LAB EVAL 1

UCS749

Conversational Ai: Speech processing and Synthesis

Name	-	Jayant Garg
Roll_no	-	102117156
Group	-	4CS6
Submitted to	-	B.V. Raghav

Paper - Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition

Summary: -

The document describes Google's Speech Commands dataset, designed for training and evaluating keyword spotting systems. It covers the dataset's structure, methodology for data collection, and testing protocols. The dataset targets on-device voice interfaces for limited-vocabulary speech recognition, featuring over 100,000 utterances across 35 words. Evaluation methods and example models are included.

Dataset Description: -

The **Speech Commands Dataset** contains thousands of 1-second audio clips of 35 spoken words, mainly used for keyword recognition. Examples of which used core commands include "yes", "no", "up", "down", "left", "right", "on", "off", "stop", and "go". It also includes background noise and "unknown" labels for other words, making it suitable for training models to recognize spoken commands in real-time applications. The size of the dataset is 2.2 GB.

Data Flow

1. Data Preparation:

The `SpeechCommandsDataset` class wraps the original `torchaudio.datasets.SPEECHCOMMANDS` to preprocess the audio data. Each audio file is padded to 16,000 samples (1 second) using the `pad_audio` function, ensuring consistency. The audio is also reshaped to be a 1D tensor, and a channel dimension is added to prepare it for input into the convolutional layers.

2. DataLoader Setup:

The `SpeechCommandsDataset` is loaded into a `DataLoader` for batch processing, with a batch size of 16 and shuffling enabled for training.

3. Label Counting:

A `Counter` object is used to count the distribution of the labels in the dataset, which helps in converting the labels into indices for model training.

4. CNN Model Definition:

A simple convolutional neural network (`SimpleCNN`) is defined with two 1D convolution layers followed by ReLU activations, an adaptive pooling layer, and two fully connected layers. The model is designed to process audio inputs and classify them into one of the command labels.

5. Model Training:

The model is trained for 5 epochs. In each epoch, for every batch of inputs, the optimizer updates the model weights based on the cross-entropy loss between the predicted outputs and the actual labels (converted to indices using the `label_counter`). The training loss is accumulated and printed at the end of each epoch.

6. Model Evaluation:

- After training, the `evaluate_model` function computes the model's accuracy. It uses the trained model to make predictions on the dataset and compares the predictions with the actual labels. The accuracy is printed as a percentage of correct predictions.