International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

AudiForgery Detection Using LFCC and Deep Neural Networks

Mr. Adwait Ashish Sonawane[1], Mr. Kunal Kishor Turankar[2], Ms. Agasti Anil Chavan[3], Prof. Avinash Taskar[4]

[1,2,3,4]Department of Computer Science and Engineering, Sandip University, Nashik

Abstract

The rapid advancement of digital audirecording devices, mobile applications, and the Internet of Things (IoT) facilitates the collection and analysis of speech and audidata for applications such as voice authentication and smart embedded systems. However, the widespread availability of sophisticated audiediting tools, including Adobe Audition and various mobile apps, increases the risk of auditampering for malicious purposes. This presents a critical challenge, especially in sensitive contexts such as legal proceedings, where manipulated audimay be presented as genuine evidence. Due tthe cost and complexity of digital watermarking and signature-based techniques, most real-world audirecordings lack embedded authentication and remain vulnerable tforgery. Our implemented system focuses on developing an audiforgery detection system capable of identifying tampering in audifiles, including copy-move and splicing forgeries. The system leverages linear-frequency cepstral coefficients (LFCCs) and machine learning techniques tdetect manipulations with high accuracy and reliability. It outlines the functional requirements, system architecture, key methodologies, and the integration of deep learning models for tampering detection. The approach alsaddresses the challenges of detecting forgeries in realworld noisy environments and evaluates system performance using metrics such as accuracy, precision, and recall. Future directions include optimizing model inference and addressing emerging forgery techniques.

Keywords: Audiforgery detection, copy-move forgery, audisplicing, machine learning, LFCC, real-time inference

1. Introduction

PROJECT IDEA

In recent years, the evolution of digital audirecording and editing technologies has greatly facilitated the manipulation of audicontent. Sophisticated audiediting tools, such as Adobe Audition and mobile apps, have made it possible for anyone with minimal expertise ttamper with audirecordings. This is particularly concerning in sensitive domains like legal proceedings, journalism, law enforcement, and forensics, where the authenticity of audievidence is paramount. Malicious alterations, including copy-move and splicing forgeries, can be performed seamlessly without leaving any perceptible traces, making it increasingly difficult for the human ear tdiscern tampered content from the original. As audirecordings continue tbe used as critical evidence in various sectors, the need for an advanced detection system that can reliably identify such manipulations becomes crucial. • We tend tfocuses on developing an intelligent system for detecting audiforgeries, particularly copy-move and splicing manipulations. These types of tampering are

among the most common and involve either copying and relocating segments within the same audifile (copy-move forgery) or combining segments from different sources intone recording (splicing). While these alterations can significantly change the semantic content or meaning of a recording, the forgeries are often subtle enough tgunnoticed by conventional methods. Therefore, a more sophisticated approach, combining signal processing techniques and machine learning models, is necessary tensure the integrity of audifiles. • The implemented system employs a multi-stage approach for tampering detection, leveraging state-of-the-art methods for feature extraction and pattern recognition. Central tthis system are linear-frequency cepstrum coefficients. which are used textract distinguishing features from the audisignal. LFCC is particularly suited for audianalysis as it models the human auditory system's perception of sound by converting the signal's time domain intits frequency domain. This permits for effective detection of subtle changes in the audispectrum that may indicate tampering. • By addressing these critical challenges, we aim tcontribute tthe development of a secure and reliable framework for audiforgery detection, ensuring that digital audiremains a trustworthy medium in today's.

Motivation of the project

The rapid proliferation of digital auditechnologies and editing tools has transformed the way we produce, share, and consume audicontent. While these advancements have enriched various fields—such as entertainment, communication, and education—they have alsopened the door tpotential misuse. The ability teasily edit and manipulate audirecordings poses significant risks, especially in sensitive contexts where authenticity is crucial. Instances of audiforgery, such as altering recorded conversations, fabricating evidence, or creating fake audiclips, have raised serious concerns across multiple sectors. In legal settings, tampered audirecordings can mislead investigations, compromise trials, and undermine the integrity of the judicial process. In journalism, manipulated audican spread misinformation and erode public trust in media sources. • Traditional methods of audiverification, such as digital watermarking or embedding signatures, often fall short in real-world applications due ttheir cost, complexity, and lack of widespread adoption. Many audirecordings, particularly those captured in real-time or on consumer-grade devices, lack embedded authentication, leaving them vulnerable tforgery. The challenge lies in developing a detection system that can effectively identify tampering without requiring prior knowledge of the original audicontent. This gap highlights the need for innovative solutions that can provide reliable audiintegrity checks, particularly in environments where the stakes are high. • The legal and forensic communities require robust tools tverify the authenticity of audievidence. Detecting tampering in audirecordings is not only a matter of technical accuracy but alsa legal imperative. As audievidence increasingly becomes part of investigations and court proceedings, the demand for reliable forensic tools that can identify alterations becomes critical. Law enforcement agencies need tbe equipped with effective systems that can quickly and accurately assess audievidence, enabling them tuphold justice and maintain public trust in legal outcomes. • In summary, the motivation behind this project stems from the urgent need tcombat the growing threat of audiforgery in an increasingly digital world. By developing a robust detection system that combines signal processing techniques with advanced machine learning models, we aim tprovide an effective solution that enhances the reliability of audicontent across various

