National College of Ireland

Configuration Manual

Classification of Human Age Group by Implementing Deep Learning Models on Audio Data

MSc Research Project
MSc in Data Analytics

Srijan Kumar Pandey
Student ID: X18127312

School of Computing
National College of Ireland

Supervisor: Dr. Rashmi Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Srijan Kumar Pandey……
Student ID:	x18127312……
Programme:	MSc in Data Analytics…… **Year:** 2019-2020..
Module:	Research Project……
Lecturer:	……
Submission Due Date:	……
Project Title:	Classification of Human Age Group by Implementing Deep Learning Models on Audio Data……
Word Count:	2171…… **Page Count:** 9……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Srijan Kumar Pandey……

Date: 28th September 2020……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual: Classification of Human Age Group by Implementing Deep Learning Models on Audio Data

Srijan Kumar Pandey
Student ID: x18127312

1 Introduction

A configuration manual can be explained as the characteristics set used while carrying out research. It contains information about the hardware specifications, software specifications, and detailed information on versions of our libraries, tools, and techniques used for the successful completion of our project. It also gives information about the types of computer systems used, the servers we're working on, software and programming language used. The research was aimed at building a model that can classify the human age group using audio data. The research was motivated by the vast application of audio classification in areas like recommender systems, security systems, and speech analytics. The model was trained using CNN and RNN-LSTM with time distributed layers. The performance of both models is compared through parameters like accuracy. In this section of the report, we'll discuss the physical and functional attributes of our work.

2 Specifications

Our hardware and software specifications play a very important role in our project. The time taken to complete a project is directly dependent on these two factors. If we have a basic configuration and our data size is big it will take a lot of time to execute. The data on which I was working on contained 2138 audio recordings whose total size was 1 GB. Now we'll go ahead and see the hardware specifications of my system in the table below.

Table 1: system specifications

System	Specification
Windows edition	Windows 10 Home
Processor	Intel core i5 @ 1.60GHz
RAM	8 GB
System type	64-bit operating system

As we can see in **table 1** my system is a Windows 10 64-bit operating system with an Intel Core i5 processor and 1.60GHz processor. Due to the processor being not so powerful and considering the size of the data the research was planned to perform in Google Colaboratory.

2.1 Google colaboratory

Google colaboratory is a machine learning environment that is created by Google to perform machine learning operations. It is free to use and work on the cloud which means it is not completely dependent on our system hardware. It also offers GPU which I do not have in my hardware. The reason to perform the work in google colaboratory was that it is fast and requires no setup to use. I used google colaboratory GPU to perform my research. To open google colab we need to go to the link [1] and then click on the file. The next step is to click on a file and then click on New notebook.

This will give us a new notebook and then we'll see how to use GPU in colab in the next segment.

The first step after we open our new notebook is to click on Runtime and then click on Change Runtime type.

This will open a window where you can select GPU.

3 Dataset Information

The dataset was obtained from [2] where we had 2138 audio samples in mp3, A CSV files where demographics details for those 2138 speakers were saved, and a .txt file which had a sentence spoken by people of different linguistic backgrounds in English.

4 Configuration of Google Colab

As mentioned above the research was performed in google colab so below are some specifications which were used

Google Collaboratory offers free 12.72 GB of RAM and 107.77 GB of disk space and coding was done using Python version 3.

To code through google Collaboratory the dataset must be first uploaded to google drive. Once the data is uploaded to google drive we have to connect our colab with google drive and it is done by the codes shown below.

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

[1] https://colab.research.google.com/notebooks/intro.ipynb
[2] https://www.kaggle.com/rtatman/speech-accent-archive

Once we get the link we have to click on it. Select the account with which I want to connect my colab. This would be the same account on which the dataset was uploaded. It will generate a code that we have to copy and paste here and then press enter. This will connect colab to our google drive.
The next step is to mount our dataset with colab which is done by following codes.

```
from google.colab import drive
drive.mount('/Research')
```

We have to follow the same steps as above and our dataset will be mounted with colab.

5 List of Libraries used along with its versions

There were multiple libraries used which are shown below along with their versions in **table 2**.

