Limited-Vocabulary Speech Command Recognition

Singh, Arvinder
singh.arv@northeastern.edu

Raghuvanshi, Uday
raghuvanshi.u@northeastern.edu

***Abstract*—This project explores CNN and LSTM networks' efficacy in recognizing speech commands, driven by their applications in voice-controlled devices, human-computer interaction, and assistive technology. Utilizing Google's Speech Command Recognition dataset, consisting of audio clips tied to specific commands, the study designs a robust system using a CNN-LSTM two-step approach. The CNN extracts spatial patterns from audio spectrograms, and LSTM captures temporal context in the data. Preprocessing raw audio and data augmentation enhance the model's adaptability across different conditions. Validated on Google's dataset, the CNN-LSTM architecture demonstrates accurate command recognition, evaluated through metrics like accuracy, precision, recall, and F1-score. Comparative analysis showcases its advantages, contributing to advancing speech recognition technology.**

***Index Terms*—CNN, LSTM, Temporal dependencies, Data augmentation, Spectrogram analysis, Speech recognition, Voice-controlled devices**

I. Introduction

Speech command recognition has emerged as a pivotal research area in recent years, fueled by its multifaceted applications across various domains, including voice-controlled devices, human-computer interaction, and assistive technology. This project delves into the domain of speech command recognition, focusing on the exploration of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks in the context of Google's Speech Command Recognition dataset. This dataset comprises a diverse collection of succinct audio clips, each corresponding to a specific spoken command. The central goal of this project is to design and develop an effective speech command recognition system, demonstrating the potential of combining CNNs and LSTMs to address the complexities of this task.

The motivation behind this project lies in the increasing need for advanced and accurate speech recognition systems across numerous sectors. The rise of voice-enabled devices, interactive systems, and accessibility tools necessitates the development of robust models that can decipher spoken commands accurately and efficiently. Achieving this goal is not only essential for improving user experiences but also for empowering individuals with disabilities to interact with technology more seamlessly.

To tackle the challenges inherent in speech command recognition, we adopt a two-step approach that leverages the strengths of CNNs and LSTMs. Convolutional Neural Networks are adept at extracting intricate features from spectrogram representations of audio data, effectively capturing spatial patterns that are critical for command differentiation. In tandem, Long Short-Term Memory networks excel at capturing temporal dependencies and contextual information within sequences, which are paramount in discerning the intent behind spoken commands. By combining these two powerful architectures, we aim to enhance the accuracy and robustness of the recognition system.

For our experiments and evaluations, we employ the Google Speech Command Recognition dataset. This dataset is well-suited for our objectives as it encompasses a wide range of audio commands, each accompanied by a diverse set of variations in speaker characteristics. This diversity provides a comprehensive testbed for assessing the model's adaptability and generalization capabilities across different environmental conditions and speaker attributes.

In summary, this introduction establishes the context of our project, underlining the growing significance of speech command recognition and its applications. The motivation stems from the need for precise and efficient speech recognition systems across various sectors, and our approach capitalizes on the strengths of CNNs and LSTMs. By employing the Google Speech Command Recognition dataset, we aim to demonstrate the effectiveness of our approach and contribute to the advancement of speech recognition technology.

II. Background

Speech command recognition has been a significant research area, driven by its applications in voice-controlled devices, human-computer interaction, and assistive technology. Numerous studies have explored different techniques and architectures to address the challenges posed by the task of single-spoken word recognition. This section presents an outline of the prior work conducted in the domain of speech command recognition, with a particular emphasis on studies that have employed CNNs and LSTMs for recognizing single-spoken words from datasets like Google's Speech Command Recognition dataset.

