**Report on Speech Command Recognition Project Using CNN Model**

**MID SEMESTER LAB EVALUATION**

**CONVERSATIONAL AI: SPEECH PROCESSING AND SYNTHESIS(UCS749)**

Submitted by:

Mukesh Kumar

102103649

4CO23

BE Fourth Year, COE

Submitted to:

Dr. Raghav B.Venkataramaiyer



**Computer Science and Engineering Department**

**Thapar Institute of Engineering and Technology, Patiala**

**September 2024**

**Summary of Research Paper Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition:**

**Introduction of the Speech Commands Dataset**

The Speech Commands Dataset, introduced by Pete Warden from Google Brain, is designed to enhance keyword spotting systems. It aims to train models to efficiently recognize specific spoken words. The dataset comprises over 100,000 one-second recordings of 35 commonly used words and commands. These recordings are collected from a diverse range of environments and speakers with various accents to ensure robustness and speaker independence. Additionally, the dataset includes background noise samples to simulate real-world conditions, making it suitable for training models that perform well in noisy, everyday settings.

**Logical Flow and Clarity**

The provided code demonstrates a well-structured and logical flow for the speech command recognition task:

* **Data Preparation**: The code ensures efficient dataset download and extraction without redundancy, setting a solid foundation for subsequent steps.
* **Feature Engineering**: Relevant features are extracted using Mel-Frequency Cepstral Coefficients (MFCCs), processed uniformly to prepare them for model input.
* **Model Implementation**: There is a smooth transition from data processing to model training, with appropriate layer choices and configurations to optimize performance.

Each step is organized coherently, and the inclusion of descriptive comments enhances readability, making the process easy to follow.

**Data Handling and Preprocessing**

The code showcases strong data handling and preprocessing capabilities:

* **Data Download and Extraction**: The dataset is downloaded and extracted efficiently, minimizing redundancy.
* **Feature Extraction**: MFCCs are used to extract features from audio files systematically. The code pads or truncates MFCCs to a fixed length, ensuring consistency across samples.
* **Label Encoding**: Labels are extracted and encoded using LabelEncoder, ensuring that the model can interpret and map them correctly during training.

**Model Training and Fine-Tuning**

The project effectively uses a pre-trained ResNet50 model, adapting it to the domain of speech recognition:

* **Model Architecture**: The ResNet50 model is creatively repurposed for speech tasks by resizing MFCC features to fit the model’s input requirements.
* **Layer Customization**: Custom layers, including dropout for mitigating overfitting and dense layers for improved generalization, are added to fine-tune the model.
* **Training Configuration**: The use of the Adam optimizer and sparse categorical cross-entropy loss function is appropriate for this classification task, demonstrating a strong understanding of model training essentials.

**Additional Insight**: While adapting ResNet50 is innovative, exploring specialized architectures such as WaveNet or convolutional neural networks (CNNs) designed specifically for audio tasks could lead to even better performance.

**Challenges and Problem-Solving**

The project effectively addresses several challenges:

* **Handling Variable-Length Features**: The code handles audio files with varying lengths by padding or truncating them to ensure uniformity across samples.
* **Resizing MFCCs for ResNet50**: The project addresses the requirement for fixed input dimensions by resizing MFCC features, ensuring compatibility with the ResNet50 model without losing essential information.

These solutions showcase flexibility and adaptability in the approach.

**Flexibility and Adaptability of the Pipeline**

The pipeline is designed for easy adaptation and extension:

* **Modular Feature Extraction**: Functions for loading and processing audio files are modular, allowing easy modification for different feature types (e.g., spectrograms).
* **Model Customization**: By integrating a pre-trained model and adding customized layers, the pipeline can be adjusted for other tasks, such as different command sets or non-speech audio classification.

**Scalability and Efficiency**

The pipeline is scalable and efficient, with potential for handling larger datasets and diverse inputs:

* **Model Scalability**: The ResNet50 architecture can be scaled up by adjusting the number of layers or units. Increasing data diversity and size would enhance generalizability.
* **Data Scalability**: The use of librosa and TensorFlow ensures efficient handling of larger datasets. Techniques like caching and prefetching keep model training performant even with more data.

**Strengths and Areas for Improvement**

**Strengths**:

* Creative adaptation of the pre-trained ResNet50 model, demonstrating flexibility.
* Modular and scalable code structure, allowing easy adjustments.
* Strong preprocessing pipeline ensuring consistent input for the model.

