio\ Data]$

The Complete() is an iterative function. It calculates the distance between the averaged datas x and y and compares with the previous value of the input data to get the minimun distance as:

$$Complete(X_i(n), X_j(n), Size\ of\ Window) = \min_{x,y \in \mathbb{M}} (y - x) \tag{2.3}$$

where $\mathbb{M} = \frac{\sum_{j=i}^{i*Size\ of\ Window} m_{*,j}}{i*Size\ of\ Window}$

Here, we take the norm by considering a time slice of averaged data x and compare it with all the values in the averaged data y. Therefore, the complexity of the computation is $\mathcal{O}(n^2)$.

2.6 RbQ-c Approximate

Relevance-based Quantization Closest Similarity (RbQ-c) Approximate is defined by:

$$Distance(n) = Approximate(X_i(n), X_j(n), Size\ of\ Window) \tag{2.4}$$

where $\forall n \in [1, number\ of\ windows]$
and $i \in [1, Size\ of\ Audio\ Data]$
and $j \in [i, Size\ of\ Audio\ Data]$

The Approximate() calculates the min of the norm of averaged data x and y :

$$Approximate(X_i(n), X_j(n), Size\ of\ Window) = \min(x(k) - y(k)) \tag{2.5}$$

where time k is defined $\forall k \in [1, \frac{Column\ Size\ of\ Data}{Size\ of\ Window}]$

and averaged melspectrum data $x, y \in m$ and $m = \frac{\sum_{j=i}^{i*Size\ of\ Window} M_{*,j}}{i*Size\ of\ Window}$

Here, we compute the minimum distance between two centroids in a time synchronous manner. Therefore, the complexity of this computation is $\mathcal{O}(n)$.

We apply the different scene similarity measurement approaches discussed in Section 2.6, 2.5 and 2.4 for both MFCC method and OpenL3 embedding method and compare their outputs. We find the best suitable approach for the two methods.

Chapter 3

Experiment

3.1 Dataset

The project is carried out on DCASE 2013 dataset which is the publicly available[7]. We understand that the dataset was initially constructed for assigning the classof a given recording for the task of acoustic scene classification. Our dataset con-sists of a public subset, each made up of 100 acoustic scene recordings and evenlydivided into 10 acoustic scene classes:bus, busy street, office, open air market,park, quiet street, restaurant, supermarket, tube, and tube station. The recordingswere made by three different recordists at a wide variety of locations in the Greater-London area over a period of several months. In order to avoid any correlation be-tween recording conditions and label distribution, all recordings were carried outunder moderate weather conditions, at varying times of day and week, and eachrecordist recorded each scene type [6].

3.2 Loading the dataset and Computation of Distance Matrix

In the preprocessing step, we load the 100 audio files. Then, to compute the similarity between two acoustic scenes, we make use of a distance matrix which is a symmetric matrix. Let us see how the computation of the similarity between two scenes is done for the two main approaches we do.

3.2.1 Distance Matrix for MFCC

After importing the essential libraries, we provide the path where the audio file is stored like path of the Google Drive in our case. We make use of the $glob$ module to store the path of each audio data.

Then the most important part in the preprocessing step is the loading of the audio and converting it into MFCCs. When we load the data using the $load$ module in $librosa$ module, it returns the audio time series and its sampling rate. Also, we make use of the mfcc features in $librosa$ library to convert the loaded audio time series to generates the mel-frequency cepstral coefficients (MFCC). We provide the value of n_mfcc which returns the number of MFCCs to be returned. In our case it was 40. Therefore, for each data we obtain an array of dimension 40 x 1292 where 40 is the number of features and 1292 is the number of frames (time).

When compared to Open-L3, the computation of this data is faster and easier. Then we use this $data$ to compute the early integration , the RbQ-c Complete and the RbQ-c Approximate using the formulas mentioned in the Chapter 2 to obtain the distance matrix for each.

