**CNN BASED SPEAKER RECOGNITION IN LANGUAGE AND TEXT-INDEPENDENT SMALL-SCALE SYSTEM**

**1. INTRODUCTION:**

Speaker recognition is the identification of a person from the characteristics of voices. Recently, due to a large increase in the field of smart devices, there have been many studies conducted based on how to identify the speaker so as to be able to give a personalized experience to its users. However, most of the studies are dependent on the fact that the user speaks a keyword in order to activate the device used to identify them (Text-dependent speaker recognition) [1]. This makes interacting with the devices a bit monotonous. This study has been done in order to overcome those keyword and language barriers and be able to recognize the user whatever he speaks

**1.1 Objective of the project:**

Speaker Recognition is the ability of the system to recognize the speaker from the set of speaker samples available in the system. It is of 2 types, one uses a keyword, called text dependent systems, and another one can recognize the voice in any language/text, also called as text-independent speaker recognition. In this paper, a text-independent, language independent speaker recognition system is implemented using dense & convolution neural networks. Speaker recognition has found several applications in upcoming electronic products like personal/home assistants, telephone banking and biometric identification. In this paper, we explore a system that uses MFCC along with DNN and CNN as the model for building a speaker recognition system.

**2. LITERATURE SURVEY:**

**"Deep speaker: an end-to-end neural speaker embedding system."**

We present Deep Speaker, a neural speaker embedding system that maps utterances to a hypersphere where speaker similarity is measured by cosine similarity. The embeddings generated by Deep Speaker can be used for many tasks, including speaker identification,  
verification, and clustering. We experiment with Res CNN and GRU architectures to extract the acoustic features, then mean pool to produce utterance-level speaker embeddings, and train using triplet loss based on cosine similarity. Experiments on three distinct datasets suggest that Deep Speaker outperforms a DNN-based i-vector baseline. For example, Deep Speaker reduces the verification equal error rate by 50% (relatively) and improves the identification accuracy by  
60% (relatively) on a text-independent dataset. We also present results that suggest adapting from a model trained with Mandarin can improve accuracy for English speaker recognition.

**"Speaker Recognition using MFCC and Vector Quantization"**

Speaker recognition is one of the most essential tasks in the signal processing which identifies a person from characteristics of voices . In this paper we accomplish speaker recognition using Mel-frequency Cepstral Coefficient (MFCC) with Weighted Vector Quantization algorithm. By using MFCC, the feature extraction process is carried out. It is one of the nonlinear cepstral coefficient functions. Then the pattern matching is accomplished by evaluating the similarity of the unknown speaker and the trained models from the database. For this process, weighted vector quantization is proposed that takes into account the correlations between the known models in the database. Experimentations express that the new methodologies provide higher accuracy and it can observe the correct speaker even from shorter speech samples more reliably.

**"Text Independent speaker identification based on MFCC and Deep Neural Networks"**

Speaker identification is one of the most popular fields in information technology world as mobile phones, security issues and forensics. Different researchers had investigated speaker identification concentrating on features extraction techniques or on identification stage but here in this paper we had tried to have both to have a more robust and accurate speaker identification model. In this paper, we had proposed a hybrid model of text independent speaker identification using mel frequency cepstral coefficients and deep neural network implemented and tested using speech separation challenge database in three different noise levels 0,6,12 dB. The proposed model showed superiority on the state of art models.

**"DNN based Speaker Recognition on Short Utterances"**

This paper investigates the effects of limited speech data in the context of speaker verification using deep neural network (DNN) approach. Being able to reduce the length of required speech data is important to the development of speaker verification system in real world applications. The experimental studies have found that DNN-senone-based Gaussian probabilistic linear discriminant analysis (GPLDA) system respectively achieves above 50% and 18% improvements in EER values over GMM-UBM GPLDA system on NIST 2010 coreext-coreext and truncated 15sec-15sec evaluation conditions. Further when GPLDA model is trained on short-length utterances (30sec) rather than full-length utterances (2min), DNN-senone GPLDA system achieves above 7% improvement in EER values on truncated 15sec-15sec condition. This is because short length development i-vectors have speaker, session and phonetic variation and GPLDA is able to robustly model those variations. For several real world applications, longer utterances (2min) can be used for enrollment and shorter utterances (15sec) are required for verification, and in those conditions, DNN-senone GPLDA system achieves above 26% improvement in EER values over GMM-UBM GPLDA systems.

**"Neural Network Based Speaker Classification and Verification Systems with Enhanced Features"**

This work presents a novel framework based on feed-forward neural network for text-independent speaker classification and verification, two related systems of speaker recognition. With optimized features and model training, it achieves 100% classification rate in classification and less than 6% Equal Error Rate (ERR), using merely about 1 second and 5 seconds of data respectively. Features with stricter Voice Active Detection (VAD) than the regular one for speech recognition ensure extracting stronger voiced portion for speaker recognition, speaker-level mean and variance normalization helps to eliminate the discrepancy between samples from the same speaker. Both are proven to improve the system performance. In building the neural network speaker classifier, the network structure parameters are optimized with grid search and dynamically reduced regularization parameters are used to avoid training terminated in local minimum. It enables the training goes further with lower cost. In speaker verification, performance is improved with prediction score normalization, which rewards the speaker identity indices with distinct peaks and penalizes the weak ones with high scores but more competitors, and speaker-specific thresholding, which significantly reduces ERR in the ROC curve. TIMIT corpus with 8K sampling rate is used here. First 200 male speakers are used to train and test the classification performance. The testing files of them are used as in-domain registered speakers, while data from the remaining 126 male speakers are used as out-of-domain speakers, i.e. imposters in speaker verification.

**“Deep Neural Network Approaches to Speaker and Language Recognition”**

The impressive gains in performance obtained using deep neural networks (DNNs) for automatic speech recognition (ASR) have motivated the application of DNNs to other speech technologies such as speaker recognition (SR) and language recognition (LR). Prior work has shown performance gains for separate SR and LR tasks using DNNs for direct classification or for feature extraction. In this work we present the application of single DNN for both SR and LR using the 2013 Domain Adaptation Challenge speaker recognition (DAC13) and the NIST 2011 language recognition evaluation (LRE11) benchmarks. Using a single DNN trained for ASR on Switchboard data we demonstrate large gains on performance in both benchmarks: a 55% reduction in EER for the DAC13 out-of-domain condition and a 48% reduction in Cavg on the LRE11 30 s test condition. It is also shown that further gains are possible using score or feature fusion leading to the possibility of a single i-vector extractor producing state-of-the-art SR and LR performance.

**“Text-Independent Speaker Verification Using 3D Convolutional Neural Networks”**

In this paper, a novel method using 3D Convolutional Neural Network (3D-CNN) architecture has been proposed for speaker verification in the text-independent setting. One of the main challenges is the creation of the speaker models. Most of the previously-reported approaches create speaker models based on averaging the extracted features from utterances of the speaker, which is known as the d-vector system. In our paper, we propose an adaptive feature learning by utilizing the 3D-CNN s for direct speaker model creation in which, for both development and enrollment phases, an identical number of spoken utterances per speaker is fed to the network for representing the speakers' utterances and creation of the speaker model. This leads to simultaneously capturing the speaker-related information and building a more robust system to cope with within-speaker variation. We demonstrate that the proposed method significantly outperforms the traditional d-vector verification system. Moreover, the proposed system can also be an alternative to the traditional d-vector system which is a one-shot speaker modeling system by utilizing 3D-CNNs.

