**NOTE:**

Do Not Change the formatting of any page. All pages have already been formatted for you. Just start writing. Before writing, YOU MUST CHECK the formatting before starting writeup, so that you may remember what is the font size of heading and its text, sub-heading and its text, sub-sub-heading and its text, the text of table and so on. This box is just for hint. Remove it once you understand the hint.

# CHAPTER 1: INTRODUCTION

This is an introductory chapter. After reading this chapter, you will understand the system, like why this system is important, why we are using it, how it will work, what are the objectives of the proposed system and in what kind of applications it can be implemented.

## System Introduction

Living in a world surrounded by different forms of sound from different sources, our brain and auditory system is constantly identifying each sound that it hears, in its way. Classifying audio or sound has been a major field of research for many years now and there has been many tried and tested methods with different models and features which have proven to be useful and accurate. Classification of audio can range from fields like multimedia, bioacoustics monitoring, intruder detection in wildlife areas to audio surveillance, and environmental sounds.

There are three different stages that are attached to the classification of sound; pre-processing of the audio signal, specific spectral feature extraction, and finally the classification of the audio signal. Signal pre-processing samples the input audio signal into various fragments which are utilized for extricating essential features. Zero-crossing rate, spectral flux, centroid, Chroma vector, MFCC (Mel-frequency Cepstral Coefficient), and poly features are among the most well-known handcrafted features for audio classification. For the sake of our approach to audio classification, we have chosen the following features: Chroma gram, Mel spectrogram, Spectral Contrast, Tonnetz, MFCC etc. The idea of MFCC is to convert time-domain signals into frequency domain signals and use Mel filters to mimic cochlea that has more filters at low frequency and fewer filters at high frequency. Thus, it is safe to conclude that the feature MFCC and its characteristics are focused on the audibility of the human hearing system that can accommodate the dynamic nature of true-life sounds with the way that they are treated with feature vectors for classification.

## Background of the System

An early method to handle sound recognition problem was proposed by Sawhney and Maesx, which modeled the temporal evolution of audio features and employ recurrent neural networks and a k-nearest neighbor criterion to model the mapping between features and categories. However, such temporal feature could hardly obtain an accuracy larger than 70%. Then Eronen m mn et al. discovered that the MFCC feature that based on human recognition of soundscape and how human ears discern different frequencies. Some soundscape research extract MFCC features to describe the local spectral envelop of audio signals and classify with SVM (radial basis function kernel) and random forest.

The result shows accuracy of approximately 70% which is much lower than deep learning method. Other strategy using HMM to translate an audio into events performs poorly in the presence of interfering acoustic noise. Since noise is unavoidable in practical situation, we must find features and models more robust. Therefore, researchers tend to find powerful machine learning methods like CNN, RNN with high dimensional input features such as spectrograms. These improve robustness because of the discriminative capabilities of the back-end classifiers. Our work is distinctive in the following ways. We will apply CNN with two different features: Mel and MFCC, to compare the performance of different features. And then we will try model to predict by features.

## Objectives of the System

* + To identify the object in sound.
  + It can be used in security systems.
  + It can help in controlling traffic signals by identifying sound of traffic jam.
  + It can be used in vehicles to determine if the vehicle is in running state or starting state.

## Significance of the System

This study deals with the role of sounds in object recognition in humans. Indeed, some objects are easily associated with a sound, i.e., some objects possess either a typical sound or category of sounds. This is the case, for example, of objects such as “bell” or “motorbike.” Other objects do not possess typical sounds or can be associated with particular sounds only with difﬁculty. This is the case, for example, of objects such as “table” or “pillow. ”Given that objects can be classiﬁed as a function of whether they possess or not a typical sound, a legitimate question is whether the typical sounds play any role in the visual recognition of the related objects. There are at least two opposed scenarios to frame this question.

In the ﬁrst scenario, upon the presentation of a visual object the system ﬁrst accesses an abstract representation of that object and then depending on the task at hand accesses the representations of information related to that object: among these representations is the representation of the typical sound. Thus, in this scenario, the access to the typical sound is post-categorical, in the sense that the object is ﬁrst recognized as an instance of a particular kind (e.g., a “dog”) and then the related information is retrieved. Here, the typical sound may be activated but, since its retrieval follows the identiﬁcation of the object, it does not play any role in the recognition of the object. In the second scenario, all stored representations associated to a given object are immediately and mandatorily activated upon the visual presentation of an instance of that kind of object. Here, the identiﬁcation of the object does not consist in the activation of an abstract semantic representation of this objects but instead corresponds to the activation of all stored representations. In other words, object identiﬁcation is the activation of object knowledge. So, this system is important in classification and recognition of object sounds.