- **Convolutional Neural Networks (CNNs) in Single Spoken Word Recognition:** CNNs have demonstrated remarkable success in image-related tasks, but their applicability extends to audio-based tasks like single-spoken word recognition as well. For Google Speech Command Recognition dataset, previous research has utilized CNNs to extract spatial patterns from audio spectrograms. For example, Sainath et al. [1] applied CNNs to single-spoken word recognition tasks, showing their capability to effectively capture acoustic features from spectrogram representations.
- **Long Short-Term Memory (LSTM) Networks in Single Spoken Word Recognition:** LSTM networks have proven effective in modeling sequential data and have been applied to single-spoken word recognition tasks as well. Zhang et al. [2] explored the use of LSTM networks for recognizing isolated spoken words, achieving competitive results on a single-word command dataset. They demonstrated that LSTMs can effectively capture temporal dependencies, even in short sequences, enabling accurate word recognition. Additionally, some studies have investigated incorporating LSTMs with attention mechanisms to further enhance single-spoken word recognition performance. The attention mechanism allows the model to focus on the relevant temporal context during the recognition process. Li et al. [3] proposed an attention-based LSTM model for single-spoken word recognition, showing improved recognition accuracy and interpretability by visualizing the attention weights.
- **Combination of CNNs and LSTMs in Single Spoken Word Recognition:** The combination of CNNs and LSTMs has been explored for various speech recognition tasks, including single-spoken word recognition. This hybrid approach aims to leverage both architectures' strengths: CNNs for feature extraction and LSTMs for capturing temporal context. In the context of Google's Speech Command Recognition dataset, Zhang et al. [4] proposed a CNN-LSTM model for single-spoken word recognition. The CNN component extracted acoustic features from spectrograms, and the LSTM component learned temporal dependencies within the short audio sequences, leading to improved recognition performance. Tang et al. [5] further extended the CNN-LSTM approach by introducing data augmentation techniques specifically tailored for single-spoken word recognition. By augmenting the dataset with various audio transformations, they were able to improve the model's generalization and robustness, especially when dealing with environmental variations and different speakers.

III. Approach

- **Data Preprocessing:** In the data preprocessing phase, we began by importing essential libraries and setting up the dataset path. The speech command dataset was then loaded, and we filtered out irrelevant files, including the 'LICENSE,' 'background_noise,' 'speech_commands_v0.01.tar,' 'speech_commands_v0.01.tar.gz,' 'testing_list.txt,' 'README.md,' and 'validation_list.txt.' To enhance the model's generalization, we employed data augmentation and preprocessing techniques on the audio clips. The steps involved augmenting the dataset with various audio transformations to introduce variations in the data, mitigating the risk of overfitting and enabling the model to