3.2.2 Distance Matrix for L^3-Net

The major change in the OpenL3 part when compared to MFCC is the loading of data. We make use of $soundfile$ library instead of the $librosa$ library. We use the $read()$ in the soundfile module to load the data. Ths function returns 2 values: the data and the sampling rate. We use this returned values as the input to OpenL3 library function $get\_audio\_embedding()$ with other essential parameters. These essential parameters contains the content type, the input representation, embedding size and hop size. These parameters require proper tuning else it would take the default values.

We use the content time as env since our audio files are from the environment. The input representation is chosen to be $mel256$ since we learnt that 256-bin variant performs the best when compared to $mel128$ and $linear$ modes. Open-L3 generally consists of two type of embedding's: 512 and 6144. We test for both these. Next, we tune the hop size of openl3 to get approximately the same number of frames as in MFCC.

After proper tuning of all the parameters and then performing the data embedding for the given audio files we obtain the dimension as 1296 x 6144 and 1296 x 512 for the two embedding sizes. Since we need the number of frames to be same for each data, we take the transpose so the the dimension is 6144 x 1296 and 512 x 1296 respectively.

The python snippet to run the transpose of the obtained data dimension is as follows:

```
data_emb1.append(np.transpose(data_emb[i]))
```
(3.1)

Then we use this $data\_emb$ to compute the early integration , the RbQ-c Complete and the RbQ-c Approximate using the formulas mentioned in the Chapter 2 to obtain the distance matrix for each and find the best suitable among these to compute the similarity between acoustic scenes.

After carrying out the experiment we see that Early Integration is most obvious, easiest and efficient way. RbQ-c Approximate is little more complex but it has the same computational complexity as in Early Integration i.e. $\mathcal{O}(n)$. But when it comes to RbQ-c Complete, we see that the computation is more complicated ie. it has complexity of $\mathcal{O}(n^2)$. So let us find if it is worth to use RbQ-c Complete for MFCC and L^3-Net.

Chapter 4

Results

4.1 Using Mel-Frequency Cepstral Coefficients (MFCCs)

MFCCs are computed for windows of 50ms and hops of 25ms with full frequency range. We obtain a dimension of dataset as 40 x 1292 since the number of coefficients to be returned by MFCC is set as 40. To evaluate the different representations introduced in the Chapter 2, we apply it to dataset and compare the results.

The plot for Early Integration and RbQ-c Complete is shown in Figure 4.1

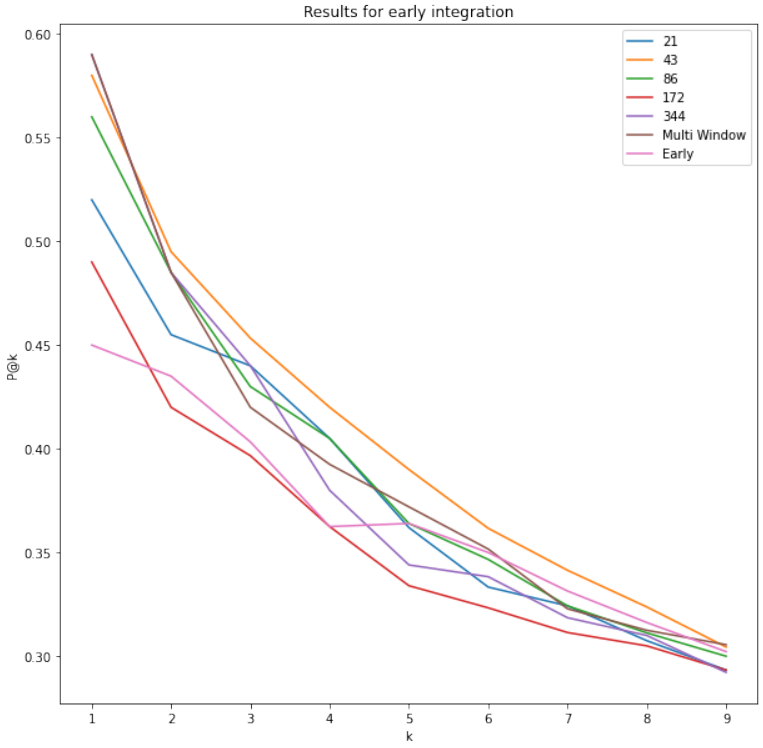

FIGURE 4.1: Early Integration and RbQ-c Complete using MFCC

The plot for RbQ-c Approximate is shown in Figure 4.2.