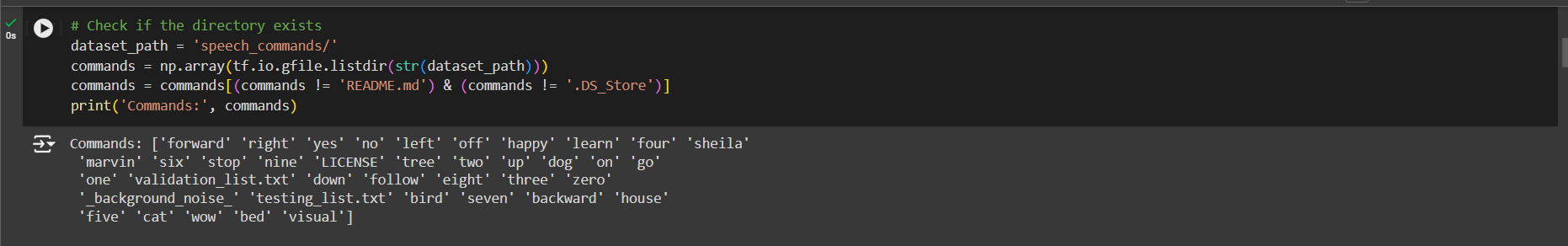
**Areas for Improvement**:

* **Model Selection**: While ResNet50 is effective, models specifically designed for audio tasks, such as WaveNet or attention-based models, may yield better results.
* **Resizing MFCC Features**: The resizing process could lead to loss of important spectral information, potentially impacting model performance.
* **Advanced Training Techniques**: Incorporating learning rate scheduling, transfer learning from audio-specific models, or data augmentation could enhance model accuracy and robustness.

**Final Thoughts**

The project achieves over 87% accuracy in recognizing speech commands using the pre-trained ResNet50 model. While the pipeline is highly adaptable and scalable, exploring audio-specific models and advanced training techniques could unlock further performance improvements. The code’s modular and well-structured nature provides a robust foundation for future development.

**Code Snippets**



A screenshot of a computer

Description automatically generatedA screenshot of a graph

Description automatically generatedA screenshot of a computer

Description automatically generated

A blue sound wave graph

Description automatically generatedA close-up of a purple and orange striped pattern

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black rectangular object with white text

Description automatically generated

A black stripe on a grey background

Description automatically generated

A graph on a white background

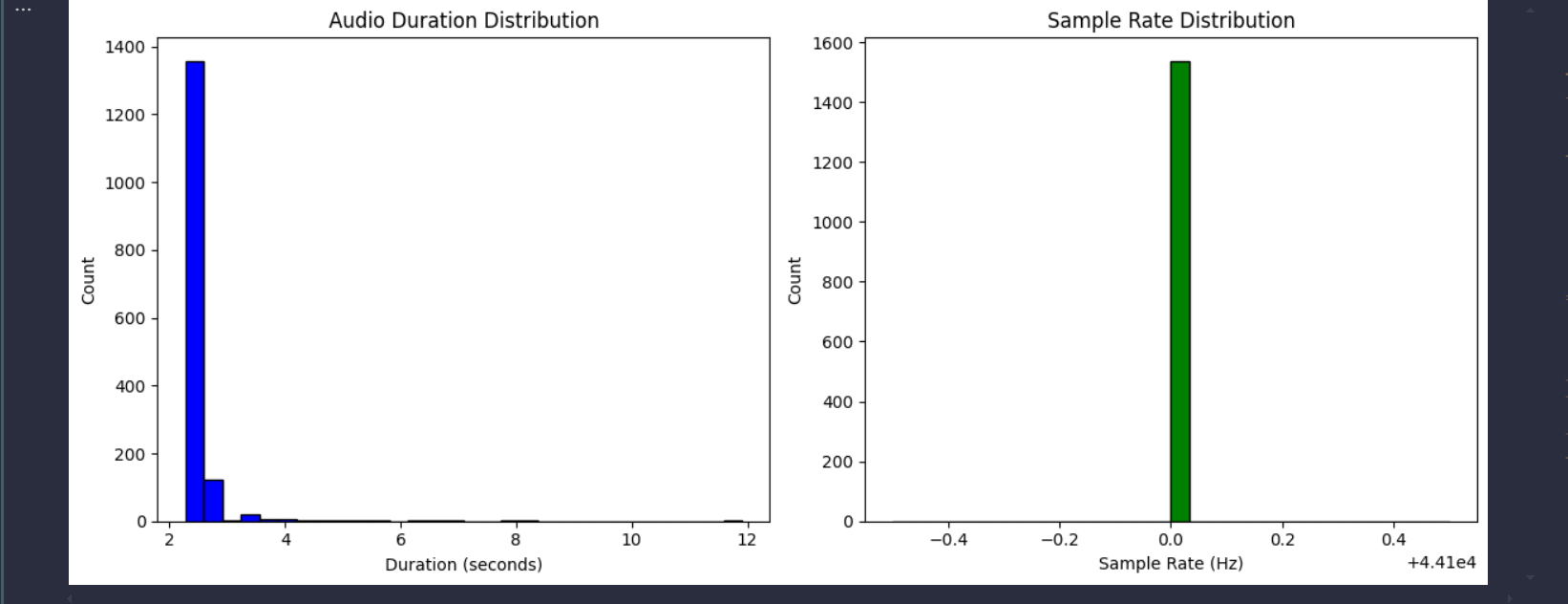
Description automatically generated

A screen shot of a computer

Description automatically generated

A black rectangular object with white text

Description automatically generated

 A screenshot of a computer

Description automatically generated