# CHAPTER 2: REQUIREMENT SPECIFICATIONS

In this chapter you will be able to understand requirements of the system. Understanding the requirements or requirement gathering is most important part for building a system. In this chapter you will understand both user and system level requirements. It includes functional requirements, non-functional requirements, system requirements and behavioral requirements.



## Product Scope

The scope of this work is focused on a urban sounds classification. The elaboration of this proposal is organized as follows describe the research field of Urban. As an important processing step, feature extraction plays a critical role that will significantly affect the final classification performance. We try to break through the performance bottleneck, using novel feature sets extracted with image information retrieval techniques. It describes the experiments applying convolutional neural network (CNN), a state-of-the-art image digit recognition algorithm, to automatic extraction of sound pattern features. The system architecture, the characteristics of CNN and the classification performance are explained. studies the invariance of the widely used MFCC feature set to Urban Sounds.

## Product Description

### Product Perspective

From the technology perspective, object sound recognition has a long history with several waves of major innovations. Most recently, the field has benefited from advances in deep learning and big data. The advances are evidenced not only by the surge of academic papers published in the field, but more importantly by the worldwide industry adoption of a variety of deep learning methods in designing and deploying speech recognition systems.

### Product Functionality

* Detect Object by Voice File
* Detect Object by Recording
* Detect Object in Real-time

### Users and Characteristics

In this System user can be anything like if it is used in security system the user will the security system that will be connected to this system and generate an output by the response of this system.

### Operating Environment

This application will install on windows and minimum GPU will be required 2.17, in case of security system input device is required to accept sounds from the environment. Microphone will be used for input to the system. Or the audio file can be upload to get the result.

## Specific Requirements

### Functional Requirements

#### Take voice input in Real-time

#### The system shall be able to take input in Real-time and shows the output on interface.

#### Take voice input by audio file

The system should be able to process audio file and shows output to the interface.

#### Use Case Diagram

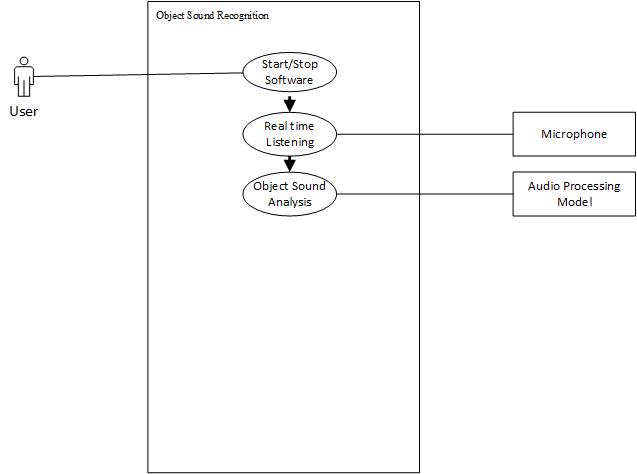
..

#### Use Case Descriptions

* User start or stop the software
* System will interact with the microphone while real-time listening
* System will interact with audio processing model while performing object sound analysis

### Behavioral Requirements

* User will click on start button
* Application will receive sound from the environment by using microphone
* This sound will be sent to the Audio Processing Model
* Model returns the matching result if any found.



### External Interface Requirements

#### User Interface

Chart

Description automatically generatedThere is only one screen for the user to start and stop the system and view the system’s performance.

#### Other Interfaces (if any)

Cutting-edge GPU is required if it will be developed in a local lab. Preferably, 5800+ Quad core GPU would be fine. 8000+ better. Since the dataset is more than 10 GB, preferably, 4 GB or more amount of RAM is needed. 1 TB + of the mass storage device is needed.

Related to software interface, Application has the button on the main page to start and stop the system and also a graph to see the sound’s performance.

Chart

Description automatically generated

There is no communication interface needed. The application works alone. Application gets data from the environment or files and processes the data and shows the results. There is no API’s or any communication interface used.

## Non-functional Requirements

### Performance Requirements

#### Response Time

The system will work with real-life data. Response time to identifying shouldn't exceed 300 msec.

#### Error Handling

When the system fails to predict in a given response time, it should give an alert message to the interface.

#### Workload

The system should be able to handle identifying given input while the microphone is taking input.