**"End-to-end attention based text-dependent speaker verification."**

A new type of End-to-End system for text-dependent speaker verification is presented in this paper. Previously, using the phonetic discriminate/speaker discriminate DNN as a feature extractor for speaker verification has shown promising results. The extracted frame-level (bottleneck, posterior or d-vector) features are equally weighted and aggregated to compute an utterance-level speaker representation (d-vector or i-vector). In this work we use a speaker discriminate CNN to extract the noise-robust frame-level features. These features are smartly combined to form an utterance-level speaker vector through an attention mechanism. The proposed attention model takes the speaker discriminate information and the phonetic information to learn the weights. The whole system, including the CNN and attention model, is joint optimized using an end-to-end criterion. The training algorithm imitates exactly the evaluation process — directly mapping a test utterance and a few target speaker utterances into a single verification score. The algorithm can smartly select the most similar impostor for each target speaker to train the network. We demonstrated the effectiveness of the proposed end-to-end system on Windows 10 “Hey Cortana” speaker verification task.

**“Research on Different Feature Parameters in Speaker Recognition”**

Feature parameters extraction is critical for speaker recognition research. The paper presents the function of pitch, formant and Mel frequency central coefficient (MFCC) in speaker recognition. It can increase the identification rate effectively for feature parameter sorts the speech corpus. Using Euclid Distance to compare feature parameters is very effective.

**“Speaker Recognition System using Gaussian Mixture Model”**

The medical field is one of the most important are as that is researched by numerous researchers in order to develop the existing system. Though there have been numerous new and innovative systems developed, there are always some challenges present in the field that needs to be solved. It is important for a healthcare system to make sure that it is secure enough and any leak of the patient data could lead to numerous other risk factors. In this paper, we propose a speaker recognition system that makes use of the Gaussian Mixture Model which identifies a person from characteristics of their voice to authenticate them to access the medical records. The model is able to determine the gender of the person to be authenticated and also recognizes the person. The model uses Cepstral Analysis as the feature extraction technique using the feature Mel Frequency Cepstral Coefficient (MFCC). The proposed work is observed to work well when compared with other existing systems and could be of much use in performing and giving access to the authorized employees in a health care system.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

There are various approaches to solving the speaker recognition problem. The solution is defined by two parameters : The feature of the voice signal to be used like Mel Frequency Cepstral Coefficients (MFCC), Linear Prediction Cepstral Coefficient (LPCC), Perceptual Linear Prediction (PLP) and the modelling technique used to learn the voice samples like Artificial Neural Networks (ANN), Gaussian Mixture Model (GMM), vector quantization etc. Studies have also been done where multiple techniques are combined, such as using different models for feature extraction and classification. Of these choices, majority approaches use MFCC as the feature to be used since it is most effective for speaker recognition. Using MFCC features and, studied phenomic based speaker recognition systems. However, the accuracies seem to fluctuate based on the words and syllables pronounced. Also, relatively lower accuracies have been achieved using GMM, Vector Quantisation and Deep Neural Networks (DNN)

**Disadvantages of Existing System:**

1. Less Prediction.
2. Security is less.
3. **3.2 Proposed System**

Speaker recognition refers to the process of identifying a person based on their vocal characteristics. As the number of available smart devices continues to grow exponentially, there have been several experimentation with several methods for determining the identity of the speaker, thus because it can cater to each individual user's preferences. The majority of investigations, however, rely on the fact that user activates gadget by saying a keyword aloud(Speaker recognition based on text) to determine who they are. Due to this, using the gadgets might get tedious after a while. Because of the difficulties with keyword research, this study was undertaken.Considering linguistic difficulties in addition to user recognition no matter what he says

**Advantages of Proposed System:**

1. Security is more.

2. More Prediction.

**MODULES:**

1. Upload Dataset: Using this module we will upload speech recordings of five different persons
2. MFCC Processing: Using this module we are removing noise from audio and then applying MFCC algorithm to extract speech features and this features will be used to train both CNN and Random forest algorithms
3. runCNN: using this module we will apply above MFCC features on CNN algorithm to build CNN training model and then apply test data on that trained model to predict or recognize persons
4. runRandomForest: Above MFCC features will be used to train random forest algorithm and then apply test data on random forest to calculate prediction accuracy
5. Graph: using this module we will plot CNN accuracy and loss graph
6. Predict: Using this function we will upload test speech data and then CNN will recognize person from that speech.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loa ded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**SEQUENCE DIAGRAM:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COMPONENT DIAGRAM:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

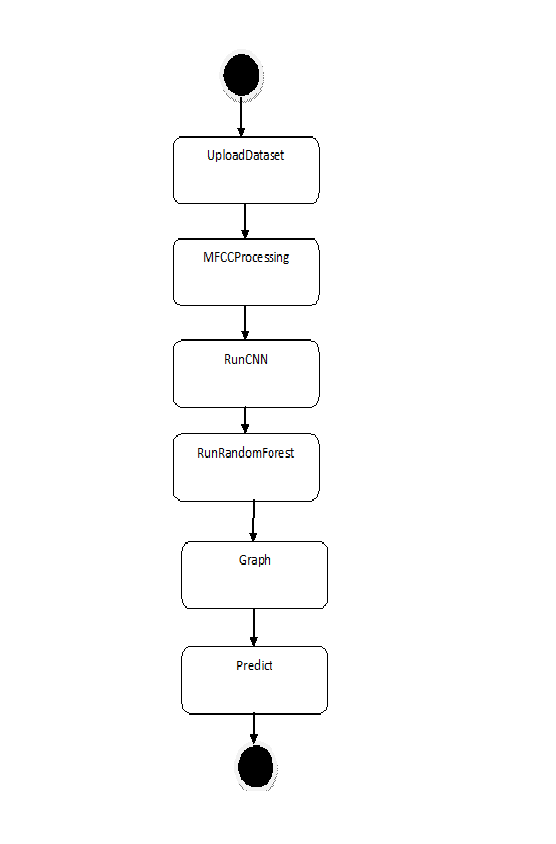
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**ACTIVITY DIAGRAM:**

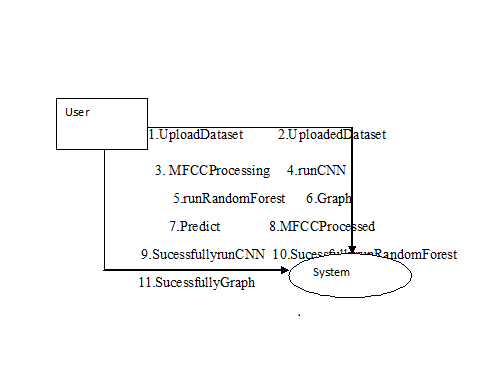
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent



**Data flow :**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results



**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code:**

**DepressionDetection.py**

from tkinter import \*

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from tkinter import simpledialog

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from scipy.fft import fft

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import f1\_score

import seaborn as sns

from sklearn.metrics import confusion\_matrix

main = tkinter.Tk()

main.title("A Time-Frequency Based Suspicious Activity Detection for Anti-Money Laundering") #designing main screen

main.geometry("1000x650")

global filename

global fscore

global X\_train, X\_test, y\_train, y\_test

global X, Y

global le1, le2, le3, dataset, rf

def upload():

global filename, dataset

filename = filedialog.askopenfilename(initialdir = "Dataset")

text.delete('1.0', END)

text.insert(END,filename+' Loaded')

dataset = pd.read\_csv(filename)

text.insert(END,str(dataset.head()))

label = dataset.groupby('Label').size()

label.plot(kind="bar")

plt.title("Transaction Graph 0 Means Normal Transaction & 1 Means Money Laundering Transaction")

plt.show()

def preprocessDataset():

global X, Y

global le1, le2, le3, dataset

text.delete('1.0', END)

dataset.fillna(0, inplace = True)

cols = ['type','nameOrig','nameDest']

le1 = LabelEncoder()

le2 = LabelEncoder()

le3 = LabelEncoder()

dataset[cols[0]] = pd.Series(le1.fit\_transform(dataset[cols[0]].astype(str)))

dataset[cols[1]] = pd.Series(le2.fit\_transform(dataset[cols[1]].astype(str)))

dataset[cols[2]] = pd.Series(le3.fit\_transform(dataset[cols[2]].astype(str)))