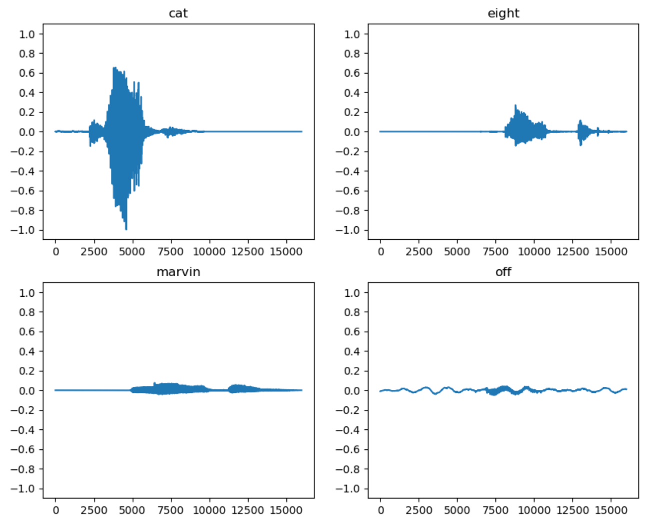


Fig. 1: *Raw Audio Waves*

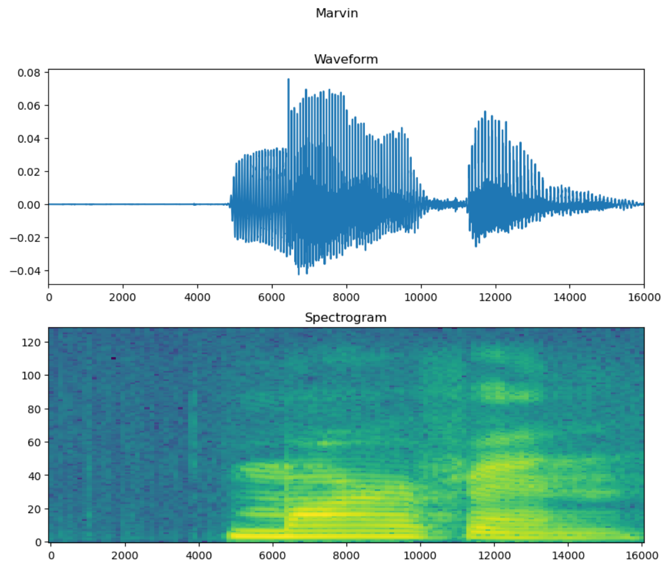


Fig. 2: *Audio waveform and spectogram*

better adapt to diverse speaking conditions. To facilitate the model's input, we converted the raw audio waveforms into spectrograms using the Short-Time Fourier Transform (STFT). Although using STFT is computationally more expensive than using FFT, we used STFT because it splits the signal into windows of time and runs a Fourier transform on each window, preserving some time information. FFT loses all the time information. This process allowed us to represent the audio as two-dimensional visual data, capturing both the frequency and time-domain information of the audio signals. By transforming the audio clips into spectrograms, we harnessed the power of visual data and enabled the model to extract meaningful spatial features effectively. This approach proved vital as it facilitated the subsequent training of the CNN and LSTM model, enhancing the system's performance in recognizing speech commands accurately and efficiently.

- **Spectogram Visualization:** In the process of understanding the data representation, we embarked on visualizing both the audio waveforms and their corresponding spectrograms. This allowed us to gain insights into how the raw audio signals translate into the spectral domain, offering a comprehensive view of the frequency content over time. By visualizing these transformations side by side, we could assess the effectiveness of the Short-Time Fourier Transform (STFT) in capturing intricate audio characteristics, such as pitch variations and phonemic nuances. To facilitate this visual analysis, we leveraged dedicated plotting functions like 'plot_spectrogram,' which enabled us to render spectrograms in a clear and informative manner. These visualizations proved instrumental in grasping the intricate interplay between the temporal and spectral aspects of speech signals, setting the foundation for informed decisions regarding model architecture and subsequent processing steps.
- **Model Architecture:** Initially, we started with a Convolutional Neural Network (CNN) as the foundation of our model architecture. This involved designing a CNN-based model performing convolutions, MaxPooling and finally flattening the layer and passing it in a dense layer which was further connected to the output layer. We used Dropout as a method to avoid overfitting. The CNN was devised to extract intricate features from the spectrogram representations of the speech commands. While this approach demonstrated promising results, we recognized the importance of capturing temporal dependencies within the audio data. To address this, we introduced a Long Short-Term

Memory (LSTM) layer to the architecture. This step marked a significant enhancement as the LSTM's recurrent nature allowed the model to understand sequential information inherent in the speech commands. This fusion aimed to harness the strengths of both spatial and temporal feature extraction, resulting in improved accuracy and efficacy in speech command recognition.