### Safety and Security Requirements

* Make sure system will not crashed due to electricity issue
* System will be safe from electricity shocked.
* System will be safe from theft.
* There is no hurdles between the environment and system input place.

### Software Quality Attributes

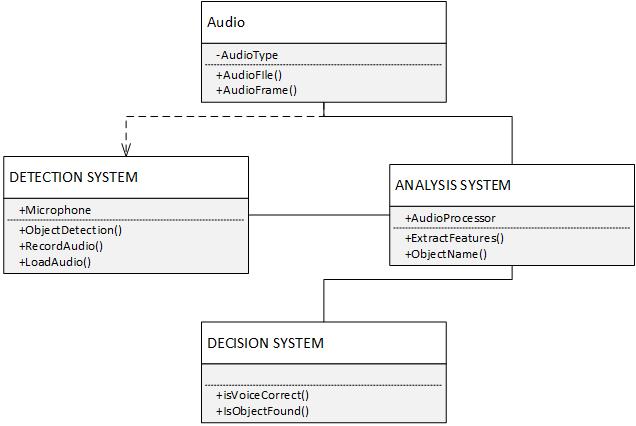
|  |  |
| --- | --- |
| Term | Definition |
| Functionality | System designed to identify the objects. |
| Reliability | System should operate 99.9 % in runtime. |
| Robustness | System should successfully identify urban sounds objects which will be trained with training data. |
| Portability | System should work Ubuntu 16.04 or later OS version. |
| Efficiency | Source codes of the system should be clear and operate efficiently. |

# CHAPTER 3: DESIGN SPECIFICATIONS

In this chapter, you will understand design specifications of object sound detection and recognition. You will find out different diagrams like class diagram for logical design purposes, sequence diagram for system interaction purposes and so on.

## System Design

## Logical Design

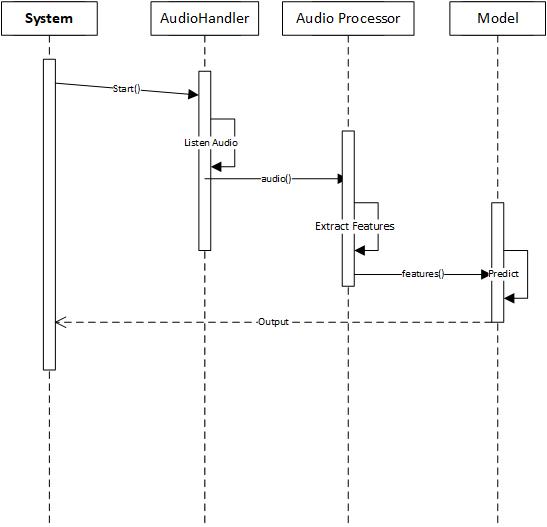


## System Architecture

TO DO: Entity Relationship Diagram (ERD) in case of software having database for information storage>> or system diagram if project is a hardware based one or system model (algorithm and flow design) if project is a machine learning or a deep learning based one. For gaming based projects flow diagram.

## System Interaction and Use Cases

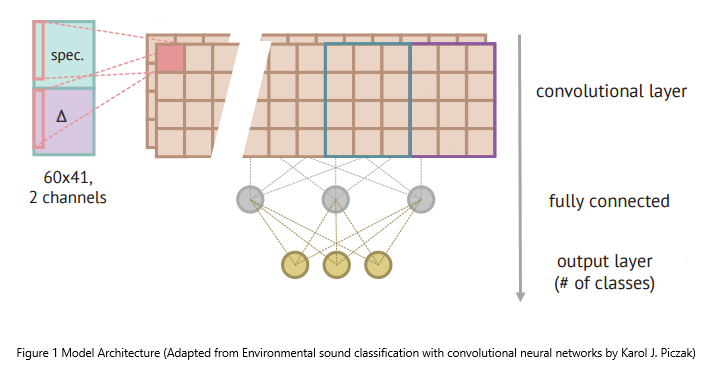
* When user clicks on the start button, it will start the **Audio Handler** that record the audio file.
* Then it sends this file to the **Audio Processor** that Extract features from the file.
* Then it sends these features to the **Model**.
* **Model** generates prediction that sends back to the user as output.



## Algorithmic Viewpoint

<*The detailed design description of operations (methods, functions), this applies to components, classes, and individual methods as design entities. The UML diagrams used to serve the purpose are Decision Table, Pseudo Code.*

TO DO: Provide Decision Table or Pseudo Code for the complete software>>



# CHAPTER 4: DEVELOPMENT AND TOOLS

This chapter provide the description about the tools and techniques used in this project. The tool used is Spyder with language python. The different libraries of python are used in this project.