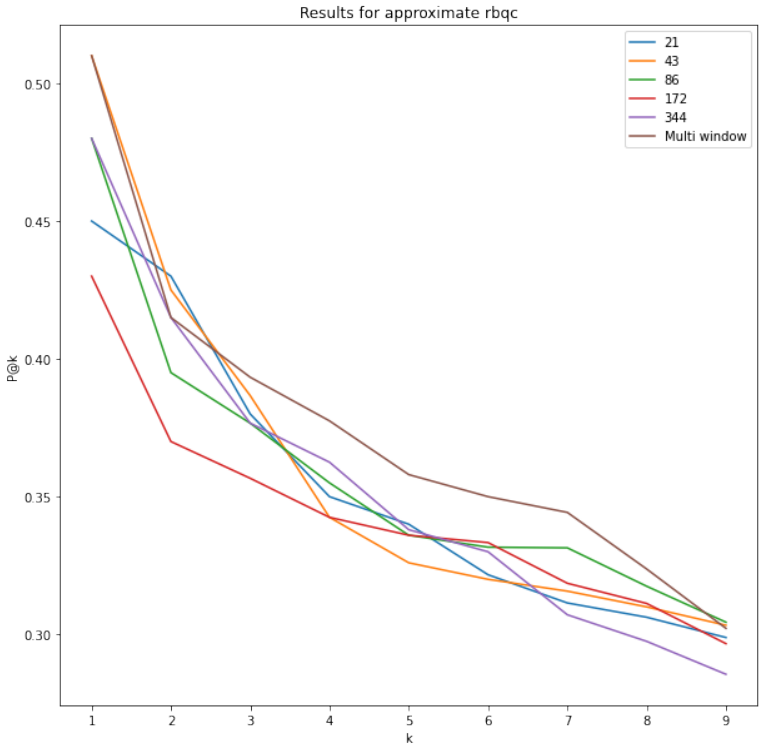

FIGURE 4.2: RbQ-c Approximate using MFCC

We observe that both RbQ-c Approximate and RbQ-c Complete outperform Early Integration method. When we compare RbQ-c Approximate with RbQ-c Complete, we observe that RbQ-c Complete is time asynchronous and performs well with small window sizes but we must compromise the computational complexity, whereas, RbQ-c Approximate is synchronous and has less computational complexity but requires larger window size for beter performance. This helps to conclude that using a Relevance-Based quantization (RbQ) method helps to improve the similarity measures between the scenes in MFCC.

4.2 Using OpenL3

4.2.1 Using an embedding of size 512

We perform the L3 embedding in place of MFCC's in order check whether we obtain better results with this embedding type or not. In the L^3-Net method, we

should import the openL3 library which is an open source and can be accessed anywhere.

We used M256 since we found from [4] that M256 outperforms the linear spectrograms and M128. M128 still performs better than the Linear variant, suggesting that Mel bins indeed capture relevant information in the audio signal more efficiently.

To compare with MFCC, we tune the hop size and take the transpose of the obtained matrix to get approximately the same number of frames as in MFCC. We obtain a hop size of 0.022799 in order to achieve the similar dimension. The resultant matrix was of dimension 512 x 1296.

