AI model documentation

Jarpula Bhanu Prasad
AI21BTECH11015

Kola Akshitha
AI21BTECH11017

1 Introduction

The AI model implemented for liveliness detection is designed to distinguish between genuine (real) and spoofed (fake) audio samples. This model leverages audio signal processing techniques and deep learning architectures to effectively analyze the temporal and spectral characteristics of the audio, ensuring robust detection of spoofing attacks such as voice synthesis or playback. By extracting Mel-frequency cepstral coefficients (MFCCs) from raw audio data, the model learns to classify the audio as either live or non-live based on distinctive voice characteristics.

2 Libraries and Tools Used

The following libraries and tools were utilized for implementing the AI model for liveliness detection:

- **Python:** The programming language used for development.
- **TensorFlow:** For building and training the deep neural network model.
- **Scikit-learn:** For preprocessing tasks such as label encoding and evaluation metrics.
- **Librosa:** For extracting Mel-frequency cepstral coefficients (MFCCs) from audio files.
- **Imbalanced-learn:** For resampling the dataset to address class imbalance using the `RandomOverSampler`.
- **Matplotlib:** For visualizing training performance and results.

3 Dataset Information

Source: The dataset used in this project is sourced from Kaggle, an open dataset platform.
Description:

- The dataset consists of labeled audio files in `.wav` format.

- It contains two classes:
  - **Real:** Genuine audio samples where the speaker is physically present.
  - **Fake:** Spoofed audio samples, including synthetic voices or pre-recorded audio.
- Each `.wav` file corresponds to one of these classes, making the dataset suitable for supervised learning tasks.

4 Data Preprocessing and Handling

4.1 Label Encoding

Audio samples are categorized into two labels:

- **Real:** Indicates genuine audio samples where the speaker is physically present.
- **Fake:** Represents spoofed audio samples, including synthetic voices or pre-recorded audio.

The labels are encoded using LabelEncoder from scikit-learn to convert them into numerical values suitable for model training.

4.2 Resampling and Splitting

To address any class imbalance, the dataset is resampled using the RandomOverSampler technique. This ensures that both classes (Real and Fake) are equally represented during training, preventing the model from learning biased patterns.

The data is then split into training and test sets (80% training, 20% testing) for model evaluation.

5 Model Architecture

5.1 Feature Extraction

The model begins by extracting MFCC features from the input audio. MFCCs are widely used in speech recognition and speaker verification tasks because they capture the power spectrum of the audio signal, which is highly correlated with human speech patterns.

1. **MFCC Calculation:** The audio signal is passed through a series of transformations to compute a set of MFCCs, which are the most prominent features representing the short-term power spectrum of sound. The number of coefficients used is typically set to 40, reflecting the most critical spectral components of speech.
2. **Feature Scaling:** The MFCC features are scaled (mean of the features over time) to ensure they are normalized and appropriate for input to the neural network.

5.2 Neural Network Design

The model uses a fully connected deep neural network (DNN) architecture built with TensorFlow. The layers and structure are designed to process the extracted features and classify the input as either real or fake.

1. **Input Layer:** Accepts a fixed-length feature vector corresponding to the scaled MFCC features of the audio sample.
2. **Hidden Layers:** The neural network consists of three dense (fully connected) layers:
   - **First Hidden Layer:** 128 neurons, activated by ReLU. This layer learns complex representations of the MFCC features.
   - **Second Hidden Layer:** 256 neurons, also activated by ReLU. It enables deeper feature extraction and patterns learning.
   - **Third Hidden Layer:** 128 neurons, activated by ReLU, further refining the model's ability to detect liveliness.
3. **Regularization:** Dropout layers (with a dropout rate of 0.5) are placed after each hidden layer to prevent overfitting and to ensure that the model generalizes well to unseen data.
4. **Output Layer:** The output layer uses softmax activation to provide a probability distribution over the two classes, that is real and fake.

5.3 Model Loss and Optimizer

1. **Loss Function:** The model uses categorical cross-entropy loss, a standard loss function for multi-class classification tasks, which computes the difference between the predicted probabilities and the true class labels.
2. **Optimizer:** Adam optimizer is used to minimize the loss function. It adapts the learning rate for each parameter and is known for its efficiency in training deep networks.

6 Model Training and Evaluation

6.1 Training

The model is trained for 200 epochs with a batch size of 2. During training, the following hyperparameters are used:

- **Batch Size:** 2
- **Epochs:** 200

Validation: The model is validated on the test data after every epoch to monitor performance and avoid overfitting.

6.2 Evaluation Metrics

The primary evaluation metric for this model is accuracy, which measures the proportion of correct predictions made by the model on the test set.

6.3 Performance

1. **Loss:** The categorical cross-entropy loss indicates how well the model's predictions align with the actual labels. A lower loss indicates better performance.

2. **Accuracy:** The percentage of correct classifications (live or non-live) made by the model on the test dataset.

7 Model Usage

Once the model is trained, it can be used for liveliness detection in new audio files. The process involves the following steps:

- **Audio Preprocessing:** New audio samples are preprocessed to extract MFCC features.
- **Model Prediction:** The features are passed to the trained neural network for classification, where the model outputs the probability of the audio being live.
- **Decision:** Based on the output probability, the model classifies the audio as either real or spoofed.