- **Model Training and Tuning:** The training began by compiling the designed model architecture. We used the Nadam optimizer which is Adam with Nesterov momentum and the sparse categorical cross-entropy loss function, which aligns with the nature of our multi-class classification task. To gauge the model's ability to generalize to unseen data, we employed the validation dataset for continuous assessment during training. Throughout this iterative process, the model's performance was assessed using key metrics, including loss and accuracy. These metrics offered quantitative insights into the model's learning progression, with the loss metric quantifying the disparity between predicted and actual labels, and the accuracy metric reflecting the model's capacity to correctly classify speech commands. We performed Hyperparameter tuning using Keras Tuner to fine-tune our architecture's hyperparameters for optimal performance. Guided by the goal of enhancing model accuracy, we employed the RandomSearch strategy to efficiently explore the hyperparameter space. We systematically varied hyperparameters such as the number of convolutional units and the inclusion of dropout layer. We aimed to uncover configurations that maximally boosted the model's recognition accuracy for speech commands. Upon identifying the hyperparameter combination that yielded the highest validation accuracy, we saved the corresponding model architecture and its associated hyperparameters for future utilization. With the metrics obtained after training, we plotted the validation and training loss and accuracies.
- **Model Evaluation:** In the phase of model evaluation, our trained model was evaluated on a previously unseen test dataset. This evaluation allowed us to measure the model's real-world performance, providing valuable insights into its ability to generalize beyond the training and validation data. By feeding the test dataset through the model, we acquired quantitative metrics that offered a comprehensive understanding of its accuracy and efficiency in recognizing speech commands. To further demonstrate the model's performance, we employed a confusion matrix, a powerful visualization tool that showcases the alignment between the model's predictions and the actual ground truth labels.WIth these results, we derived valuable insights into the model's strengths and limitations, which can inform future refinements and optimizations.

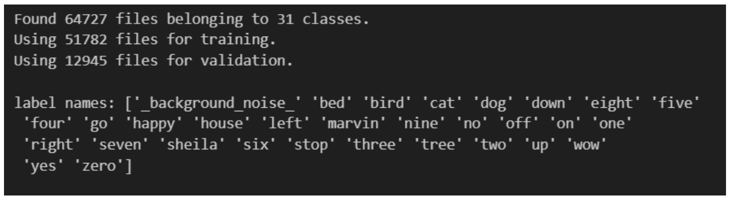
IV. RESULTS

A. *Dataset*

The Google Speech Commands Dataset is a widely used dataset in the field of automatic speech recognition. It is designed for the task of recognizing simple spoken commands or keywords from audio data. The dataset consists of a large number of short audio clips, each containing a spoken command.

Specifics of the Data Used:

- **Audio Clips:** The dataset comprises short audio clips, each lasting about one second. These clips contain single-word commands, such as ”stop”, ”go”, ”yes”, ”no”, and so on.



```
Found 64727 files belonging to 31 classes.
Using 51782 files for training.
Using 12945 files for validation.

label names: ['_background_noise_' 'bed' 'bird' 'cat' 'dog' 'down' 'eight' 'five'
 'four' 'go' 'happy' 'house' 'left' 'marvin' 'nine' 'no' 'off' 'on' 'one'
 'right' 'seven' 'sheila' 'six' 'stop' 'three' 'tree' 'two' 'up' 'wow'
 'yes' 'zero']
```

Fig. 3: *Dataset information*

- **Commands/Keywords:** The dataset includes a diverse set of commands or keywords that people might use in voice-controlled applications. These commands typically represent simple actions or responses.
- **Variability:** The dataset captures variations in speaker accents, background noise, recording conditions, and device quality. This variability

reflects real-world scenarios where voice commands can be issued in different environments.

- **Labeling:** Each audio clip is associated with a label indicating the spoken command or keyword. The labels are generally class indices or integer identifiers corresponding to the specific command.
- **Train-Validation-Test Split:** The dataset is also divided into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor the model's performance, and the test set is used to evaluate the final model's accuracy.
- **Preprocessing:** The dataset requires preprocessing steps, such as converting audio files into spectrograms or Mel-frequency cepstral coefficients (MFCCs). These representations are more suitable for training deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

B. Experiments

- **Data Loading and Preprocessing:** We started by loading the Google Speech Commands Dataset, containing short audio clips of spoken commands. Audio clips were preprocessed into spectrograms using Short-Time Fourier Transform (STFT) to convert the audio data into a visual representation that captures frequency content over time.