The plot for Early Integration and RbQ-c Complete is shown in Figure 4.3

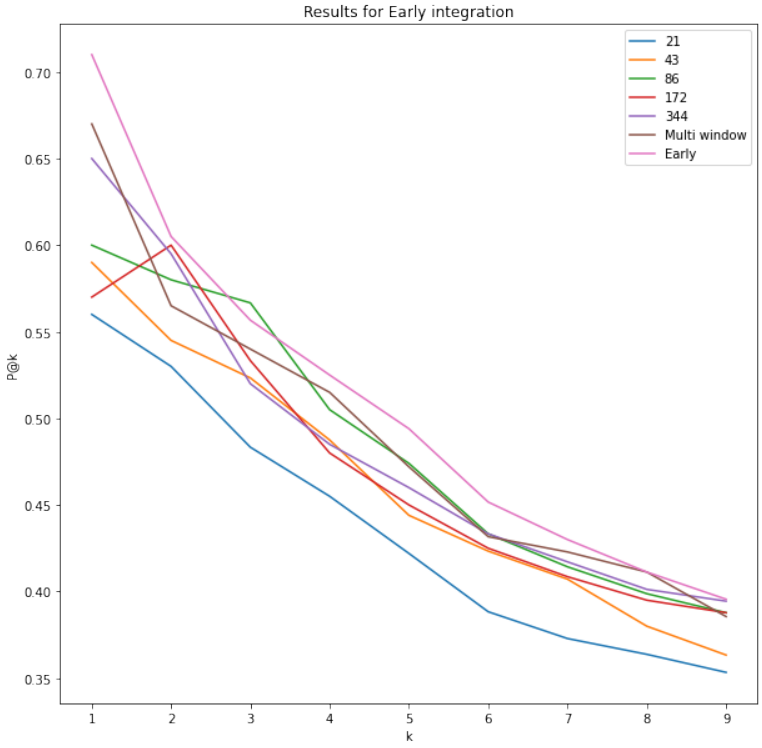

FIGURE 4.3: Early Integration and RbQ-c Complete using Open L3

The plot for RbQ-c Approximate is shown in Figure 4.4.

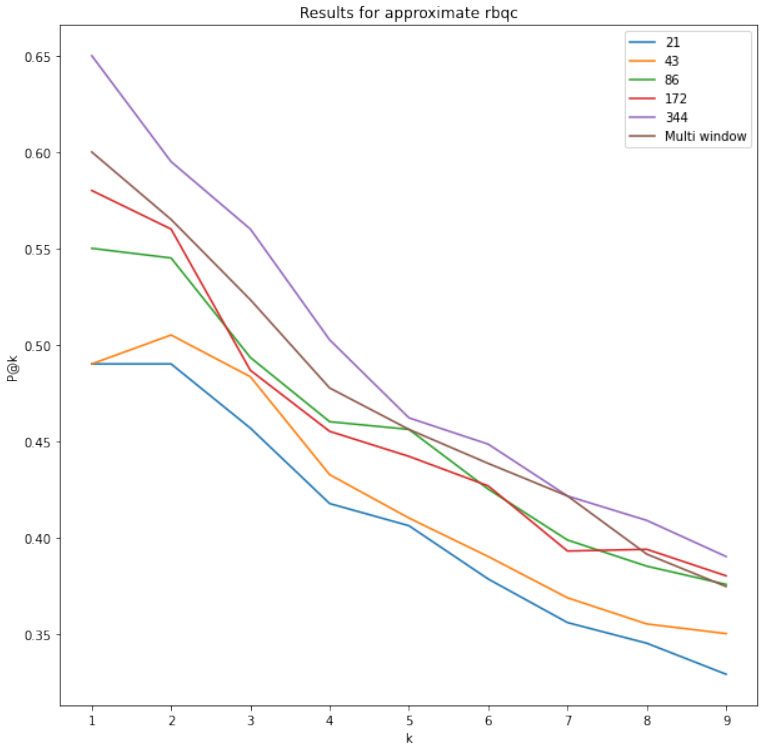

FIGURE 4.4: RbQ-c Approximate using Open L3

We observe that Early integration outperform both RbQ-c Approximate and RbQ-c Complete in case of OpenL3. When we compare RbQ-c Approximate with RbQ-c Complete, we observe that RbQ-c Complete is time asynchronous but we must compromise the computational complexity, whereas, RbQ-c Approximate is synchronous and has less computational complexity.

4.2.2 Using an embedding of size 6144

In the same way as in 4.2.1, we considered the embedding size as 6144 with same hop size of 0.022799 as run the program. Here, the resultant matrix was of dimension 6144 x 1296.