Y = dataset['Label'].ravel()

dataset.drop(['Label'], axis = 1,inplace=True)

X = dataset.values

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

text.insert(END,"Processed Dataset values\n\n")

text.insert(END,str(X)+"\n\n")

text.insert(END,"Total records found in dataset : "+str(X.shape[0])+"\n")

text.insert(END,"Total features found in dataset : "+str(X.shape[1] + 1)+"\n")

def runTransactionRandomForest():

global X, Y, fscore, rf

fscore = []

text.delete('1.0', END)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

cls = RandomForestClassifier(ccp\_alpha=0.5)

cls.fit(X\_train, y\_train)

predict = cls.predict(X\_test)

f = f1\_score(y\_test, predict,average='macro') \* 100

fscore.append(f)

CM = confusion\_matrix(y\_test, predict)

TN = CM[0][0] / len(y\_test)

FN = CM[1][0] / len(y\_test)

TP = CM[1][1] / len(y\_test)

FP = CM[0][1] / len(y\_test)

cls = RandomForestClassifier()

cls.fit(X\_train, y\_train)

rf = cls

text.insert(END,"Random Forest FSCORE on Transaction Data : "+str(f)+"\n")

text.insert(END,"False Positive Rate (FPR) : "+str(FP)+"%\n")

text.insert(END,"False Negative Rate (FNR) : "+str(FN)+"%\n")

text.insert(END,"True Negative Rate (PPV) : "+str(TN)+"%\n")

text.insert(END,"True Positive Rate (TPR) : "+str(TP)+"%\n\n")

def runTimeFrequencyRandomForest():

global X, Y, fscore, rf

fft\_data = fft(X)

X = []

for i in range(len(fft\_data)):

temp = []

for j in range(len(fft\_data[i])):

temp.append(float(fft\_data[i,j]))

X.append(temp)

X = np.asarray(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

cls = RandomForestClassifier()

cls.fit(X\_train, y\_train)

predict = cls.predict(X\_test)

f = f1\_score(y\_test, predict,average='macro') \* 100

fscore.append(f)

CM = confusion\_matrix(y\_test, predict)

TN = CM[0][0] / len(y\_test)

FN = CM[1][0] / len(y\_test)

TP = CM[1][1] / len(y\_test)

FP = CM[0][1] / len(y\_test)

text.insert(END,"Random Forest FSCORE on Transaction & Time Frequency Data : "+str(f)+"\n")

text.insert(END,"False Positive Rate (FPR) : "+str(FP)+"%\n")

text.insert(END,"False Negative Rate (FNR) : "+str(FN)+"%\n")

text.insert(END,"True Negative Rate (PPV) : "+str(TN)+"%\n")

text.insert(END,"True Positive Rate (TPR) : "+str(TP)+"%\n\n")

def predict():

global rf, le1, le2, le3

filename = filedialog.askopenfilename(initialdir = "Dataset")

text.delete('1.0', END)

dataset = pd.read\_csv(filename)

dataset.fillna(0, inplace = True)

cols = ['type','nameOrig','nameDest']

dataset[cols[0]] = pd.Series(le1.transform(dataset[cols[0]].astype(str)))

dataset[cols[1]] = pd.Series(le2.transform(dataset[cols[1]].astype(str)))

dataset[cols[2]] = pd.Series(le3.transform(dataset[cols[2]].astype(str)))

dataset = dataset.values

predict = rf.predict(dataset)

print(predict)

for i in range(len(predict)):

if predict[i] == 0:

text.insert(END,"Transaction data : "+str(dataset[i])+" ====> PREDICTED AS NORMAL\n\n")

if predict[i] == 1:

text.insert(END,"Transaction data : "+str(dataset[i])+" ====> PREDICTED AS MONEY-LAUNDERING\n\n")

def graph():

height = fscore

bars = ('Transaction Data FScore','Time Frequency Data FScore')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.title("Random Forest FScore on Transaction & Time Frequency Data")

plt.xticks(y\_pos, bars)

plt.show()

def close():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='A Time-Frequency Based Suspicious Activity Detection for Anti-Money Laundering', justify=LEFT)

title.config(bg='lavender blush', fg='DarkOrchid1')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=100,y=5)

title.pack()

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Anti-Money Laundering Dataset", command=upload)

uploadButton.place(x=10,y=100)

uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess Dataset", command=preprocessDataset)

processButton.place(x=10,y=150)

processButton.config(font=font1)

transactionButton = Button(main, text="Run Random Forest with Transaction", command=runTransactionRandomForest)

transactionButton.place(x=10,y=200)

transactionButton.config(font=font1)

tfButton = Button(main, text="Run With Transaction & Time Frequency", command=runTimeFrequencyRandomForest)

tfButton.place(x=10,y=250)

tfButton.config(font=font1)

graphButton = Button(main, text="Comparison Graph", command=graph)

graphButton.place(x=10,y=300)

graphButton.config(font=font1)

predictButton = Button(main, text="Predict Money Laundering from Test Data", command=predict)

predictButton.place(x=10,y=350)

predictButton.config(font=font1)

closeButton = Button(main, text="Close Application", command=close)

closeButton.place(x=10,y=400)

closeButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=160)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=400,y=100)

text.config(font=font1)

main.config(bg='light coral')

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| O1 | UploadDataset | Verify  dataset  updated or not | If dataset is May not be Uploaded | we cannot do any further operations | we can do further operations | | High | High |
| 02 | MFCCProcessing | Verify MFCCProcessing  dataset is updated or not | If MFCCProcessing  Dataset is may not be Updated | we cannot do any further operations | we can do further operations | | High | High |
| 03 | runCNN | Verify CNN is Run Sucessfully or not | If CNN is may not be Run Sucessfully | We cannot run  operation | We can Run the Operation | | High | High |
| 04 | runRandomForest | Verify RandomForest  Run successfully or not | If RandomForest  is may not run Sucessfully | We cannot run  operation | We can Run the Operation | | High | High |
| 05 | Graph | Verify Graph Displayed Sucessfully or not | If Graph May not Displayed Sucessfully | We cannot Get similar graph | We can get similar Graph | | High | High |
| 06 | predict | Verify is data Predicted or not | If data not predicted | We cannot generate  data | We can Generate  data | | High | High |