Table 2: Libraries and their versions used

Libraries	Versions
sklearn	0.22.2.post1
pandas	1.0.5
NumPy	1.18.5
Matplotlib	3.2.2
IPython	5.5.0
Librosa	0.6.3
scipy	1.4.1
Keras	2.4.3
seaborn	0.10.1
Tensorflow	2.3.0

The research also required the installation of a few packages which are as follows:

```
pip install pydub

pip install python_speech_features

pip install audiomentations
```

Now we will move forward and look at all the libraries used and for that, I am attaching all my codes here

```
import pandas as pd
import os
import math
import numpy as np
import matplotlib.pyplot as plt
import IPython.display as ipd
import librosa as lr
```

```
import librosa.display
from glob import glob
from tqdm import tqdm
import scipy.io
from os.path import dirname, join as pjoin
from scipy.io import wavfile
from pydub import AudioSegment
from python_speech_features import mfcc, logfbank
from sklearn.model_selection import train_test_split
from audiomentations import Compose, AddGaussianNoise, TimeStretch, PitchShift, Shi
ft
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten
from keras.layers import Dropout, TimeDistributed
from keras import regularizers
from keras.callbacks import EarlyStopping
from keras.models import Model
from keras.preprocessing import image
from keras.models import Sequential
from keras.utils import plot_model
from keras.models import load_model
import tensorflow as tf
from keras.utils import to_categorical
from sklearn.utils.class_weight import compute_class_weight
import keras
from keras import layers
from keras.layers import Convolution2D, MaxPooling2D, Dense, Flatten, Activation
from keras.layers import BatchNormalization
from keras import applications
from keras.callbacks import ReduceLROnPlateau
import pickle
from keras.callbacks import ModelCheckpoint
from sklearn.metrics import roc_curve, roc_auc_score, auc
from sklearn.metrics import  average_precision_score, precision_recall_curve
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
, classification_report, confusion_matrix
from sklearn.metrics import mean_absolute_error
import seaborn as sns
from sklearn.metrics import mean_squared_error
```

Now we will go ahead and load our dataset using

```
df = pd.read_csv("/Research/My Drive/Research/speakers_all.csv", header=0)
```

The dataset initially had 3 null columns that were removed and then 32 records which didn't have any corresponding audio and those records were removed as well using the following codes.

```
df.drop(df.columns[9:12],axis = 1, inplace = True)
```

```
df['file_missing?'].value_counts()
```

```
False    2140
True       32
Name: file_missing?, dtype: int64
```

```
audio = glob('/Research/My Drive/Research/recordings/recordings/*')
len(audio)
```

```
2138
```

The above code shows that we have 2138 audio files in recordings but the dataset had 2140 rows which mean we still have 2 rows for which we do not have a corresponding audio file. So those two records were removed so that we have recordings of all the entries in our dataset. This entire operations code is shown below

```
# Filename present in CSV but actual files not found
nf_act_files = []
for x in fnames_csv:
    if x not in src_files_nm:
        nf_act_files.append(x)

# Files present in the folder but records not found in provided CSV
nf_file_record = []
for x in src_files_nm:
    if x not in fnames_csv:
        nf_file_record.append(x)
len(nf_act_files), len(nf_file_record)
```

```
(2, 0)
```

Since all the libraries have been properly called and some processing work is done on our CSV file so now, we will go ahead and have a look at the Research folder to see its content.

```
print(os.listdir("/Research/My Drive/Research"))
```

```
['reading-passage.txt',    'speakers_all.csv',    'recordings',    'wavrecordings',
'.ipynb_checkpoints', 'vid_len_df.csv', 'top_lang_df.csv', 'new_df.csv', 'pickles',
'Models', 'inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5']
```

In the directory listing we see here the original file came with only reading passage.txt, speakers_all.csv, and recordings. The recordings had 2138 audio samples which were in .mp3 format which needed to be converted into wav files (The reason is mentioned in detail in the report) so for that, a wavrecordings folder was created to store .wav format of those mp3 files which was converted by us using AudioSegment. The code for the conversion is shown below

```
dst_folder = '/Research/My Drive/Research/wav recordings'
 for el in src_files:
    AudioSegment.from_mp3(el).export(el.replace(src_folder, dst_folder)
.replace('.mp3', '.wav'), format='wav')
```

Vid_len_df.csv was created which stored the length of our audio files and then it was stored in the form of CSV for future use. Similarly, new_df.csv stored the clean audio file length information which was created after applying a signal envelope.
Both these CSV files had an extra column with the same name where the audio lengths were stored.

