Configuration Manual

MSc Research Project
MSc Cyber-security

Atharva Lawate
Student ID: x22213325

School of Computing
National College of Ireland

Supervisor: Eugene Mclaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …….Atharva Lawate…….……………………………………………………………

Student ID: ……x22213325………………………………………………………………………….……

Programme: …MSc Cyber Security…………………………… **Year:** …2023-24…..

Module: …MSc Research Practicum……………………………………….………

Lecturer: …Eugene Mclaughlin……………………………………………………….………

Submission Due Date: …September 16, 2024……………………………………………………………….………

Project Title: …ENHANCING VOICE AUTHENTICATION SYSTEMS WITH DEEPFAKE AUDIO DETECTION…

Word Count: ……876.…………………………… **Page Count:** ……6…………….…….………

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: ……Atharva Lawate…………………………………………………………………

Date: ……September 16, 2024…………………………………………………………………

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

Atharva Lawate
X22213325

1. Introduction

The proliferation of voice authentication features has changed security across industries, from banking to smart devices. However, with the rise of deep learning technologies, the emergence of DeepFake audio poses a serious threat to these systems. DeepFake audio uses advanced AI techniques to create artificial voices that are nearly indistinguishable from real human voices. This can be used to trick voice authentication, allowing unauthorized access and security breaches.
To address this critical issue, the project "Enhancing Voice Authentication Systems with DeepFake Audio Detection" aims to integrate robust DeepFake detection mechanisms into existing voice authentication frameworks Leveraging state-of-the-art machine learning models, especially deep learning-based approaches and trying to increase

2. Overview of the program

These projects include the development and implementation of a machine learning model specifically designed to detect DeepFake audio in an acoustic reliability framework. The main features of the project are as follows.

Data Collection and Priorities:
Collect accurate data with real and DeepFake audio samples.
Perform data preprocessing steps including feature extraction (e.g., MFCCs, chroma features) and data enhancement to improve the robustness of the model.

Positive Progress:
Develop deep learning algorithms such as long-term and short-term memory networks (LSTM) to analyze temporal features of audio signals.
Use additional layers including Dense and Dropout layers to enhance the model's ability to distinguish between real and artificial tone.

3. Hardware/software requirements

3.1. Hardware production

The following hardware settings are recommended for smooth operation.

- Processor: Intel Core i7
- RAM: 16 GB
- Storage: 100 GB free space
- GPU (optional, for fast model training): NVIDIA GTX 1050 Ti or better

3.2. Software

The following software is required for the project:

- Jupyter Notebook: Used to run and write code.
- Anaconda: To manage the Python environment and dependencies.

Version Requirements:

- Jupyter Notebook: Version 6.0 or later
- Anaconda: Version 2020.11 or later

Ensure that the necessary Python libraries are installed. These include panda, numpy, mutplotlib, seaborn, scikit-sua, and tensorflow.

4. The data set

The dataset used for this project came from Kaggle.
The dataset is downloaded from Kaggle and add it to the balanced dataset directory in project folder.

5. Implementation of Project

1. Importing Libraries:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.utils import to_categorical
import seaborn as sns
```

Figure 1: Import Libraries

- pandas, numpy: libraries for data manipulation and statistical computation.
- LabelEncoder, StandardScaler: Preprocessing tools for label encoding and scaling features.
- train_test_split: Function for splitting data into training and testing.
- Sequential, LSTM, Cubic, Dropout: Keras classes for building and training LSTM models.
- to_categorical: Convert integer labels to hot-encoded format for categorical classification.
- seaborn, matplotlib.pyplot: Libraries for data visualization.

2. Data Preprocessing:

Data reading