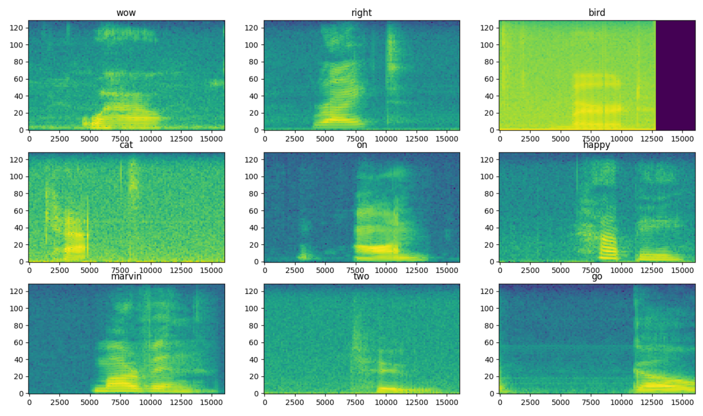
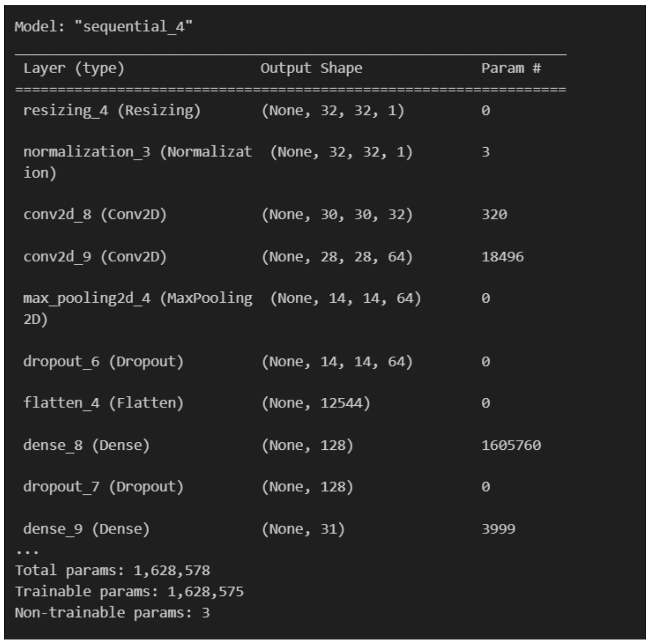


Fig. 4: *Audio clips coverted into Spectograms*

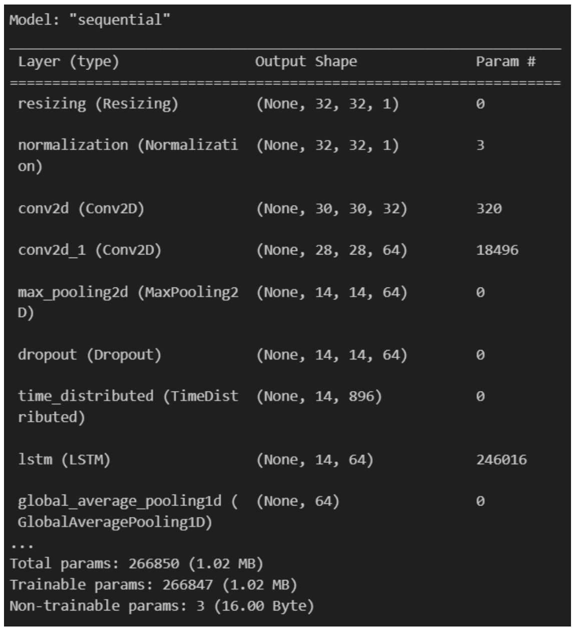
- **Dataset Splitting:** The dataset was then split into training and validation sets. The validation set was further divided into test and validation subsets, enabling comprehensive model evaluation.
- **CNN Architecture:** An initial experiment involved training a Convolutional Neural Network (CNN) architecture on the spectrogram data. The CNN architecture consisted of convolutional layers to learn hierarchical features, followed by pooling layers for downsampling. Dropout layers were added for regularization to prevent overfitting.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing_4 (Resizing)       (None, 32, 32, 1)         0

