



SPEECH RECOGNITION REPORT



Name and ID

Basmala Akram hegazy

2103150

Importing the libraries

1. **import os:**

- Provides a way to interact with the operating system. It's often used for file and directory manipulation.

2. **import glob:**

- Helps in retrieving files/pathnames matching a specified pattern. It's useful for working with multiple files.

3. **import numpy as np:**

- numpy is a fundamental package for scientific computing with Python. It provides support for arrays, matrices, and many mathematical functions.

4. **import pandas as pd:**

- pandas is used for data manipulation and analysis. It provides data structures like Series and DataFrame for handling structured data.

5. **import seaborn as sns:**

- seaborn is a statistical data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

6. **import matplotlib.pyplot as plt:**

- matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. pyplot is a module in matplotlib used for 2D plotting.

7. **import librosa.display:**

- Part of the librosa library, which is used for audio and music processing. The display module provides functions for visualizing audio data, such as waveforms and spectrograms.

Audio Plot

8. **from scipy.io import wavfile as wav:**

- scipy.io.wavfile provides functions to read and write WAV files. It's used to handle audio data in WAV format.

9. **import IPython.display as ipd:**

- IPython.display provides utilities to display various media types (images, audio, etc.) directly within Jupyter notebooks. The alias `ipd` is commonly used for displaying audio

Machine Learning Libraries

10. **from sklearn.preprocessing import LabelEncoder:**
 - LabelEncoder is used to convert categorical labels into numeric form, which is often required for machine learning models.
11. **from tensorflow.keras.utils import to_categorical:**
 - to_categorical converts a class vector (integers) to binary class matrix (one-hot encoding), which is used in classification problems.
12. **from sklearn.model_selection import train_test_split:**
 - train_test_split is used to split the dataset into training and testing sets.

Building Neural Networks

13. **from tensorflow.keras.models import Sequential, Model:**
 - Sequential is a linear stack of layers, suitable for simple models.
 - Model is used to create more complex models, including those with multiple inputs and outputs.
14. **from tensorflow.keras.callbacks import Callback, EarlyStopping:**
 - Callback is a base class to create custom callbacks during training.
 - EarlyStopping is a built-in callback that stops training when a monitored metric has stopped improving.
15. **from sklearn.metrics import confusion_matrix, classification_report:**
 - confusion_matrix and classification_report are tools to evaluate the performance of a classification model.
16. **from tensorflow.keras.layers import Conv2D, Activation, Flatten, Dense, GlobalAveragePooling2D, Dropout:**
 - These are various neural network layers provided by Keras:
 - Conv2D: A 2D convolution layer used for processing images (or spectrograms in this case).
 - Activation: Applies an activation function to the output of a layer.
 - Flatten: Flattens the input, typically used to convert 2D matrix data into a 1D vector.
 - Dense: A fully connected neural network layer.
 - GlobalAveragePooling2D: Reduces each feature map to a single number by taking the average.
 - Dropout: Regularization technique to prevent overfitting by randomly dropping neurons during training.

2- Loading Dataset

Th variables define the paths to important directories in the project, specifically where the cats_dogs dataset is stored. The paths are defined both by concatenation of strings and by directly assigning the full path.

3- Preparing the data

1-Loading Audio Files:

- `data_train` and `data_test` are lists that contain the file paths of all .wav audio files in the training and testing directories, respectively. This is done using the `glob` function, which searches for files matching a specific pattern (`**/*.wav` looks for .wav files in all subdirectories).

2- Extracting Labels:

- `labels` is a list of labels for the training data. The label is extracted by splitting the file path twice: first to get the parent directory of the file, and second to get the directory name (either 'cat' or 'dog').

3- Creating a DataFrame:

- `file_path`: A pandas Series object that stores the paths of the training audio files.
- `labels`: A pandas Series object that stores the labels ('cat' or 'dog') corresponding to each file.
- `data`: A DataFrame created by combining `file_path` and `labels` as columns. This DataFrame is then shuffled (`sample(frac=1)`) to randomize the order of the rows, and the index is reset.

4- Display Sample:

- `data.head()`: Displays the first few rows of the DataFrame to give an overview of the file paths and their corresponding labels.

5- Extracting Labels:

- labels: A list is created by mapping each file path in data_train to its label (either 'cat' or 'dog'). This is done by splitting the file path twice to access the parent directory name, which represents the label.

6- Creating a DataFrame:

- file_path: A pandas Series that stores the file paths of the training audio files as strings.
- labels: A pandas Series that stores the corresponding labels ('cat' or 'dog').

7- Combining Data:

- data: A DataFrame created by combining the file_path and labels Series. This DataFrame pairs each file path with its label.