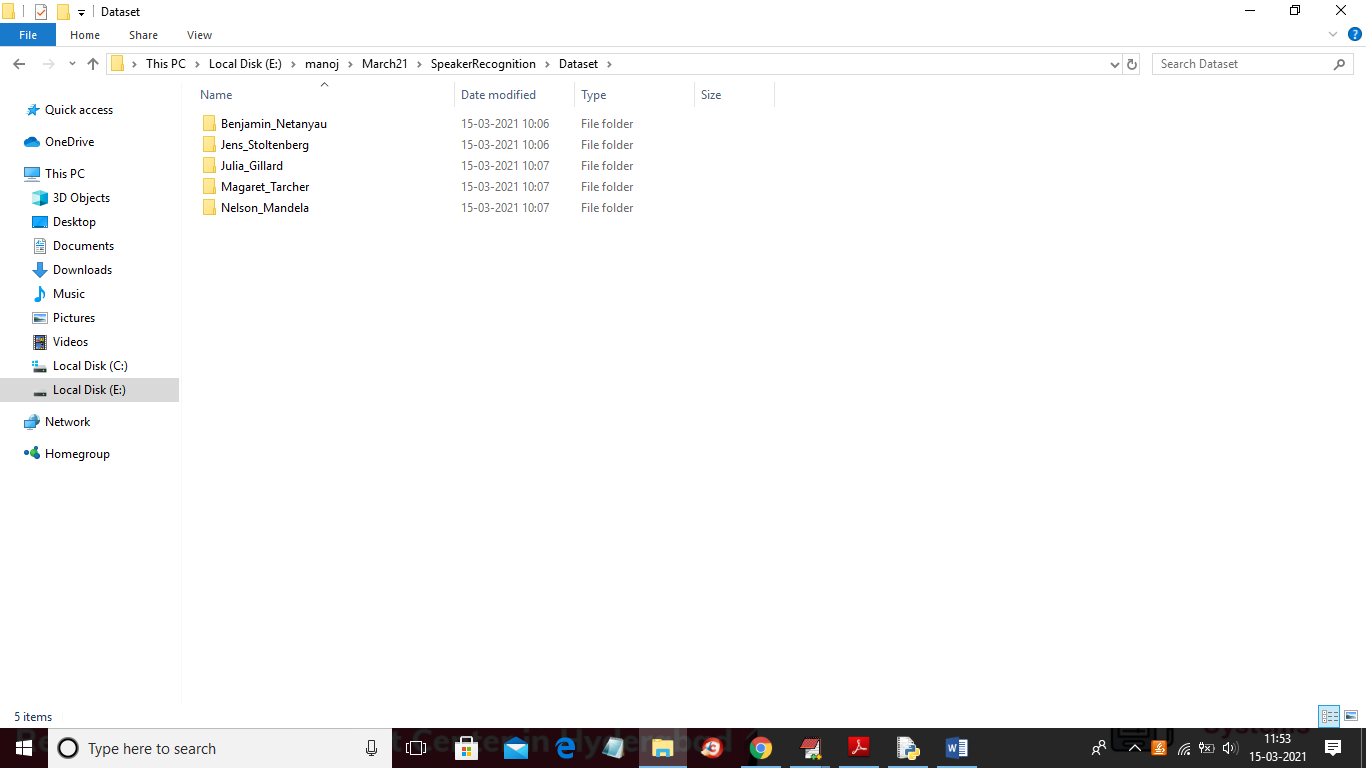
**7. SCREENSHOTS:**

CNN BASED SPEAKER RECOGNITION IN LANGUAGE AND TEXT-INDEPENDENT SMALL-SCALE SYSTEM

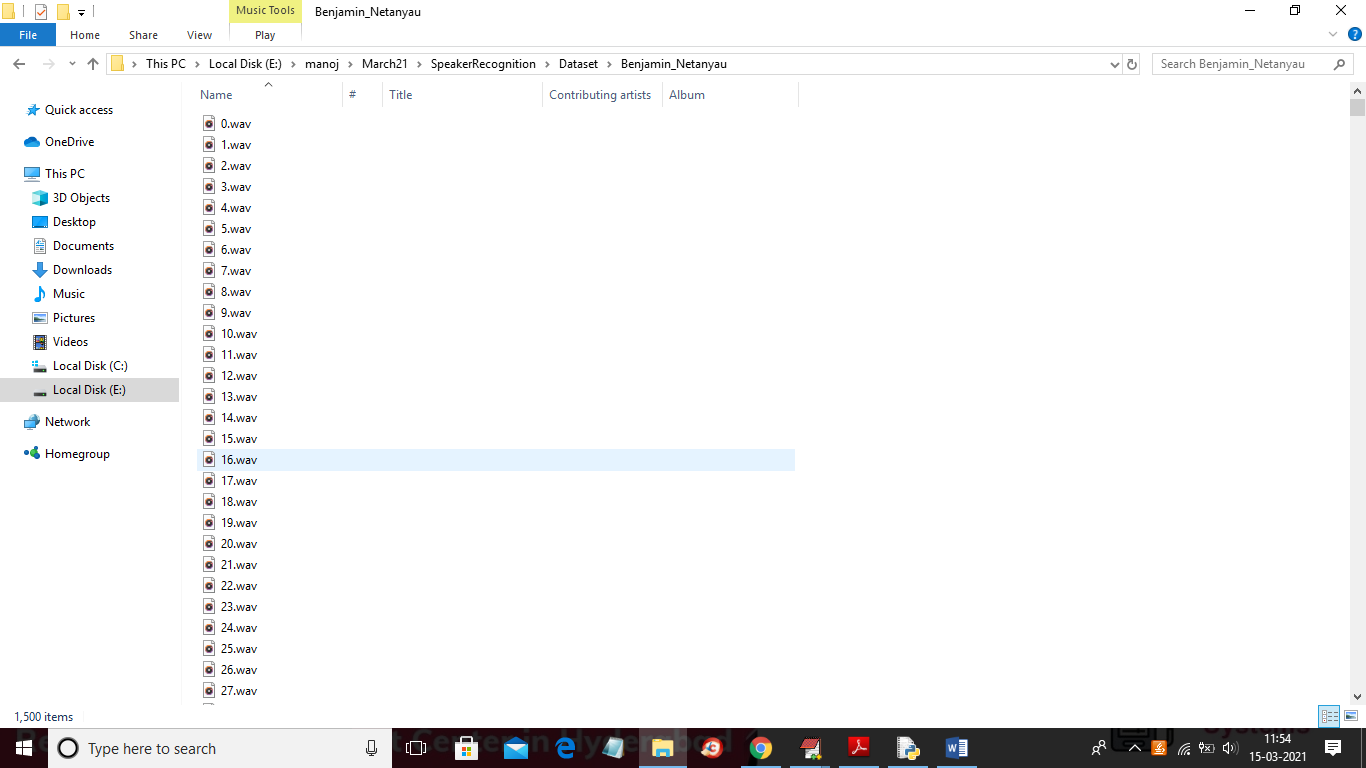
In this paper requirement student has said CNN based recognition in title and in paper he mention about to use Random Forest so I implement both algorithms and comparing accuracy of both algorithms. To implement this project we have designed following modules

1. Upload Dataset: Using this module we will upload speech recordings of five different persons
2. MFCC Processing: Using this module we are removing noise from audio and then applying MFCC algorithm to extract speech features and this features will be used to train both CNN and Random forest algorithms
3. runCNN: using this module we will apply above MFCC features on CNN algorithm to build CNN training model and then apply test data on that trained model to predict or recognize persons
4. runRandomForest: Above MFCC features will be used to train random forest algorithm and then apply test data on random forest to calculate prediction accuracy
5. Graph: using this module we will plot CNN accuracy and loss graph
6. Predict: Using this function we will upload test speech data and then CNN will recognize person from that speech.