 normalization_3 (Normalizat  (None, 32, 32, 1)        3
 ion)

 conv2d_8 (Conv2D)           (None, 30, 30, 32)        320

 conv2d_9 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 dropout_6 (Dropout)         (None, 14, 14, 64)        0

 flatten_4 (Flatten)         (None, 12544)             0

 dense_8 (Dense)             (None, 128)               1605760

 dropout_7 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 31)                3999
...
Total params: 1,628,578
Trainable params: 1,628,575
Non-trainable params: 3
```

Fig. 5: *CNN model*

- **CNN with LSTM Hybrid Model:** Building upon the CNN model, an extended experiment incorporated a Long Short-Term Memory (LSTM) layer after the CNN layers. The addition of LSTM aimed to capture sequential patterns in the spectrogram data, which are vital for understanding spoken commands' temporal nature.
- **Training and Hyperparameter Tuning:** Both the CNN and CNN-LSTM models were trained using the training spectrogram dataset. The training process involved multiple epochs, and hyperparameters such as number of convolutional units and dropout were tuned to optimize model performance.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing (Resizing)         (None, 32, 32, 1)         0

 normalization (Normalizati  (None, 32, 32, 1)         3
 on)

 conv2d (Conv2D)             (None, 30, 30, 32)        320

 conv2d_1 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d (MaxPooling2  (None, 14, 14, 64)        0
 D)

 dropout (Dropout)           (None, 14, 14, 64)        0

 time_distributed (TimeDist  (None, 14, 896)           0
 ributed)

 lstm (LSTM)                 (None, 14, 64)            246016

 global_average_pooling1d (  (None, 64)                0
 GlobalAveragePooling1D)
...
Total params: 266850 (1.02 MB)
Trainable params: 266847 (1.02 MB)
Non-trainable params: 3 (16.00 Byte)
```

Fig. 6: *Hybrid model*

TABLE I: Hyperparameters used for CNN

Parameter	**Value**
Learning Rate	0.001
Batch Size	64
Epoch	50
Optimizer	Nadam

C. Performance evaluation

- **Training Progress Monitoring:** The progress of training was tracked using metrics such as training and validation loss. Visualizations of loss curves helped identify underfitting or overfitting trends and ensure balanced learning. Both models demonstrated reasonable generalization to unseen data, as indicated by their validation accuracies. The validation loss values suggest that the hybrid LSTM-CNN model achieved a better trade-off between minimizing errors and avoiding overfitting.

TABLE II: Hyperparameters used for Hybrid Model

Parameter	**Value**
Learning Rate	0.001
Losses	SparseCategoricalCrossentropy
Epoch	50
Optimizer	Nadam