2. Literature Review

Several studies have explored audiforgery detection using different approaches:

1. Spread-Spectrum Watermarking of AudiSignals

Category	Details
Conference/Journal	IEEE Transactions on Signal Processing
Authors	Kirovski, J., Malvar, H. S.
Review of the Papers	Introduces a method for embedding watermarks in audisignals using spread-spectrum techniques.
Description	Demonstrates imperceptible information embedding for copyright protection and authentication.
Mathematical Terms	Spread-spectrum technique, modulation, SNR

2. Digital AudiForensics: A First Practical Evaluation on Microphone and Environment Classification

Category	Details
Conference/Journal	MM and Sec'07 - Proceedings of the Multimedia and Security Workshop
Authors	Kraetzer, C., Oermann, A., Dittmann, J., Lang, A.
Review of the Papers	Evaluates microphone and environment classification methods for detecting digital audiforgeries.
Description	Implements machine learning algorithms for classifying recording environments and microphone types.
Mathematical Terms	Statistical classifiers, feature vectors, precision, recall

3. Detecting Digital AudiForgeries by Checking Frame Offsets

Category	Details
Conference/Journal	Proceedings of the 10th ACM Workshop on Multimedia and Security
Authors	Yang, R., Qu, Z., Huang, J.
Review of the Papers	Proposes a frame offset checking method tidentify audio segment manipulations.
Description	Focuses on timing differences between audiframes tdetect inconsistencies.
Mathematical Terms	Frame offset calculation, similarity measures, detection accuracy

4. Detecting Splicing in Digital Audios Using Local Noise Level Estimation

Category	Details
Conference/Journal	2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
Authors	Pan, X., Zhang, X., Lyu, S.
Review of the Papers	Presents a technique for detecting audisplicing by analyzing local noise characteristics.
Description	Quantifies noise levels tidentify unnatural transitions

	indicative of tampering.
Mathematical Terms	Statistical properties (mean, variance), local noise level estimation

5. Exposing Digital AudiForgeries in Time Domain by Using Singularity Analysis with Wavelets

Category	Details
Conference/Journal	Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security
Authors	Chen, J., Xiang, S., Liu, W., Huang, H.
Review of the Papers	Utilizes singularity analysis combined with wavelet transforms tdetect forgeries in audirecordings.
Description	Discusses how wavelet transforms reveal singular points in audisignals, indicating manipulation.
Mathematical Terms	Wavelet transforms, singularity spectrum, coefficients

3. Software Requirements

3.1 Functional Requirements

- **Tampering Detection:** Accurately classifies audifiles as either "Forgery with type" or "Real," based on detection algorithms.
- **Result Output:** Displays classification results tthe user, providing a clear indication of the system's findings.
- **Model Training and Evaluation:** Facilitates training on diverse datasets and evaluates performance using metrics such as accuracy, precision, and recall.

3.2 Non Functional Requirements

- **Availability:** We will ensure high system uptime, enabling continuous access for training and inference, with cloud infrastructure tsupport anytime access.
- **Reliability:** We will deliver consistent and accurate detection of tampered audio, minimizing false positives and false negatives for reliable performance.
- **Maintainability:** We will make it easy tupdate and maintain the system, especially the machine learning models and feature extraction methods, for long-term adaptability.
- **Security:** We will secure the system tprotect datasets and models from unauthorized access or tampering, safeguarding data and model integrity.

3.3 Constraints

- **User Interface Constraints:** We aim for a simple UI with upload functionality for file uploads, ensuring output is easy tinterpret.
- **Hardware Constraints:** Limited access thigh-end GPUs may slow down training, particularly with large datasets.
- **Software Constraints:** Dependencies on third-party libraries like librosa and TensorFlow could introduce bugs or version conflicts.
- **Operational Constraints:** While the system can operate offline after setup, handling large datasets may require cloud storage or API deployment.

- **Assumptions and Dependencies:** We assume access tlabeled datasets for training, and the system relies on libraries such as librosa and frameworks like Keras for functionality.

3.4 Hardware Requirements

1. **CPU:**

- Minimum: Intel Core i3 or AMD Ryzen 3
- Recommended: Intel Core i5/i7 or AMD Ryzen 5/7
- Details: A multi-core processor is essential for efficient data processing and model training. Higher clock speeds and more cores will enhance parallel processing capabilities, especially during training phases involving complex computations.