To implement above project we have used speech audio of five different persons and this dataset saved inside ‘Dataset’ folder. Below screen shots showing all five persons folders



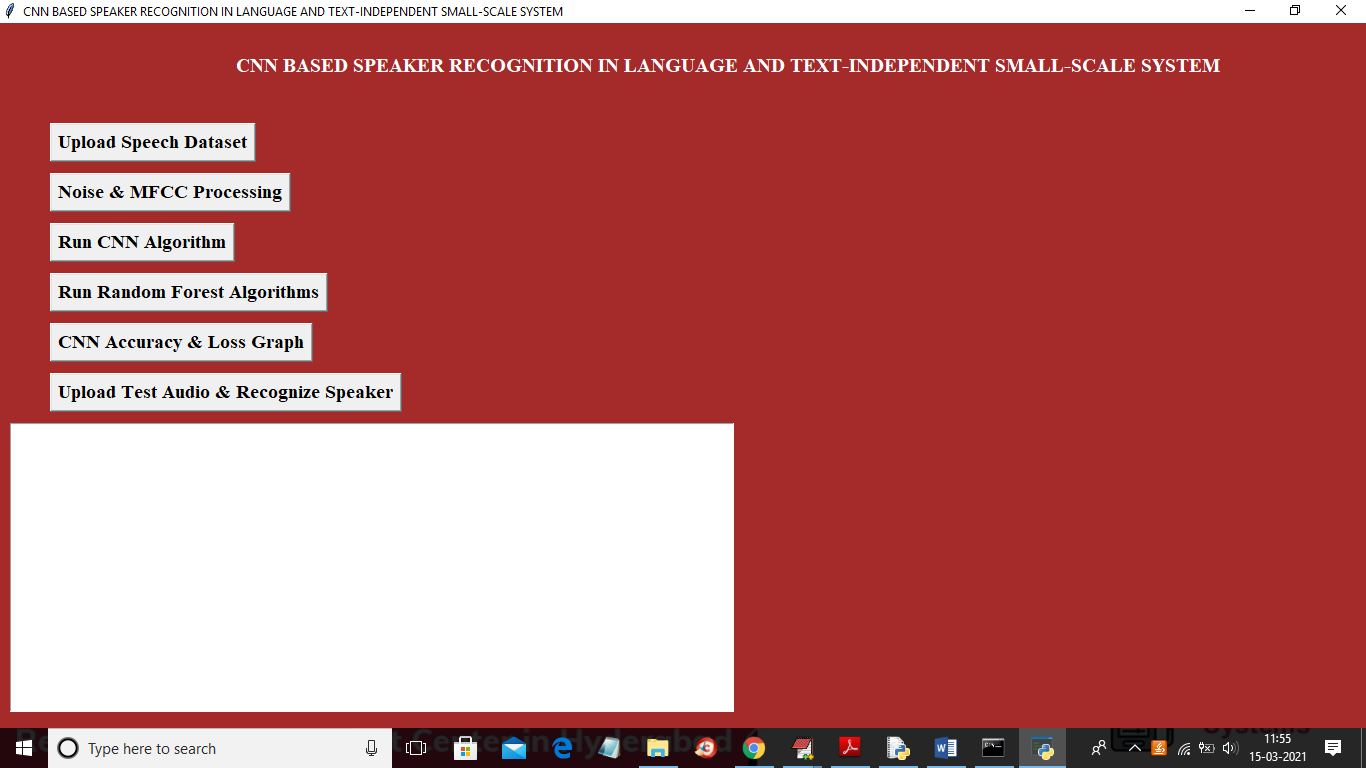
In above screen go inside any folder to see speech audio files of that person like below screen



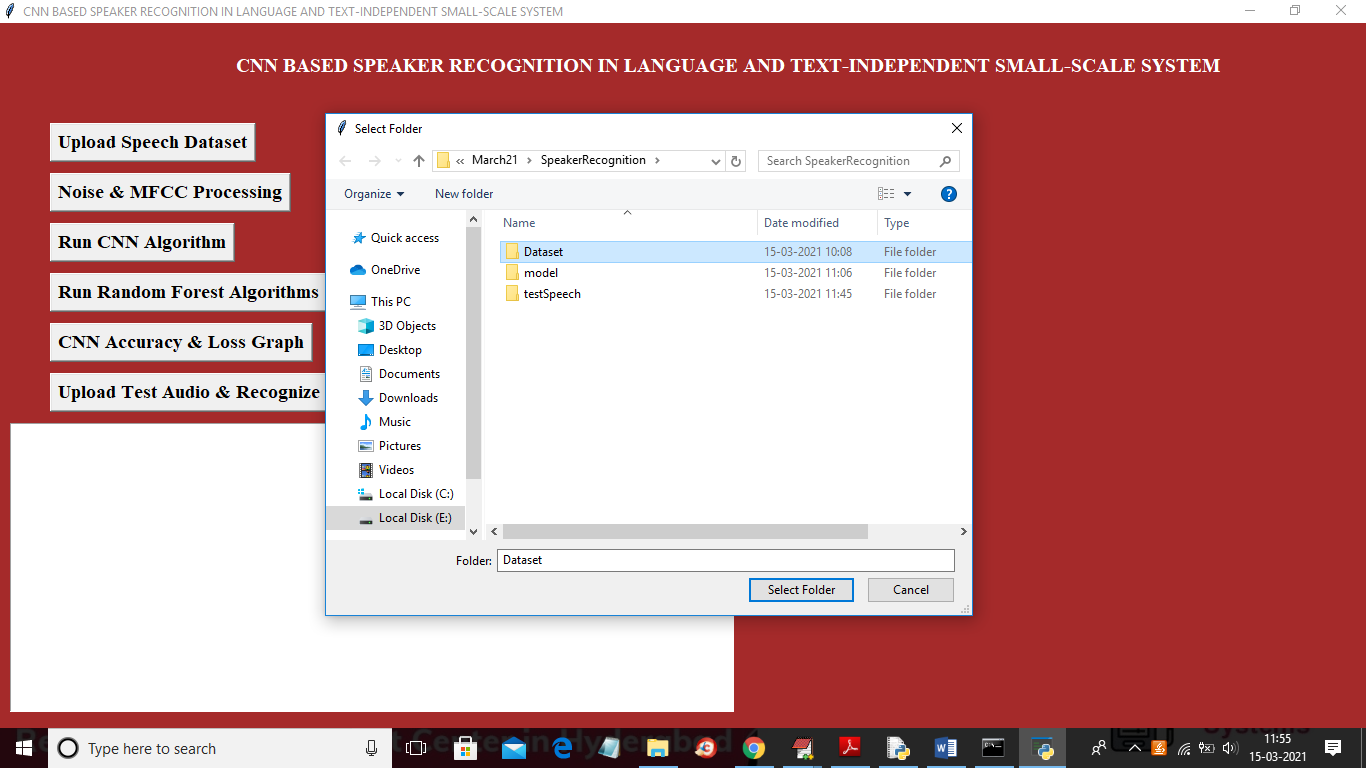
In above screen showing all audio files dataset

