12 SEPTEMBER 2024

PROBLEM REPORT

UCS749: CONVERSATIONAL AI: SPEECH PROCESSING AND SYNTHESIS

Prepared by: Kartik Koul
4CO26, Roll Number 102103819

Submitted to: Dr B.V. Raghav
Assistant Professor, CSED

02

ABOUT THE PAPER

The paper describes the development process of a unique dataset designed to further the training and evaluation of keyword spotting models. It recognises and states the requirements for such locally stored models that detect the triggers for the virtual assistants available on modern devices. The paper succinctly details the processes of data collection, cleaning, and evaluation.

ABOUT THE AUTHOR

Pete Warden is an influential AI researcher and engineer known for his contributions to TensorFlow and TensorFlow Lite. He focuses on enabling machine learning on low-power devices, such as smartphones and IoT hardware, and is recognized for creating the SpeechCommands dataset for voice recognition tasks.

03

ABOUT THE DATASET

Following is a short note on the dataset itself:

The SpeechCommands dataset is a collection of audio recordings designed for training and evaluating machine learning models in simple voice recognition tasks. It consists of thousands of 1-second clips of 35 spoken words, such as "yes," "no," "stop," and digits. Created by Pete Warden, it is commonly used for keyword spotting and real-time voice control in low-power devices like smartphones and embedded systems. The dataset aids in developing models for recognizing commands in limited-resource environments.

PROBLEM STATEMENT

This evaluation involves a comprehensive study of the SpeechCommands dataset, starting with summarizing Pete Warden's paper that introduces the dataset. The dataset must then be downloaded and statistically analyzed to provide valuable insights. A classifier is to be trained to distinguish between the voice commands in the dataset, with performance evaluation against standard benchmarks. Additionally, 30 samples of each command must be recorded in one's voice to create a new dataset. The classifier will then be fine-tuned to adapt to the user's voice, with its performance being reported.

04

A BRIEF ANALYSIS OF THE DATASET

1. **Class Distribution**: The dataset contains an uneven number of samples across different classes, which may require balancing techniques. For example, common words like "yes" and "no" tend to have more examples than less frequent words.
2. **Diverse Speakers**: The dataset includes recordings from a diverse group of speakers, which ensures robustness in recognizing voice commands across various accents, pitches, and speaking speeds. This is an important factor for generalizing the classifier to different users.
3. **Audio Characteristics**: Each audio clip is sampled at 16 kHz and is trimmed or padded to 1 second in length.
4. **Background Noise**: The dataset includes specific "background noise" files, which help models become resilient to real-world noise interference.
5. **Speaker IDs**: Each recording includes metadata about the speaker (user id), which allows for speaker-dependent or independent modeling.

05

STATISTICALLY ANALYSING THE DATASET

Statistical analysis of the dataset has been performed by extracting Mel Frequency Cepstral Coefficients (MFCCs) and spectrograms from audio samples. MFCCs are widely used features in speech processing as they capture the power spectrum of the sound, while spectrograms provide a time-frequency representation of the audio. The code (provided in the GitHub repository) processes 100 audio samples, computing both the mean and standard deviation of the MFCCs for each sample, and then aggregates the results to calculate the average mean and standard deviation across the dataset. This statistical approach helps understand the distribution of audio features in the dataset, offering insights into the variability of speech patterns. Additionally, visualizations of both MFCCs and spectrograms are provided to illustrate the feature extraction process. These plots offer a qualitative view of the speech signal's structure, helping to confirm the extraction's effectiveness.

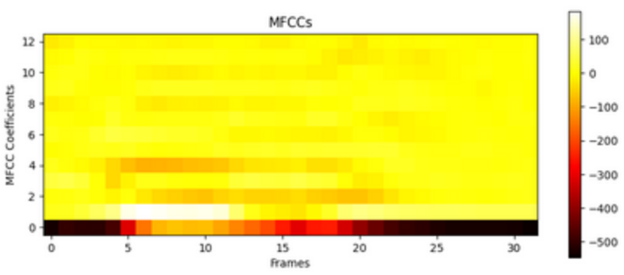

Figure: MFCC Plot of the dataset

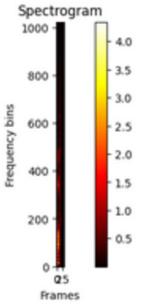

Figure: Spectrogram Plot of the dataset

06

APPROACH TAKEN FOR MODEL DEVELOPMENT:

This code uses a Convolutional Neural Network (CNN) to implement a keyword recognition system. It starts by loading audio files from subfolders, where each subfolder name represents a class label. The audio is pre-processed to ensure uniform length and normalized amplitude. The data is then split into training and validation sets.

The CNN architecture consists of three convolutional layers with increasing filter sizes (32, 64, 128), each followed by max pooling. This structure helps the model learn hierarchical features from the raw audio waveform. The flattened output feeds into a dense layer with dropout for regularization before the final classification layer.

The model is trained using the Adam optimizer and sparse categorical cross-entropy loss. It runs for 50 epochs, which should be enough to see if the model is learning effectively without overfitting. After training, the model is saved for future use. This approach is straightforward and effective for audio classification tasks, though one might want to experiment with hyperparameters or model architecture to optimize performance for their specific dataset.

07

DEVELOPMENT OF CUSTOM DATASET AND FINE-TUNING

As detailed by the task requirements, a custom dataset was prepared with 30 audio samples for each keyword in the original dataset.
The audio was self-recorded with length of the samples ranging from a little less than 1 second to well above 1 minute.

The audio for the keywords was recorded with the microphone array of my PC, in the .wav format and low audio quality, to simulate the conditions in which a mobile virtual assistant is expected to be activated.

5 samples were recorded for the background noise samples. Tapping of a desk, tapping of a pen, the sound of a video, the sound of a song and the sound of jangling keys were used as the “noises” for this task.

In the fine-tuning process, the model had its last two layers removed, after which new layers were sequentially added. These included a Flatten layer to convert the 2D output to a 1D vector, a Dense layer with 128 neurons and ReLU activation for feature extraction, a Dropout layer with a 0.2 rate to prevent overfitting, and a final Dense output layer with softmax activation corresponding to the number of target classes. The model was compiled using the Adam optimizer, sparse categorical crossentropy as the loss function, and accuracy as the evaluation metric. It was trained with a batch size of 16 for 2000 epochs, using a shuffled and batched dataset with prefetching enabled for optimized data loading, and validated on the validation dataset to monitor generalization performance.

08

SUBMISSIONS:

1. A colaboratory file detailing the pre-processing of the Speech Commands dataset, preliminary model definition and training, pre-processing of the custom dataset, fine-tuning of the model and checksum verification of the model assets.
2. A colaboratory file with a rudimentary statistical analysis of the dataset.
3. The project report.
4. The initially developed model in .h5 format.
5. The fine-tuned model in .h5 format.
6. Drive link to the .zip file of the Speech Commands dataset.
7. Drive link to the .zip file of the Custom Dataset.