2. **RAM:**

- Minimum: 8 GB
- Recommended: 16 GB or more
- Details: Sufficient RAM is critical for handling large datasets and facilitating smooth operation during model training and testing. More RAM allows for larger batch sizes and reduces the likelihood of memory bottlenecks.

3. **GPU:**

- Minimum: Nvidia GTX 1350 or equivalent
- Recommended: Nvidia GTX 1660, RTX 2060, or better (e.g., RTX 3060,
- RTX 3080)
- Details: A dedicated GPU accelerates the training of deep learning models significantly. The recommended GPUs support CUDA and provide better memory bandwidth and processing capabilities, leading treduced training times and improved model performance. VRAM of at least 4GB is advisable for handling larger models and datasets.

4. **STORAGE**

- Minimum: 256 GB HDD
- Recommended: 512 GB SSD or higher
- Details: While HDDs can suffice for basic storage needs, SSDs greatly enhance read/write speeds, which is crucial when dealing with large datasets and model files. SSDs facilitate faster data access during model training and testing phases, improving overall system responsiveness. It's alsadvisable thave ample storage space for storing raw audidata, preprocessed datasets, and trained model checkpoints.

3.5 Software Requirements

Tbuild and develop the AudiForgery Detection System, the following software dependencies are essential. Each dependency serves a specific purpose and ensures the effective execution of the project.

3.5.1. Python

- Version: 3.7 or later

- Overview: Python is a high-level, interpreted programming language known for its simplicity and versatility. It is widely used in data science, machine learning, and audiprocessing applications.
- Importance: The specified version ensures compatibility with the latest libraries and features, facilitating smoother development and integration of various packages.

3.5.2. Deep Learning Libraries

- **TensorFlow:**
  - Overview: TensorFlow is an open-source library developed by Google for numerical computation and machine learning. It provides a flexible architecture for building and deploying machine learning models.
  - Importance: TensorFlow's capabilities in building deep neural networks make it an ideal choice for complex audiclassification tasks. It supports various model types, including convolutional and recurrent neural networks.
- **Keras:**
  - Overview: Keras is a high-level API for building and training deep learning models. It can run on top of TensorFlow, making it easy tconstruct neural networks.
  - Importance: Keras simplifies the model development process with its user-friendly interface, allowing for quick prototyping and experimentation.
- **PyTorch (optional):**
  - Overview: PyTorch is another popular open-source machine learning library, developed by Facebook. It is known for its dynamic computation graph, which allows for more flexibility during model training.
  - Importance: PyTorch is particularly favored in research environments for its intuitive design and ease of use, especially for tasks requiring rapid experimentation.

3.5.3. AudiProcessing Libraries

- **LibROSA:**
  - Overview: LibROSA is a Python package specifically designed for music and audianalysis. It provides tools for feature extraction, audiloading, and manipulation.
  - Importance: Key functionalities include extracting audifeatures like Mel-Frequency Cepstral Coefficients (MFCC), spectrograms, and chroma features, which are vital for audiclassification and forgery detection.
- **PyDub:**
  - Overview: PyDub is a simple and easy-to-use library for audifile manipulation. It supports various audiformats and allows for simple operations like trimming, concatenating, and exporting audifiles.
  - Importance: It can be useful for preprocessing audifiles, such as normalizing volume, changing file formats, or segmenting audidata intmanageable pieces for analysis.

3.5.4. Additional Packages

1. **NumPy:**

- Overview: NumPy is a fundamental package for numerical computing in Python. It provides support for arrays and matrices, along with a collection of mathematical functions toperate on these data structures.

- Importance: It is essential for efficient data manipulation, especially when handling large datasets of audifeatures.

2. SciPy:

- Overview: SciPy is an open-source library used for scientific and technical computing. It builds on NumPy and provides additional functionality for optimization, integration, interpolation, eigenvalue problems, and more.
- Importance: SciPy's advanced mathematical capabilities can be leveraged for more complex audianalysis and model optimization tasks.

3. Matplotlib:

- Overview: Matplotlib is a plotting library for Python that enables the creation of static, animated, and interactive visualizations.
- Importance: It is crucial for visualizing audidata, model performance metrics, and results through various types of plots (e.g., waveforms, spectrograms, confusion matrices).

3.6 Interfaces

3.6.1. User Interface (UI):

We will provide a user-friendly platform where users can upload audifiles, initiate detection, view results, and download reports easily.

3.6.2. Hardware Interface:

Our system will integrate with GPUs or cloud computing services for efficient model training and inference, ensuring optimal hardware utilization.

3.6.3. Software Interface:

We will integrate with essential libraries like librosa for feature extraction, TensorFlow/Keras for model training, and leverage APIs for seamless deploy

4. System Architecture

1. AudiForgery Detection App (Tkinter GUI)

This is the user-facing interface of the entire system.