8- Shuffling:

- The DataFrame data is shuffled (sample(frac=1)) to randomize the order of the rows, and the index is reset.

9- Display Sample:

- data.head(): Displays the first few rows of the DataFrame, showing the file paths and their corresponding labels.

4-Data Visualization

1- Label Count Plot:

- A bar plot shows the count of 'Dog' and 'Cat' labels in the dataset.

5-Data Augmentation

2- Waveform Plots:

- The code loads and plots the waveform of the first 4 audio files in the dataset.

3- Spectrogram and Waveform Plots:

- For each of the first 4 audio files, the code plots both the waveform and its spectrogram using Short-Time Fourier Transform (STFT).

6-Data Preprocessing

1- Extract_features Function:

- Loads an audio file and extracts its Mel-frequency cepstral coefficients (MFCCs), which are a set of features representing the audio's frequency content.
- The extracted MFCCs are averaged (mean) across time to create a single feature vector.

2- Featuresdf DataFrame:

- After extracting features from multiple audio files, the features and their corresponding class labels (e.g., 'cat' or 'dog') are stored in a DataFrame called featuresdf.

7-Data Preprocessing

1- Extract_features Function:

- Loads an audio file and extracts its Mel-frequency cepstral coefficients (MFCCs), which are a set of features representing the audio's frequency content.
- The extracted MFCCs are averaged (mean) across time to create a single feature vector.

2- Featuresdf DataFrame:

- After extracting features from multiple audio files, the features and their corresponding class labels (e.g., 'cat' or 'dog') are stored in a DataFrame called featuresdf.

8-Data Extraction

1- Convert Features and Labels to Arrays:

- X: Converts the list of features (from the DataFrame) into a NumPy array.
- y: Converts the list of class labels into a NumPy array.

2-Train-Test Split:

- Splits the data into training and testing sets.
- `x_train, y_train`: Data and labels for training.
- `x_test, y_test`: Data and labels for testing.
- `test_size=0.2`: 20% of the data is used for testing.
- `random_state=42`: Ensures reproducibility by fixing the random seed.

9-Model Building

1- Model Creation:

- **Sequential Model**: A simple feedforward neural network.
- **Layers**:
 - `Dense(64)`: First hidden layer with 64 neurons and ReLU activation.
 - `Dropout(0.2)`: Randomly drops 20% of neurons during training to prevent overfitting.
 - `Dense(46)`: Second hidden layer with 46 neurons and ReLU activation.
 - `Dropout(0.2)`: Another dropout layer.
 - `Dense(32)`: Third hidden layer with 32 neurons and ReLU activation.
 - `Dropout(0.2)`: Another dropout layer.
 - `Dense(2)`: Output layer with 2 neurons (for binary classification) and softmax activation.

2- Model Compilation:

- `loss='categorical_crossentropy'`: Loss function for multi-class classification.
- `optimizer='adam'`: Adam optimizer.
- `metrics=['accuracy']`: Track accuracy during training.

3- Training the Model:

- **Early Stopping:** Stops training if validation loss doesn't improve after 3 epochs.
- **Fit the Model:** Train the model on training data, validating on test data, with early stopping.
- **Plot Accuracy:** Visualizes the training and validation accuracy over epochs.

10-Model Evaluation

Model Evaluation:

- **Training Accuracy:**
 - `model.evaluate(x_train, y_train, verbose=0)`: Evaluates the model on the training data.
 - `score[1]`: Retrieves the accuracy from the evaluation.
 - `print("Training Accuracy...")`: Prints the training accuracy as a percentage.
- **Testing Accuracy:**
 - `model.evaluate(x_test, y_test, verbose=0)`: Evaluates the model on the test data.
 - `score[1]`: Retrieves the accuracy from the evaluation.
 - `print("Testing Accuracy...")`: Prints the testing accuracy as a percentage.

10-Model Tesing

1- Prediction Function:

- **prediction_(path_sound):** This function predicts the class (cat or dog) for an input audio file.
 - **Extract Features:** `data_sound = extract_features(path_sound)` extracts features from the audio file.
 - **Prepare Input:** The features are reshaped into the required format (`X.reshape(1,40)`).
 - **Model Prediction:** `pred_ = model.predict(X)` predicts the probabilities for each class.
 - **Class Prediction:** `pred_ = np.argmax(pred_,axis=1)` finds the class with the highest probability, and `le.inverse_transform(pred_)` converts it back to the original label.
 - **Print Result:** The predicted class is printed.

2- Testing the Function:

- **path_sound:** Specifies the path to the test audio file.
- **prediction_(path_sound):** Calls the prediction function to predict the class of the given audio file.
- **ipd.Audio(path_sound):** Plays the audio file so you can listen to it.