SCREEN SHOTS

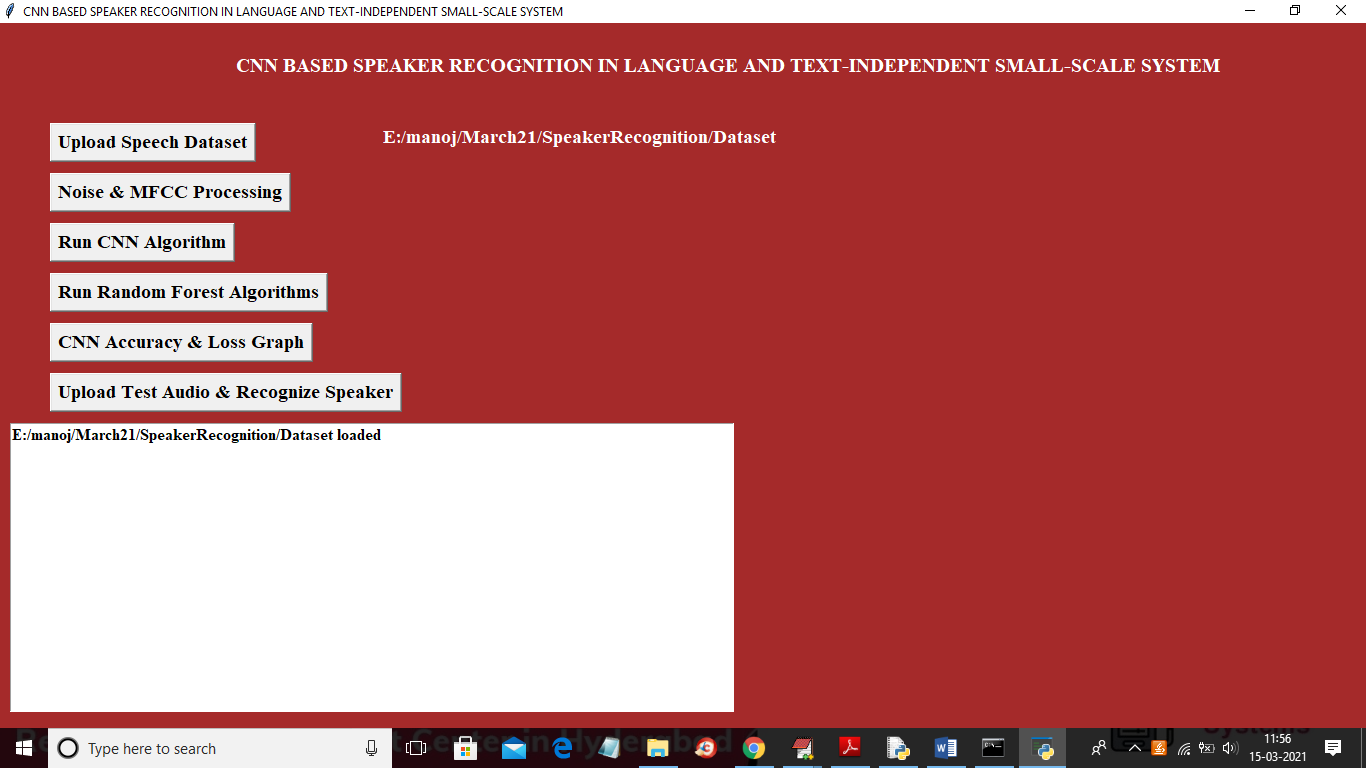
To run project double click on ‘run.bat’ file to get below screen



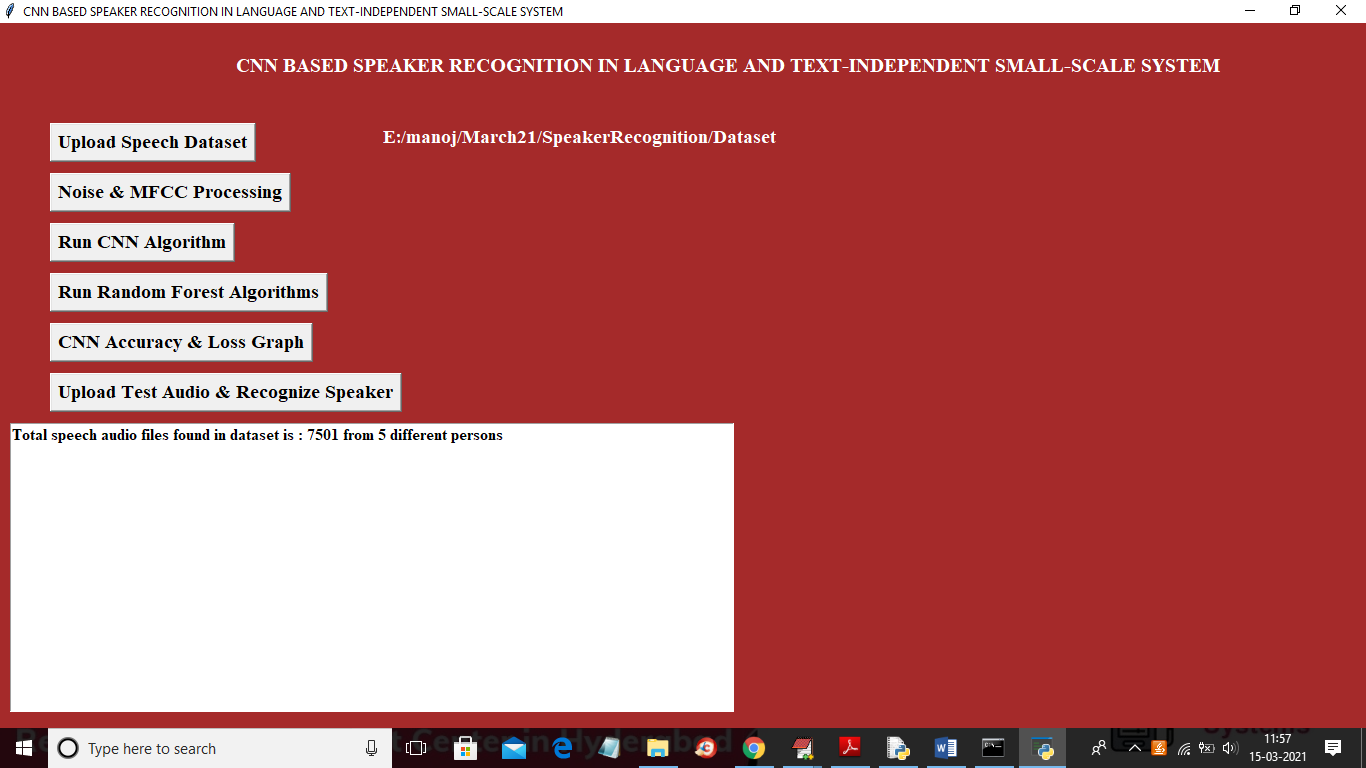
In above screen click on ‘Upload Speech Dataset’ button to upload dataset



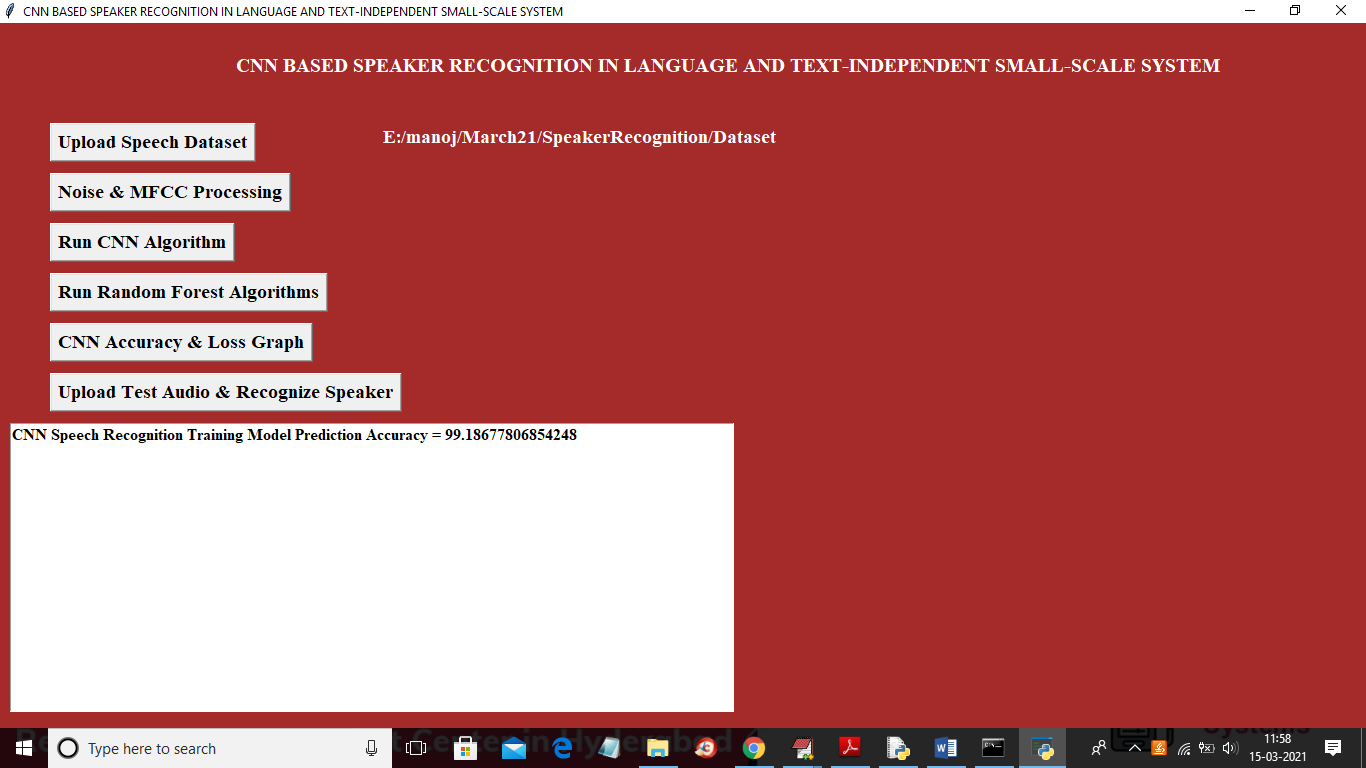
In above screen selecting and uploading ‘Dataset’ folder and then click on ‘Select Folder’ button to load dataset and to get below screen