## Introduction

## While deploying the application, we need a system that has 1 TB+ mass storage device and containing minimum 4 GB of RAM. We need python 3.9 or more to deploy the system. We will import some libraries to manage data and to building the model. Than we write function to convert our audio data into images. We use LIBROSA to extract audio time-series and sampling rate of audio files. Then we apply processing.

## Development

### Tools and Technologies Used

Python 3.9 or above version need to use the application, Cutting-edge GPU is required if it will be developed in a local lab. Preferably, 5800+ Quad core GPU would be fine. 8000+ better. Since the dataset is more than 10 GB, preferably, 4 GB or more amount of RAM is needed. 1 TB + of the mass storage device is needed.

CONVOLUTIONAL NEURAL NETWORK

DATASET: URBANSOUND8K

### External APIs/ External Hardware

There is no API’s used in the application. Application works standalone.

## System Implementation

1. To begin with, we import the Pandas and Numpy packages, along with a memory profiler to monitor memory use due to the large amount of data conversion that we need to perform on Kaggle’s servers.
2. We then install libav tools, an open source audio processing framework in order to access LibROSA. If you’re doing this locally, you should do the same via console.
3. We import the necessary Keras libraries to build our network as well as other necessary auxiliary packages. Of particular note is the garbage collector package, which allows us to clean up RAM during our data conversion process. Finally, we also build working directories in our Kaggle instance in order to store our converted images.
4. Next, we begin our data conversion process by defining the functions that will convert our .wav files into images, in .jpg format. Briefly, we extract the audio time-series and sampling rate of each .wav file using LibROSA, before building and plotting a spectrogram of the data and saving it as a corresponding image.
5. With that defined, we convert our training data. We will do these in batches of 2000 images at a time, utilizing the garbage collector package to optimize memory use in-between batches.

## Data and Information flow in the system

## Application will get the audio file of few seconds and convert into image format by extract time-series and sampling rate using LIBROSA Library. Then application gave this image file to the algorithm to get the process the image and it checks if it contains any target sound from urban sounds.

## User Interfaces

Chart

Description automatically generated

## Additional Modules and API Provided

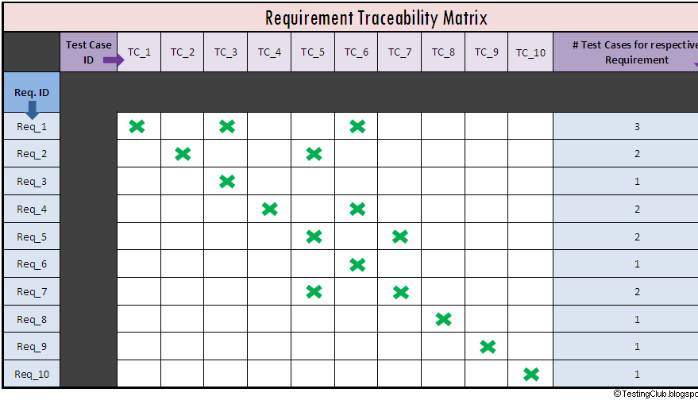
There is no API’s used

# CHAPTER 5: QUALITY ASSURANCE

## Introduction

Quality assurance is one of the most important phases of a system’s development life cycle. In this phase we verify our system to test either it is working according to the requirements of the user or is it working properly according to the functional and non-functional requirements.

## Traceability Matrix



## Test Plan

Table 5.1: Test case for system start up

|  |  |
| --- | --- |
|  | TC-01 |
| Test name | System’s start up |
| Date of test | 22/05/2022 |
| Name of application | Object sound detection and recognition |
| Description | Microphone starts recording while user click on run button. |
| Input | Click on start button. |
| Expected output | Start recording. |
| Actual output | Start Recording. |
| Test Role (Actor) | Team Member |
| Test verified by | Team Member/Supervisor |

Table 5.2: Test case for system’s Microphone

|  |  |
| --- | --- |
|  | TC-02 |
| Test name | System’s microphone |
| Date of test | 22/05/2022 |
| Name of application | Object sound detection and recognition. |
| Description | When driver clicks on start button, microphone enabled for real time to capture sound. |
| Input | None |
| Expected output | Capture sound in Real time |
| Actual output | Capture sound in real time |
| Test Role (Actor) | Team Member |
| Test verified by | Team Member/Supervisor |