Components:

- AudiUpload Button:

Opens file dialog tselect .wav files.

Once a file is selected, it triggers prediction (starts backend pipeline).

- Display Controls:
  - Toggle between Waveform and Spectrogram.
  - Visual markers for forgery time intervals (highlighted in red).

2. Feature Extraction (LFCC + DCT via scipy)

- Compute Spectrogram using librosa.stft:
  - Converts time-domain signal ttime-frequency domain.
  - Returns a 2D matrix of frequencies over time.

3. Log Power Spectrum:

- Applies np.log1p treduce the dynamic range (enhances subtle differences).

4. DCT (Discrete Cosine Transform):

- Performed using scipy.fftpack.dct.

- Extracts Low-Frequency Cepstral Coefficients (LFCCs).
- These are compact, time-invariant features used for classification.

Padding + Normalization

5. **Audifiles differ in duration, resulting in a different number of feature time frames.**
6. **Machine learning models need inputs of fixed shape. Actions:**
7. **Padding/Truncation:**

- Ensures all LFCC matrices have the same length (e.g., 200 frames).
- Normalization:
- Applies Standard Scaler (zermean, unit variance).
- Makes model training stable and consistent.
- Scaler is saved with joblib tbe reused during prediction.

4. Pre-trained Model Inference

- One of the pre-trained models is loaded (Dense, CNN, or LSTM trained in STAR10.py).
- These are Keras models saved using .h5 format.

Prediction:

- Model predicts one of three classes:
- 0: Real
- 1: Copy-Move Forgery
- 2: AudiSplicing

5. Forged Region Detection (Hardcoded)

- Always returns two-time segments (e.g., 1.0–2.5s and 4.0–5.5s).
- Purpose is tvisualize where forgery may have occurred.
- In future implementations, this can be replaced with:
- Anomaly detection across feature frames.
- Signal segmentation algorithms.

6. Results Display in GUI

- Classification Result: Class name with confidence percentage.
- Forged Region Highlights: Plotted as red spans on:
- Waveform (amplitude over time)
- Spectrogram (frequency over time)
- AudiControls: Playback using Pygame mixer:
  1. Play
  2. Pause
  3. Stop

4.2 Data Design

4.2.1 Data Structure

The application uses several key data structures in memory during processing:

- **AudiFeatures (LFCC)**
- Type: numpy.ndarray
- Shape: (max_len, n_coeffs) → typically (200, 20)
- Description: This is a 2D array of LFCC features extracted from the audisignal.

- Purpose: Input tmachine learning models for classification.
- **Labels**
- Type: int
- Values:
- 0 = Real
- 1 = Copy-Move Forgery
- 2 = AudiSplicing
- Purpose: Used for supervised learning classification during training.
- **Feature and Label Collections**
- X (Features): numpy.ndarray of shape (N, 200, 20)
- y (Labels): numpy.ndarray of shape (N,)
- Purpose: Stores the dataset used for training and validation of models.
- **Model Outputs**
- Predicted Class: int (0, 1, or 2)
- Confidence Scores: float array of length 3 (Softmax output)
- Returned As: String for GUI display, e.g., "Copy-Move Forgery (Confidence: 92.13%)"
- **Forged Regions**
- Type: List[Tuple[float, float]]
- Format: [(start_time_sec, end_time_sec), ...]
- Purpose: Identifies and displays forged intervals in waveform/spectrogram.

4.2.2 Database description

There is nformal database used, However, we have recorded our voice notes for testing and training:

1. File System (Directory-Based Storage)

Dataset Location:

C:\\Users\\HP\\PycharmProjects\\STAR_final\\Data\\Train

Organized intsubfolders:

□ /real

□ /copy_move

□ /audio_splicing

Each subfolder contains .wav audifiles corresponding ttheir class label.

2. Model and Scaler Storage

Model Files:

Format: .h5 (Keras HDF5 model files)

Example: audio_forgery_best_model.h5, best_model_dense.h5

Scaler File:

Format: .pkl (Pickled StandardScaler using joblib)

File: lfcc_scaler.pkl

Purpose: Reuse trained models and normalization parameters for prediction in the GUI.

3. Logs

File: execution_log.txt

Contents: Textual log of events like audiloading, prediction, and errors.
Purpose: Useful for debugging and tracking application behavior.

4.2.3 UML Class Diagram

Fig 4.2.3.1 UML Class Diagram

1. AudiProcessor

It Handles feature extraction from audifiles.
Methods:
+ extract_lfcc_features(audio_path): Uses Short-Time Fourier Transform (STFT), log-spectrum, and Discrete Cosine Transform (DCT) tgenerate LFCC (Linear Frequency Cepstral Coefficients). Pads or truncates tfixed time frames and returns the average LFCC vector.
+ prepare_dataset(): (not shown in this script but referenced from earlier code) prepares training data by extracting LFCCs from labeled audifiles.
Importance:
ModelBuilder for training.
predict_audio() for feature input tthe ML model.