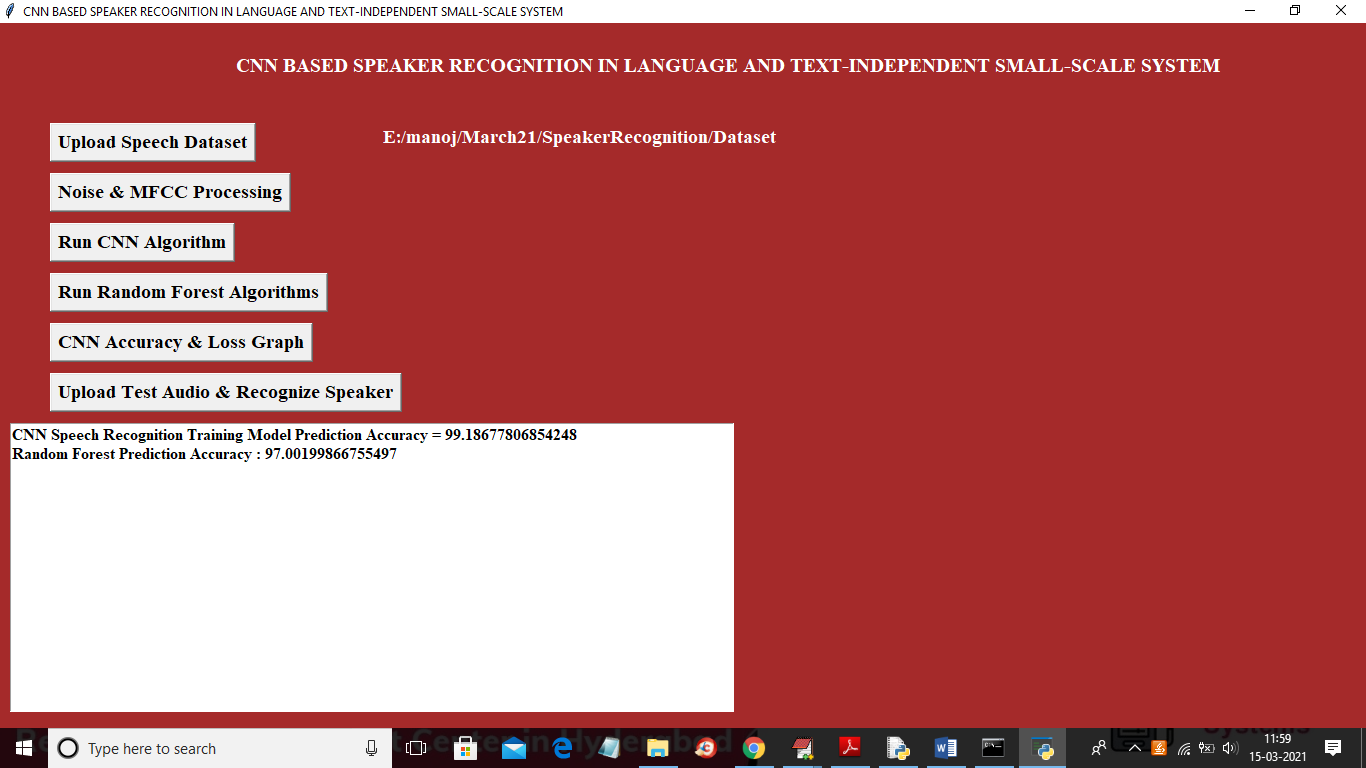
In above screen dataset loaded and now click on ‘Noise & MFCC Processing’ button to read all audio files and then remove noise and apply MFCC to extract features



In above screen application extracted features from 7501 audio files belonging to 5 different persons and now dataset features are ready and now click on ‘Run CNN Algorithm’ button to train CNN with above dataset



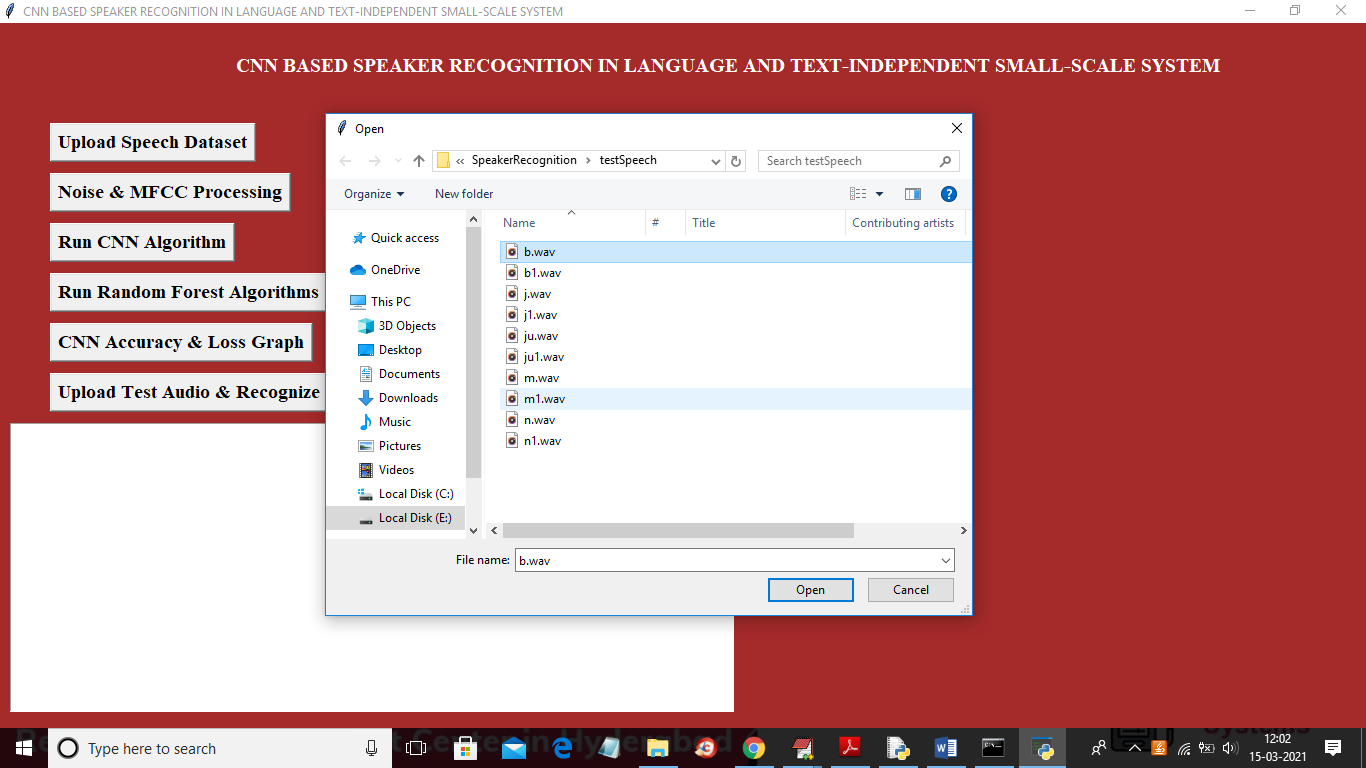
In above screen CNN training process completed and we got its accuracy as 99.18% and now click on ‘Run Random Forest Algorithm’ button to train random forest with above dataset and then calculate its accuracy