The plot for Early Integration and RbQ-c Complete is shown in Figure 4.5

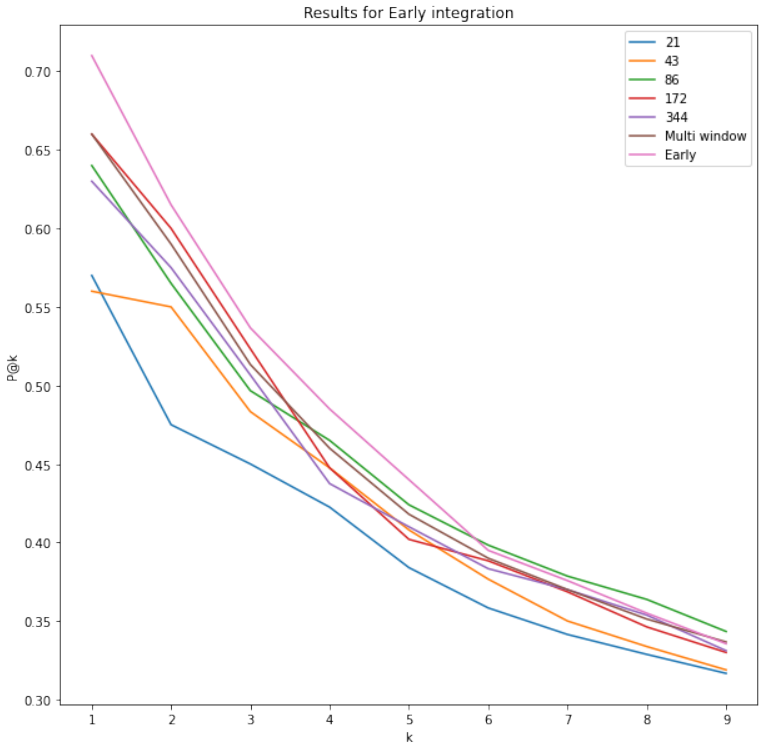

FIGURE 4.5: Early Integration and RbQ-c Complete using Open L3 and with an embedding size of 6144

The plot for RbQ-c Approximate is shown in Figure 4.6.

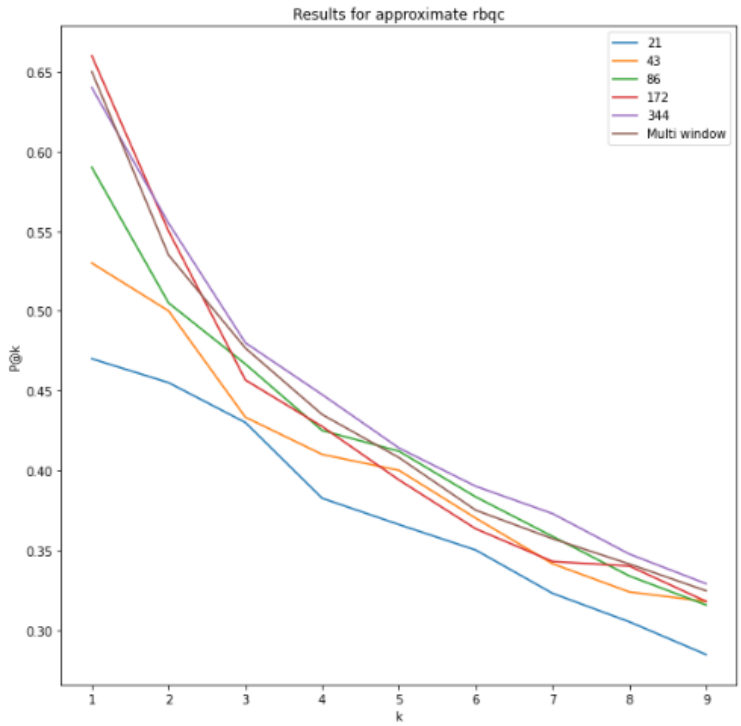

FIGURE 4.6: RbQ-c Approximate using Open L3 and an embedding size of 6144

We observe that Early integration outperform both RbQ-c Approximate and RbQ-c Complete in case of OpenL3. When we compare RbQ-c Approximate with RbQ-c Complete, we observe that RbQ-c Complete is time asynchronous but we must compromise the computational complexity, whereas, RbQ-c Approximate is synchronous and has less computational complexity.