2. Model Builder

It Provides different deep learning architectures tprocess LFCC features.
Methods:
+ build_dense_model()
+ build_cnn_model()
+ build_lstm_model()
Importance:
ModelTrainer tobtain the desired model architecture.

3. Model Trainer

It Trains and evaluates machine learning models.
Attributes:
- X_train, y_train: training data
- X_val, y_val: validation data
- scaler: used tnormalize input LFCC vectors

Methods:
- + train_models(): trains models using compiled architectures and tracks best accuracy.
- + evaluate_models(): evaluates model performance on validation data.

Importance:
- Models from Model Builder
- Features from AudiProcessor

4. Prediction Engine

Method:
- + predict_audio(audio_path, model_path, scaler_path):
- Loads pre-trained model (.h5)
- Loads scaler (.pkl)
- Calls extract_lfcc_features for input feature extraction.
- Calls detect_forged_regions() tidentify likely tampered regions.

- Returns classification (e.g., "AudiSplicing") and forged region time intervals.

Importance:

- GUI (AudiForgery App) tclassify uploaded audifiles.

5. detect_forged_regions()

It currently a placeholder that returns hardcoded forged time regions (e.g., (1.0s–2.5s)).

6. log_message()

- Logs status messages texecution_log.txt and prints them tconsole.
- Used across the system ttrack execution and debug.

7. AudiForgery App

It acts as Main Graphical User Interface (GUI) for the user.

Key Attributes:

- GUI layout attributes: waveform_frame, spectrogram_frame, plot_option, result_label, etc.
- Audiplayback: audio_data, sr, and pygame integration
- Visualization: uses matplotlib for waveform/spectrogram plots

Key Methods:

- + upload_audio(): Handles audifile selection and triggers prediction.
- + update_plot(): Displays either waveform or spectrogram with forged regions highlighted.
- + play_audio(), + pause_audio(), + stop_audio(): Controls audiplayback using pygame.

Importance:

- predict_audio() for inference
- extract_lfcc_features() and detect_forged_regions() indirectly
- matplotlib, librosa, and tkinter for visualization and interface

8. External Libraries

These provide core functionality across the app:

- tkinter: GUI
- librosa: Audiloading and STFT
- matplotlib: Visualization
- tensorflow: ML model inference
- joblib: Loading scaler
- pygame: Audiplayback
- pydub, PIL, NumPy, SciPy: Signal processing, image handling Relationships Summary
- AudiForgery App is the main controller that uses almost every other component.
- predict_audio() is a central function that coordinates prediction, feature extraction, and region detection.
- Helper modules (AudiProcessor, Model Builder, Model Trainer) could be fully class-based for improved modularity, but they are logically grouped here.

Relationships Summary

- AudiForgery App is the main controller that uses almost every other component.
- predict_audio() is a central function that coordinates prediction, feature extraction, and region detection.
- Helper modules (AudiProcessor, Model Builder, Model Trainer) could be fully class-based for improved modularity, but they are logically grouped here.

5. Project Implantation

5.1 Overview of Project Module

5.1.2. Training Module – STAR10.py

This script handles the entire machine learning pipeline, from data preprocessing tmodel training and saving.

1. Feature Extraction:

Function: extract_lfcc_features(audio_path)

Description: Converts raw audiinta 20-dimensional feature vector using Linear Frequency Cepstral Coefficients (LFCC).

Key Steps:

- Loads audiusing librosa.
- Applies STFT (Short-Time Fourier Transform) tget power spectrogram.
- Applies logarithmic scaling.
- Applies DCT (Discrete Cosine Transform).
- Returns a mean-compressed LFCC feature vector.

2. Dataset Preparation:

Function: prepare_dataset(dataset_path)

Description: Loads .wav files from the directory structure, extracts features, and associates them with class labels.

Label Mapping:

- real: 0
- copy_move: 1
- audio_splicing: 2

Outputs:

- Feature matrix X (shape: [n_samples, n_features])
- Label array y (shape: [n_samples])

3. Feature Normalization:

Uses StandardScaler from scikit-learn.

Saves the fitted scaler using joblib.dump().

4. Model Building and Training:

Function: build_model(input_shape)

Architecture:

- Input Layer → Dense(128, ReLU) → Dropout(0.3)
- Hidden Layer → Dense(64, ReLU) → Dropout(0.3)
- Output Layer → Dense(3, Softmax)
- Trained with sparse_categorical_crossentropy loss and Adam optimizer.
- Training epochs: 20, Batch size: 16.
- Trained model is saved as audio_forgery_model.h5.

Prediction Utility:

- Function: predict_audio(audio_path)
- Loads model and scaler.
- Extracts and scales LFCC features from new audio.

- Returns the predicted class with confidence.