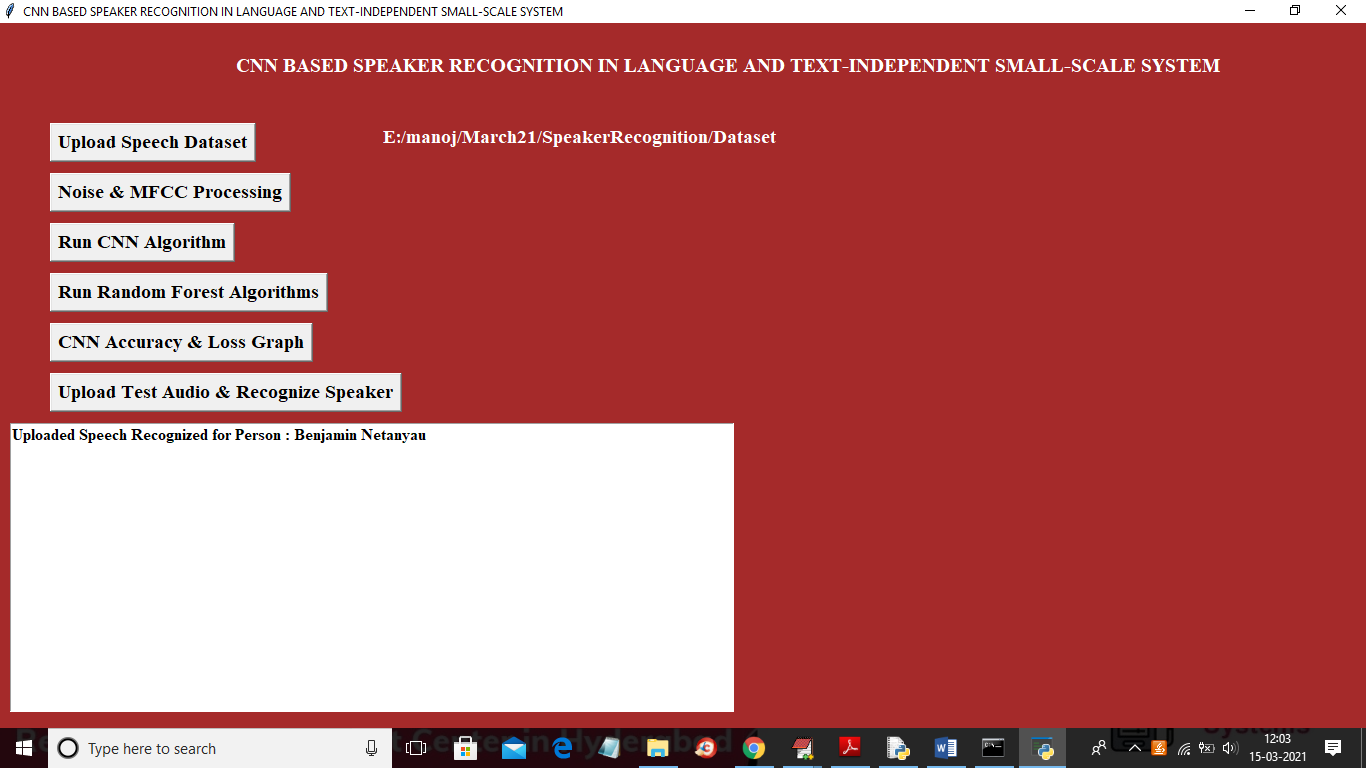
In above screen we got random forest accuracy as 97% and CNN got more accuracy than random forest and now click on ‘CNN Accuracy & Loss Graph’ button to get below graph



In above graph x-axis represents epoch/iterations and I took 10 epoch and in above graph while increasing epoch we can see accuracy get increase and loss get decrease to 0 which means model built with accurate prediction. In above graph yellow line represents accuracy and blue line represents loss. Now click on ‘Upload Test Audio & Recognize Speaker’ button to upload test speech and then CNN will recognize person from that speech



In above screen I am selecting and uploading ‘b.wav’ file and then click on ‘Open’ button to predict person from that file like below screen



In above screen in text area we can see person recognized as ‘Benjamin Netanyan’ and similarly you can upload other files and recognize and after recognize then that file will play voice

**8. CONCLUSION:**

The language independent, text independent speaker recognition system was developed with the idea that the future smart devices should have the ability to recognize their user’s voice without the boundaries of languages and keywords. An accuracy of 75-80% was achieved using the CNN model (Table 2). Further studies can be conducted to improve the accuracy of the model and to scale up the number of users. Speaker recognition can be applied to multiple domains across the industry. In the IoT world, it can change the way we interact with smart devices. Without saying “Ok Google” or “Hey Siri”, one would be able to communicate with such devices in natural language. Also, the device would be able to personalize the experience for the recognized user. Speaker recognition is also useful in biometric verification. In its current state, it can be used to build a simple lab attendance system which would not require any specialized biometric device.

**9. REFERENCES:**

[1] Zhang, Chunlei, and Kazuhito Koishida. "End-to-End TextIndependent Speaker Verification with Triplet Loss on Short Utterances." In Interspeech, pp. 1487-1491. 2017.

[2] Li, Chao, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. "Deep speaker: an end-to-end neural speaker embedding system." arXiv preprint arXiv:1705.02304 (2017).

[3] Geeta Nijhawan, Dr. M. K. Soni, July 2014, "Speaker Recognition using MFCC and Vector Quantisation", International Journal on Recent Trends in Engineering and Technology, Vol 11. No. 1

[4] Shahenda Sarhan, Mohamed Abu ElSoud, Nagham Mohammed Hasan, July 2015, "Text Independent speaker identification based on MFCC and Deep Neural Networks", <https://www.researchgate.net/publication/291165354>

[5] Ahilan Kanagasundaram, David Dean, Sridha Sridharan, Clinton Fookes, October 2016, "DNN based Speaker Recognition on Short Utterances", arXiv:1610.03190

[6] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, Zhenyao Zhu, 5 May 2017, "Deep Speaker: an End-to-End Neural Speaker Embedding System",arXiv:1705.02304v1 [cs.CL]

[7] Zhenhao Ge, Ananth N. Iyer, Srinath Cheluvaraja, Ram Sundaram, Aravind Ganapathiraju, 7-8 September 2017. "Neural Network Based Speaker Classification and Verification Systems with Enhanced Features", Intelligent Systems Conference 2017

[8] Fred Richardson, Douglas Reynolds, Najim Dehak, October 2015, “Deep Neural Network Approaches to Speaker and Language Recognition”, IEEE Signal Processing Letters, Vol 22, No. 10

[9] Amirsina Torfi, Nasser Nasrabadi, Jeremy Dawson, 6 November 2017, “Text-Independent Speaker Verification Using 3D Convolutional Neural Networks”, arXiv:1705.09422v4

[10] Zhang, Shi-Xiong, Zhuo Chen, Yong Zhao, Jinyu Li, and Yifan Gong. "End-to-end attention based text-dependent speaker verification." In 2016 IEEE Spoken Language Technology Workshop (SLT), pp. 171-178. IEEE, 2016.