Project Report LE1

Conversational AI: Speech Processing and Synthesis

Submitted By:

Chhavi Gupta

Roll no: 102103322

Class: 4CO12

Submitted To:

Dr. BV Raghav

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

Summary of Research Paper

The Speech Commands dataset contains 105,829 utterances of 35 words from 2,618 speakers, using hashed speaker IDs for privacy. It features a variety of background noise samples and supports automated and manual quality control methods. Evaluation metrics include Top-One Error and Streaming Accuracy, assessing model performance in keyword recognition and continuous audio streams. Version 2 improves accuracy over Version 1 and provides data for model optimization, noise tolerance, and application development. The dataset is accessible for reproducible research and diverse model benchmarking.

Dataset Analysis

The Speech Commands Dataset is a large collection of audio recordings designed for limited-vocabulary speech recognition tasks, particularly useful in IoT applications. It comprises over 65,000 one-second clips of 20 frequently used words like "Yes," "No," and directions such as "Up" and "Down." The dataset also includes random distractor words, silence, and background noise to ensure model robustness in recognizing speech commands amid non-relevant speech and noise. Recorded by more than 1,800 speakers, the dataset features diverse accents and speaking styles, enhancing its generalization to real-world conditions. The data is collected through a web-based interface, with quality controlled via automated filters and manual reviews. This variety in speech characteristics makes the dataset valuable for training, testing, and benchmarking models for keyword recognition, facilitating advancements in speech recognition systems.

Code Summary

1. **Import Libraries**: Essential libraries such as **pandas, numpy, tensorflow, librosa**, and **sklearn** are imported for data processing, model building, and evaluation.
2. **Dataset Management**:
    - **download_speech_commands()**: Function to download and prepare the Speech Commands dataset, including cleanup.
    - **use_existing_speech_commands()**: Adapted function to work with an existing dataset on a specified path.
3. **Visualization**:
    - **plot_spectrogram()**: Function to visualize audio spectrograms.
    - **librosa and matplotlib**: Used to visualize audio signals and their spectrograms.
4. **Data Preprocessing**:

    - **preprocess_dataset()**: Function to preprocess audio files, extract features (MFCCs or Mel spectrograms), and save data in JSON format.
    - **load_data():** Function to load preprocessed data from JSON.
    - **create_train_test():** Splits the dataset into training, validation, and test sets.
5. **Model Building and Training**:
    - **build_crnn_model()**: Constructs a Convolutional Recurrent Neural Network (CRNN) model for speech command classification.
    - **fit_model():** Trains the CRNN model with early stopping to prevent overfitting.
6. **Evaluation**:
    - **model_performance()**: Evaluates and reports model performance metrics, including confusion matrix and classification report.
    - **save_model()**: Saves the trained model.
7. **Testing and Prediction**:
    - Tests the model on a sample audio file and displays the prediction with confidence scores using **librosa** and **matplotlib.**
8. **Output Visualization**:
    - **Confusion Matrix**: Displays a heatmap of the confusion matrix.
    - **Prediction Confidence**: Shows the predicted probabilities for each command.

Choice of Model

CRNNs were chosen because they effectively combine convolutional layers for extracting spatial features from audio spectrograms or MFCCs with recurrent layers for learning temporal dependencies. This combination excels in recognizing speech commands, capturing both the detailed frequency patterns and the time-varying nature of spoken language. Their proven success in similar tasks and their ability to handle sequential data make CRNNs a strong choice for this problem.

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 42, 11, 32)	320
max_pooling2d_6 (MaxPooling2D)	(None, 21, 5, 32)	0
conv2d_7 (Conv2D)	(None, 19, 3, 64)	18,496
max_pooling2d_7 (MaxPooling2D)	(None, 9, 1, 64)	0
reshape (Reshape)	(None, 9, 64)	0
lstm (LSTM)	(None, 9, 128)	98,816
lstm_1 (LSTM)	(None, 128)	131,584
dense_6 (Dense)	(None, 128)	16,512
dropout_2 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 35)	4,515

Total params: 270,243 (1.03 MB)
Trainable params: 270,243 (1.03 MB)
Non-trainable params: 0 (0.00 B)

Model Performance

```
597/597 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step
CONFUSION MATRIX ------------------
[[632   2   1 ...   0   0   1]
 [  9 608   1 ...   1   3   0]
 [  0   1 603 ...   9   0   1]
 ...
 [  0   0   1 ... 612   0   2]
 [  0   0   0 ...   0 711   1]
 [  0   0   0 ...   1   2 299]]

TRAIN METRICS ---------------------
Loss: 0.17094513773918152
Accuracy: 95.0%

TEST METRICS ----------------------
Loss: 0.43271276354789734
Accuracy: 89.0%
```