Therefore, we conclude from the Section 4.2.1 and Section 4.2.2 that Early Integration is the best possible method to implement the similarity of audio scenes using OpenL3.

Now to compare between MFCC and OpenL3 is a bit difficult task since we have a complex multidimensional dataset. We need to find a possible way to compare the different factors like *Feature Type* (whether it is MFCC or OpenL3 with embedding size 512 or OpenL3 with embedding size 6144), *size of integration window* in case of RbQ-c, *type of similarity*(Early Integration, RbQ-c Complete or RbQ-c Approximate), *value of k*. It is too complex to plot this since it has 4 dimensions. Therefore, we form a tabular representation as shown in Table 4.1.

p@5	**MFCC**	**OpenL3 emb size = 512**	**OpenL3 emb size = 6144**
Early Integration	0.364	**<u>0.494</u>**	**0.440**
RbQ-c Complete			
21	0.362	0.422	0.384
43	**0.390**	0.444	0.408
86	0.364	0.474	0.424
172	0.334	0.450	0.402
344	0.344	0.460	0.410
Multi Window	0.372	0.472	0.418
RbQ-c Approximate			
21	0.340	0.406	0.366
43	0.326	0.410	0.400
86	0.336	0.456	0.412
172	0.336	0.442	0.394
344	0.338	0.462	0.414
Multi Window	0.358	0.456	0.408

TABLE 4.1: Comparison.

We use metic as precision at k=5 to compare. From the table we see that for MFCCs, the highest values are observed using RbQ-c Complete with small window size (given in bold in MFCC column) when compared to Early Integration and RbQ-c Approximate. Therefore, RbQ-c Complete is the best method for implementing MFCC.

When we look in the case of OpenL3, Early Integration has dominant values (given in bold) when compared to RbQ-c Complete and RbQ-c Approximate. This helps to add to our conclusion that Early Integration is the best suitable method for implementing similarity between two acoustic scenes for OpenL3.

Chapter 5

Conclusion

Our project describes the different representations or approaches for modeling acoustic scenes using MFCCs and OpenL3. When we compare MFCC with openL3 Method, we see that the computational cost of MFCC is much lower than that of OpenL3. MFCC obtains result faster while OpenL3 was very slow in obtaining the required data for computing the different approaches. Therefore, we set the runtime type as GPU for faster execution. Even we see that OpenL3 required more RAM space during its execution.

We also understand from our experiment that RbQ-c is generally useful for low dimensional features which are not learned like in MFCC. MFCC takes low dimensional space like 40 and if we sum everything, we leave behind a less informative data due lose of information. Therefore, with parametric similarity measure as RbQ-c, when we sum, we don't crash information in a harmful way. For much more complex and high dimensional embeddings' and that which requires learning like L^3-Net, if we sum everything, then it is less detrimental. Therefore, we recommend using Early integration for implementing the L^3-Net.

Bibliography

[1] Joakim Andén and Stéphane Mallat. “Deep Scattering Spectrum”. In: *CoRR* abs/1304.6763 (2013). arXiv: 1304.6763. URL: http://arxiv.org/abs/1304.6763.

[2] R. Arandjelovic and A. Zisserman. “Look, Listen and Learn”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 609–617.

[3] Josep Lluis Arcos, Enric Guaus, and Tan H. Ozaslan. “Analyzing musical expressivity with a soft computing approach”. In: *Fuzzy Sets and Systems* 214 (2013), pp. 65 –74.

[4] J. Cramer et al. “Look, Listen and Learn More: Design Choices for Deep Audio Embeddings”. In: *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK, 2019, pp. 3852–3856.

[5] Mathieu Lagrange et al. “The bag-of-frames approach: A not so sufficient model for urban soundscapes”. In: *The Journal of the Acoustical Society of America* 138.5 (2015), EL487–EL492.

[6] Vincent Lostanlen et al. “Relevance-based quantization of scattering features for unsupervised mining of environmental audio”. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2018.1 (Dec. 2018). DOI: 10.1186/s13636-018-0138-4. URL: https://hal.archives-ouvertes.fr/hal-01887403.

[7] Annamaria Mesaros et al. “Detection and Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.2 (2018), pp. 379–393.