5.2 Tools & Technology Used

1. Python 3.8

Used as the primary programming language for the entire project.
Offers excellent support for libraries like TensorFlow, Librosa, and Tkinter.
Facilitates rapid development of machine learning pipelines and desktop applications.

2. TensorFlow / Keras

Provides tools tbuild, train, and deploy deep learning models.
Keras (part of TensorFlow) is used tdesign a multi-layer neural network for classifying audiintreal, copy-move forgery, or spliced.
Supports GPU acceleration and model serialization (.h5 format) for deployment.

3. Librosa

A Python library dedicated taudiand music signal processing.
Used tload .wav files, perform Short-Time Fourier Transform (STFT), and visualize waveforms and spectrograms.
Essential for feature extraction, especially in generating power spectrograms used for LFCC calculation.

4. Scikit-learn

Employed mainly for feature scaling using StandardScaler.
Normalizes extracted features timprove the performance of the machine learning model.
Ensures that each feature contributes equally tthe learning process.

SciPy (fftpack)

Used tcompute the Discrete Cosine Transform (DCT), a crucial step in calculating LFCC (Linear Frequency Cepstral Coefficients).
DCT transforms frequency-domain data inta compact, informative feature vector.
Helps in reducing dimensionality and retaining meaningful patterns for classification.

6. Tkinter

Python's standard GUI toolkit used tdesign the application's user interface.
Provides widgets like buttons, labels, and frames tcreate a desktop app.
Integrates well with plotting libraries and enables interactive controls like upload, play, pause, and display areas.

7. Matplotlib

Used for visualizing audidata such as waveforms and spectrograms within the GUI.
Supports interactive plotting using FigureCanvasTkAgg in Tkinter.
Enables visual indication of forged audiregions through overlays (axvspan).

8. Pydub

Simplifies audifile conversion and manipulation.
Converts or processes audifiles before classification.
Depends on FFmpeg as its backend tdecode and encode a wide range of audiformats.

9. FFmpeg

A powerful multimedia framework used by Pydub tdecode audiformats.
Required for reading various audifile types other than native .wav.

Configured in the code using its executable path for compatibility.

10. Pillow (PIL)

A Python Imaging Library used for loading and displaying images in the GUI.

Used tmanage the background image and any static image components in the application.

Supports resizing and rendering of image elements.

11. Pygame

Used for audiplayback functionality within the GUI.

Supports loading, playing, pausing, and stopping .wav audifiles.

Provides a smooth user experience for real-time audiexamination.

12. Joblib

Used for saving and loading machine learning models and preprocessors efficiently.

Saves the StandardScaler and trained neural network for reuse without retraining.

Faster than pickle for large numpy arrays and model files.

13. OS and NumPy

os: Used for file and directory operations, such as iterating through dataset folders.

numpy: Used for numerical computations, array manipulation, and input/output with libraries like TensorFlow, Librosa, and SciPy.

Together they form the low-level infrastructure of the data pipeline and model input/output handling.

5.3 Algorithm Details

5.3.1. AudiFeature Extraction Algorithm – LFCC

This algorithm is important tTransform a raw audiwaveform inta compact, fixed-size numerical representation Linear Frequency Cepstral Coefficients (LFCC) suitable for classification.

1. AudiLoading:

The input .wav file is loaded using librosa.load() at a fixed sample rate of 16,000 Hz tensure consistency across samples.

2. Short-Time Fourier Transform (STFT):

Performed using librosa.stft() with:

n_fft = 512: Frame size for frequency resolution.

hop_length = 256: Step size between frames for time resolution.

Converts the signal inta 2D matrix (frequency bins × time frames).

Power Spectrogram Calculation: Magnitude of STFT is squared tobtain power spectrogram: $|STFT|^2$.

Logarithmic Scaling: Applies np.log1p() tthe power spectrogram tcompress dynamic range, making subtle frequency differences more detectable.

5. Discrete Cosine Transform (DCT):

Applies DCT-II (orthonormal) using scipy.fftpack.dct() across the frequency axis tdecorrelate features.

Converts log-spectral data intcepstral coefficients (LFCCs).

6. Dimensionality Reduction:

Retains only the top 20 coefficients (most informative) from the DCT output.

7. Temporal Pooling:

Computes the mean across all time frames, resulting in a fixed 20- dimensional feature vector, independent of audilength.

5.3.2. Deep Neural Network for Classification

1. There are multi-class classification with three classes:

- Class 0: Real Data
- Class 1: Copy-Move Forgery
- Class 2: AudiSplicing

2. Input:

- A 20-dimensional LFCC feature vector per audisample.

3. Model Architecture:

- Input Layer:

Dense(128) with ReLU activation
Followed by Dropout(0.3) tprevent overfitting.

- Hidden Layer:

Dense(64) with ReLU activation
Followed by Dropout(0.3) for regularization.