```
df = pd.read_csv('C:/Users/Bunny/Downloads/DATASET-balanced.csv')
```

```
df
```

	chroma_stft	rms	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4	...	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17
0	0.338055	0.027948	2842.948867	4322.916759	6570.586186	0.041050	-462.169586	90.311272	19.073769	24.046888	...	-6.686564	0.902086	-7.251551	-1.198342	4.747403	-4.986279
1	0.443766	0.037838	2336.129597	3445.777044	3764.949874	0.047730	-409.413422	120.348808	-7.161531	5.114784	...	-2.131157	-6.876417	-1.359395	0.326401	-5.420016	-2.109968
2	0.302528	0.056578	2692.988386	2861.133180	4716.610271	0.080342	-318.996033	120.490273	-24.625771	23.891073	...	-5.853725	-3.724773	-6.627182	-5.117002	-6.072106	-0.994653
3	0.319933	0.031504	2241.665382	3503.766175	3798.641521	0.047180	-404.636749	136.320908	2.308172	-3.907071	...	-1.898315	-2.046493	-7.176277	-3.293508	4.209121	0.121835
4	0.420055	0.016158	2526.069123	3102.659519	5025.077899	0.051905	-410.497925	152.731400	-18.266771	51.993462	...	-1.952340	0.810868	6.238493	6.555839	7.535542	2.849219
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11773	0.435426	0.025303	2772.575031	2728.757601	4998.670213	0.074323	-342.309753	144.490418	-79.272942	8.890874	...	-17.982819	-7.831161	-1.127167	-7.669674	-0.653850	-8.037575
11774	0.454611	0.070578	1029.274601	1519.231563	1922.927486	0.026553	-332.230408	202.603012	-0.181929	-2.146542	...	-2.018668	-2.705635	-1.589172	-2.938737	-0.972690	-1.706672
11775	0.374432	0.019063	4063.645317	3558.261357	7299.133512	0.110278	-372.149109	92.670235	-29.082432	59.736637	...	-6.628118	-3.827499	-7.287946	-2.899543	-11.508186	-1.296590
11776	0.410885	0.090499	1124.655596	1553.651133	2065.942806	0.031761	-328.062805	193.557526	6.779151	-1.304731	...	-5.437202	-4.252508	-1.258683	-2.107233	-1.018154	-2.716950
11777	0.570581	0.033022	2275.915286	3566.472303	4054.590126	0.052983	-398.812103	129.081482	3.654665	18.412062	...	4.484940	3.916474	2.186594	1.218625	3.607651	4.629877

11778 rows × 27 columns

Figure 2: Data Preprocessing

- df = pd.read_csv(...): Retrieves a data structure from the specified CSV file.

- df.info(), df.describe(), df.head(): Displays basic information about the data structure, such as number of entries, colors, summary statistics, and a few characters.

3. Handling Missing Values:

Null value removing

```
df = df.dropna()
```

```
df
```

	chroma_stft	rms	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4	...	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	
0	0.338055	0.027948	2842.948867	4322.916759	6570.586186	0.041050	-462.169586	90.311272	19.073769	24.046888	...	-6.686564	0.902086	-7.251551	-1.198342	4.747403	-4.986279	C
1	0.443766	0.037838	2336.129597	3445.777044	3764.949874	0.047730	-409.413422	120.348808	-7.161531	5.114784	...	-2.131157	-6.876417	-1.359395	0.326401	-5.420016	-2.109968	-1
2	0.302528	0.056578	2692.988386	2861.133180	4716.610271	0.080342	-318.996033	120.490273	-24.625771	23.891073	...	-5.853725	-3.724773	-6.627182	-5.117002	-6.072106	-0.994653	-1
3	0.319933	0.031504	2241.665382	3503.766175	3798.641521	0.047180	-404.636749	136.320908	2.308172	-3.907071	...	-1.898315	-2.046493	-7.176277	-3.293508	4.209121	0.121835	-5
4	0.420055	0.016158	2526.069123	3102.659519	5025.077899	0.051905	-410.497925	152.731400	-18.266771	51.993462	...	-1.952340	0.810868	6.238493	6.555839	7.535542	2.849219	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11773	0.435426	0.025303	2772.575031	2728.757601	4998.670213	0.074323	-342.309753	144.490418	-79.272942	8.890874	...	-17.982819	-7.831161	-1.127167	-7.669674	-0.653850	-8.037575	-2
11774	0.454611	0.070578	1029.274601	1519.231563	1922.927486	0.026553	-332.230408	202.603012	-0.181929	-2.146542	...	-2.018668	-2.705635	-1.589172	-2.938737	-0.972690	-1.706672	-2
11775	0.374432	0.019063	4063.645317	3558.261357	7299.133512	0.110278	-372.149109	92.670235	-29.082432	59.736637	...	-6.628118	-3.827499	-7.287946	-2.899543	-11.508186	-1.296590	-14
11776	0.410885	0.090499	1124.655596	1553.651133	2065.942806	0.031761	-328.062805	193.557526	6.779151	-1.304731	...	-5.437202	-4.252508	-1.258683	-2.107233	-1.018154	-2.716950	-3
11777	0.570581	0.033022	2275.915286	3566.472303	4054.590126	0.052983	-398.812103	129.081482	3.654665	18.412062	...	4.484940	3.916474	2.186594	1.218625	3.607651	4.629877	C

11778 rows × 27 columns

```
print(df.columns)
```

```
Index(['chroma_stft', 'rms', 'spectral_centroid', 'spectral_bandwidth',
       'rolloff', 'zero_crossing_rate', 'mfcc1', 'mfcc2', 'mfcc3', 'mfcc4',
       'mfcc5', 'mfcc6', 'mfcc7', 'mfcc8', 'mfcc9', 'mfcc10', 'mfcc11',
       'mfcc12', 'mfcc13', 'mfcc14', 'mfcc15', 'mfcc16', 'mfcc17', 'mfcc18',
       'mfcc19', 'mfcc20', 'LABEL'],
      dtype='object')
```

Figure 3: Handling Missing Values

- df = df.dropna(): Removes any rows in the data set that contain missing values.

4. Exploratory Data Analysis (EDA):