In the next section, we will go ahead and explore our CSV file and try to study what the file has to offer. Some of the EDA codes are shown below.

```
df.groupby("native_language")['age'].describe().sort_values(by=['count'],ascending=
False)

df.groupby("age")['sex'].describe().sort_values(by=['count'],ascending=False)

df.groupby('sex')['country'].describe().sort_values(by=['count'],ascending=False)
```

By performing the exploration and checking the distribution of languages sex and ages across the data it was decided to predict the age group of individuals based on their voice recordings because this would have a lot of applications in the real world and we were able to use all our data to perform this operation. For this purpose, the age column was categorized into four categories with this condition.

```
conditions = [
    (new_df['age'] >=5 ) & (new_df['age'] <= 25),
    (new_df['age'] < 5 ),
    (new_df['age'] >= 41),
    (new_df['age'] > 25) & (new_df['age'] < 41)]
choices = ['Gen Z', 'Alpha', 'Baby Boomers', 'Millennials']
new_df['age_group'] = np.select(conditions, choices, default='NA')
new_df.head()
```

As a result of the above transformation, this condition was applied in our dataset and a new column was created with the name of age_group which was categorized with the name of generations for the corresponding age group.

The next step was creating our X features which hold our audio data characteristics and y matrix for labels. The code for getting X and y features are

```
def build_random_feat():
  # tmp = check_data()
  # if tmp:
  #   return tmp.data[0], tmp.data[1]
  X = []
  y = []
  _min, _max = float('inf'), -float('inf')
  for _ in tqdm(range(n_samples)):
    rand_class = np.random.choice(class_dist.index, p=prob_dist)
    file = np.random.choice(new_df[new_df.age_group==rand_class].index)
    path = os.path.join("/Research/My Drive/Research/recordings/Clean",file+".wav")
    rate, wav = wavfile.read(path)
```

```
        age_group = new_df.at[file, 'age_group']
        rand_index = np.random.randint(0, wav.shape[0]-int(rate))
        sample = wav[rand_index:rand_index+int(rate)]
        X_sample = mfcc(sample, rate, numcep=13, nfilt=26, nfft=512)
        _min = min(np.amin(X_sample), _min)
        _max = max(np.amax(X_sample), _max)
        X.append(X_sample)
        y.append(classes.index(age_group))
    X, y = np.array(X), np.array(y)
    X = (X- _min) / (_max - _min)
    X = X.reshape(X.shape[0], X.shape[1], X.shape[2], 1)
    y = to_categorical(y, num_classes=4)
    # data = (X, y)
    # with open(pickles_path, 'wb') as handle:
    #   pickle.dump(handle, protocol=pickle.HIGHEST_PROTOCOL)
    return X, y
```

As the project progresses our y label was created using the age_group column and was used for classification. After this step, our X and y features were created from audio data, and then it was passed through CNN and RNN-LSTM with time distributed layers. The entire process has been explained in detail in the project report. We will see the results after the data is trained with these two models here

Table 3: Models used and their results

Metrics \ Models	**CNN**	**RNN-LSTM**
Accuracy	62.45%	66.07%
Precision	62.45%	66.07%
Recall	62.45%	66.07%
F1 Score	62.45%	66.07%

RNN-LSTM with time distributed layer gave higher accuracy than Fully Connected CNN as shown in table 3. The value for accuracy, precision, recall and F1 score is same for both CNN and RNN-LSTM. The execution time for RNN-LSTM model was faster than that of CNN. Adding more data and extra layers for both would help in gaining a higher accuracy.