- Output Layer:

Dense(3) with Softmax activation
Outputs probability scores for the three classes.

- Loss Function: Sparse Categorical Crossentropy (used for integer-labeled multiclass tasks).
- Optimizer: Adam with learning rate = 0.001 (efficient gradient-based optimization).
- Metrics: Model evaluated using accuracy.
- Training Setup:

Epochs = 20
Batch Size = 16
Validation Split = 0.2 (20% of training data used tvalidation performance during training).

5. Model Output:

- Final model saved as audio_forgery_model.h5 for use in the GUI.

5.3.3. Classification and Forgery Detection in GUI

1. Prediction Pipeline in Application:

AudiUpload via GUI: User selects a .wav file using a Tkinter file dialog.
Feature Extraction: Same LFCC extraction process as used during training.
Feature Normalization: Uses the pre-trained StandardScaler (lfcc_scaler.pkl) tscale input features tthe same distribution as training data.

2. Model Prediction:

The scaled feature vector is passed tthe trained neural network (audio_forgery_model.h5).
The model returns class probabilities for:
Real Data
Copy-Move Forgery
AudiSplicing

3. Result Interpretation:

The class with the highest probability is selected.
Confidence score is shown in the GUI alongside the predicted class.

4. Forgery Region Identification:
Currently uses a placeholder algorithm (detect_forged_regions) that returns hardcoded intervals indicating suspected manipulation (e.g., [(1.0, 2.5), (4.0,5.5)]).
These regions are:
Visually highlighted on both the waveform and spectrogram using matplotlib's axvspan() function (adds shaded red spans over the suspected forged regions).
Displayed in the GUI along with textual interpretation ("Meaning" and "Conclusion").

6. Results and Discussion

6.1 Experimental Setup

6.1.1 Data set

Data Source: The dataset used is a custom audidataset manually organized intthree folders:
real/ – genuine, unaltered audisamples.
copy_move/ – audisamples tampered by duplicating and repositioning segments.
audio_splicing/ – audisamples manipulated by inserting or removing audifrom other sources.
Data Format: All files are in .wav format and sampled at 16 kHz for uniformity in processing.
Data Size: Although the dataset size is not explicitly mentioned in the code, the program is designed thandle dozens thundreds of audifiles per class.

6.1.2 Performance Parameters

1 Accuracy:
Measures the proportion of correctly classified audifiles across the three classes.
2 Confidence Score:
Displayed in the GUI during inference. Reflects the model's confidence in its prediction.
3 Validation Split:
20% of the training data is automatically reserved during training for validation.
4 Loss Value:
Sparse categorical cross-entropy is used as the loss metric, minimized during model training.
5 Execution Time (In GUI):
Real-time classification (including loading, feature extraction, scaling, and prediction) completes within 2–3 seconds per file.

6.1.3 Efficiency Issues

- **Feature Extraction Time:**

Feature extraction (LFCC computation) can be time-consuming for longer audisamples due tSTFT and DCT processing
Memory Usage:
Loading and processing multiple audifiles at once could be memory-intensive depending on audilength and batch size.

- **Model Overfitting Risk:**

As Dropout is used (30% in twlayers), overfitting is partially addressed. However, a larger dataset or data augmentation would further improve generalization.

- **Forgery Detection Granularity:**

Forged region identification is currently hardcoded as a placeholder; implementing dynamic localization would increase complexity and processing cost.

6.2 Software Testing

6.2.1 Test Cases and Test Results

Test case ID	Test description	Expected Outcome	Actual Outcome	Result
TC01	Upload a real .wav file	Classified as "Real Data"	Classified as "Real Data"	Pass
TC02	Upload a spliced .wav file	Classified as "Audio-Splicing"	Classified as "AudiSplicing"	Pass
TC03	Upload a copy-move forged.wav file	Classified as "Copy-Move Forgery"	Classified as "Copy-Move Forgery"	Pass
TC04	Upload an empty .wav file	Error message for unreadable or empty file	Error message shown	Pass
TC05	Click play/pause/stop for uploaded audio	Audiplayback controlled correctly	Playback functions working as expected	Pass

6.3 RESULTS

6.3.1 Result Analysis and Discussion

- The system was able tclassify audisamples intthree distinct categories with high reliability based on LFCC features.
- Training accuracy reached acceptable levels after 20 epochs, with validation showing nmajor signs of overfitting due tDropout layers.
- The confidence scores displayed during predictions were typically above 80%, indicating strong model certainty on test data.
- While the forged region detection was simulated, it effectively enhanced the interpretability of the results in the GUI by visually highlighting suspicious areas.
- LFCC proves tbe a viable feature for audiforgery detection, offering compact and informative representation.
- GUI integration ensures the model is usable by non-technical users, providing fast and interactive feedback.