```
from matplotlib import pyplot as plt
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='LABEL')

for p in plt.gca().patches:
    plt.text(p.get_x() + p.get_width() / 2, p.get_height() + 0.1, int(p.get_height()),
             ha='center', va='bottom')

plt.show()
```

```
[ ]: scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

```
[ ]: X_reshaped = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))
```

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X_reshaped, y_categorical, test_size=0.2, random_state=42)
```

```
[ ]: model = Sequential()
     model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True))
     model.add(Dropout(0.5))
     model.add(LSTM(64))
     model.add(Dropout(0.5))
     model.add(Dense(32, activation='relu'))
     model.add(Dense(y_categorical.shape[1], activation='softmax'))

     model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
correlation_matrix = df.corr()
plt.figure(figsize=(20, 8))
sns.heatmap(correlation_matrix, cmap='coolwarm', annot = True)
plt.title('Correlation Matrix')
plt.tight_layout()
plt.show()
```

Figure 4: Exploratory Data Analysis (EDA)

- sns.countplot(…): Produces the number of observations in each category (LABEL column).
- This loop adds labels to the bars in the count plot indicating the number of instances of each class.

- The correlation is computed using the pandas library and heatmap is generated using seaborn.

5. Label Encoding and Feature Selection:

```
[ ]: label_encoder = LabelEncoder()
     y_encoded = label_encoder.fit_transform(y)
     y_categorical = to_categorical(y_encoded)
```

Figure 5: Label Encoder

- label_encoder.fit_transform(...): Transforms categorical labels into numeric values.
- y = df['LABEL']: Remove target variable (LABEL).
- X = df.drop('LABEL', axis=1): Drops attributes from the target variable.
- This loop creates a histogram for each feature in X, defined by the LABEL class, to visualize the distribution of features among the classes.

6. Data Scaling and Reshaping:

```
[ ]: scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)

[ ]: X_reshaped = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))
```

Figure 6: Data Scaling and Reshaping

- StandardScaler: It standardizes features by subtracting the mean and scaling to the unit variance.
- X_scaled: Moves the attribute matrix.
- X_reshaped: Reshapes the attribute matrix appropriately for the LSTM input (samples, time steps, attributes).

7. Train-Test Split:

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X_reshaped, y_categorical, test_size=0.2, random_state=42)
```

Figure 7: Train-Test Split

- train_test_split(…): Splits the data into training and test sets (80% train, 20% test).

8. Building the LSTM Model:

```
[ ]: model = Sequential()
     model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True))
     model.add(Dropout(0.5))
     model.add(LSTM(64))
     model.add(Dropout(0.5))
     model.add(Dense(32, activation='relu'))
     model.add(Dense(y_categorical.shape[1], activation='softmax'))

     model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 8: Building LSTM Model

- Sequence: Starts the sequence order.
- LSTM: Adds an LSTM layer of 64 units each.
- Dropout: Adds a dropout layer to prevent overloading.

- Condensed: Adds fully connected layers, with the last layer using softmax activation for multiclass classification.

9. Compiling and Training the Model:

```
[ ]: history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

Figure 9: Compiling and Training Model

- model.compile(…): Compile the model for training with Adam optimizer and categorical cross-entropy loss.
- model.fit(…): Trains the model on the training data, with 20% used for validation.

10. Evaluating the Model:

```
[ ]: loss, accuracy = model.evaluate(X_test, y_test)
     print(f'Test Accuracy: {accuracy:.2f}')
```

```
# Display the metrics
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
```

Figure 10: Evaluating model

- model.evaluate(…): Evaluates how well the trained model performs on unseen data that is test set.
- Along with this model was also evaluated with precision, recall and F1 score metrics.

11. Plotting Training and validation History:

```
[ ]: import matplotlib.pyplot as plt

     # Plot training & validation accuracy values
     plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('Model accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Validation'], loc='upper left')
     plt.show()

     # Plot training & validation loss values
     plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('Model loss')
     plt.ylabel('Loss')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Validation'], loc='upper left')
     plt.show()
```

Figure 11: Plotting Training and Validation Accuracy and Loss Values

- These plots show the accuracy and losses in the model during the training phase, and compare training and validation performance.'

References:

Chen, T., Kumar, A., Nagarsheth, P., Sivaraman, G. and Khoury, E., 2020, November. Generalization of Audio Deepfake Detection. In *Odyssey* (pp. 132-137).

Dixit, A., Kaur, N. and Kingra, S., 2023. Review of audio deepfake detection techniques: Issues and prospects. *Expert Systems*, *40*(8), p.e13322.
Almutairi, Z. and Elgibreen, H., 2022. A review of modern audio deepfake detection methods: challenges and future directions. *Algorithms*, *15*(5), p.155.