6.3.2 Graphical Interface

- AudiUpload & Classification: One-click upload and instant classification output.
- Real-Time Visualization: Displays both waveform and spectrogram, with suspected forged regions highlighted in red.
- Playback Control: Supports Play, Pause, and Stop using pygame.mixer.
- Interpretability: Shows textual feedback including:

Classification result
Time intervals of suspected forgery
Explanation (“Meaning”) and conclusion of detection User-Friendly Design: Responsive layout,

background image, bold typography, and labeled sections make it suitable for end-users and dempurposes.

7. Conclusion and Future Work

7.1 Conclusion

The implemented audiforgery detection system successfully implements a machine learning-based approach tidentify and classify forged audicontent. The system is capable of distinguishing between Real Data, Copy-Move Forgery, and AudiSplicing, leveraging Linear Frequency Cepstral Coefficients (LFCC) as the core audifeature and a deep neural network model trained using TensorFlow/Keras.

Through extensive preprocessing such as STFT, log power spectrogram conversion, and DCT-based LFCC extraction the system converts raw audiintcompact, discriminative features suitable for classification. The model architecture, composed of twhidden layers with dropout regularization, has demonstrated reliable performance and generalization over the dataset.

In addition, the integration of this model inta Tkinter-based graphical user interface (GUI) has made the tool accessible and user-friendly. Users can upload .wav audifiles, visualize waveforms or spectrograms, playback audio, and view classified results with confidence scores and interpreted forged regions.

Overall, our designed system presents a complete and functional solution for auditampering detection. It bridges the gap between machine learning techniques and real-time user interaction for digital audiforensics.

7.2 Future Work

While the current system meets its objectives effectively, several enhancements can be considered timprove accuracy, robustness, and real-world applicability:

1. Dynamic Forgery Region Detection:

The current implementation uses hardcoded forged intervals. Future versions should implement automatic detection of forged segments using signal anomalies or change-point detection algorithms tlocalize forgery more precisely.

2. Advanced Feature Sets:

Future models can integrate additional features such as MFCCs, chroma, zero-crossing rate, or deep audiembeddings (e.g., from VGGish or wav2vec) tcapture richer audicharacteristics.

7.3 Application

This audiforgery detection system has several practical and real-world applications, particularly in the domain of audiforensics and cybersecurity.

1. **Digital Forensics:** For law enforcement agencies tverify the authenticity of audi evidence, especially in court admissible materials like voice recordings, confessions, or surveillance.
2. **Media & Journalism:** Helps identify tampered or doctored interviews, leaked calls, or manipulated audinews clips, ensuring the integrity of public information.
3. **Legal and Compliance:** Can be used by legal firms tvalidate recorded contracts, witness statements, or auditestimonies.
4. **Cybercrime Investigation:** Detects and flags suspicious audiin scams, impersonation frauds, or deepfake-based blackmail threats.

5. **Academic and Research Institutions:** Useful in research on digital signal processing, machine learning, and ethics in media; it can alsserve as a project module for student training.
6. **Call Center and Telecom Monitoring:** Automatically validate recordings for compliance auditing and detect any unauthorized alterations.
7. **Broadcast Monitoring Systems:** Ensure the authenticity of live or recorded content on radiand TV platforms.

REFERENCES

1. J. Kirovski and H. S. Malvar, "Spread-Spectrum Watermarking of AudiSignals," *IEEE Transactions on Signal Processing*.
2. C. Kraetzer, A. Oermann, J. Dittmann, and A. Lang, "Digital AudiForensics: A First Practical Evaluation on Microphone and Environment Classification," in *Proceedings of the Multimedia and Security Workshop (MM&Sec)*, 2007.
3. R. Yang, Z. Qu, and J. Huang, "Detecting Digital AudiForgeries by Checking Frame Offsets," in *Proceedings of the 10th ACM Workshop on Multimedia and Security*.
4. X. Pan, X. Zhang, and S. Lyu, "Detecting Splicing in Digital Audios Using Local Noise Level Estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.
5. J. Chen, S. Xiang, W. Liu, and H. Huang, "Exposing Digital AudiForgeries in Time Domain by Using Singularity Analysis with Wavelets," in *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*.
6. H. Zhao, Y. Chen, R. Wang, and H. Malik, "AudiSource Authentication and Splicing Detection Using Acoustic Environmental Signature," *Multimedia Tools and Applications*.
7. Q. Yan, R. Yang, and J. Huang, "Copy-Move Detection of AudiRecording with Pitch Similarity," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
8. M. Imran, Z. Ali, T. B. Sheikh, and S. Akram, "Blind Detection of Copy-Move Forgery in Digital AudiForensics," *IEEE Access*.
9. L. Cuccovillo, S. Mann, M. Tagliasacchi, and A. P. Aichroth, "AudiTampering Detection via Microphone Classification," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.
10. O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, Oct. 2014.