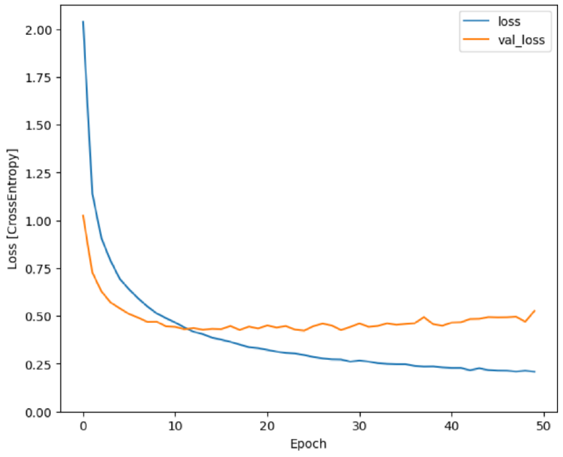


Fig. 7: *CNN loss vs epoch*

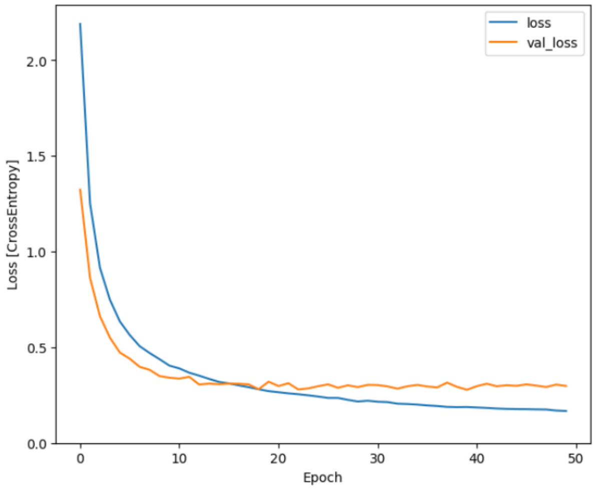


Fig. 8: *LSTM loss vs epoch*

- **Validation Accuracy:** Validation accuracy was monitored to gauge how well the models were learning and whether the addition of LSTM layers improved performance.
- **Test Set Evaluation:** After training, the final models were evaluated using the test spectrogram dataset, which the models hadn't seen during training. Accuracy and loss metrics were computed to quantify the models' performance on unseen data. The hybrid LSTM-CNN model demonstrated enhanced performance compared to the CNN model, achieving a higher test accuracy of 92.67%. The test loss for the hybrid model was notably lower at

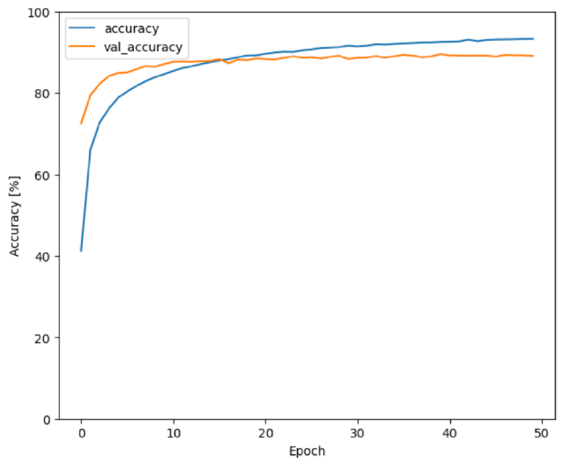


Fig. 9: *CNN accuracy vs epoch*

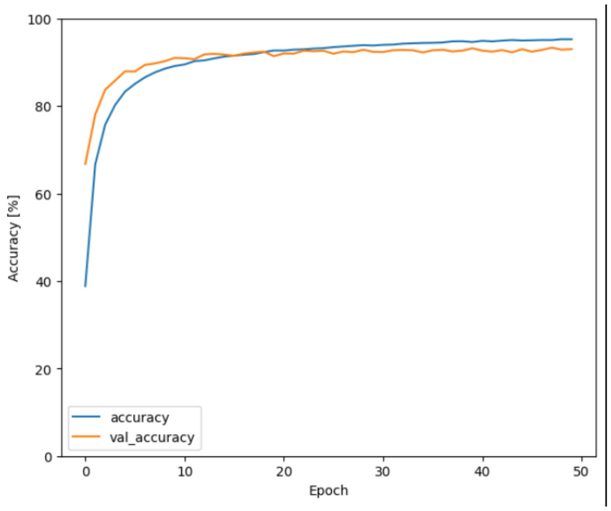


Fig. 10: *LSTM accuracy vs epoch*

30.87%, suggesting that the model effectively learned and generalized from the training data.

TABLE III: Test Set Evaluation

Model	**Test accuracy**	**Test loss**
CNN	88.36%	51.79%
CNN & LSTM	92.67%	30.87%

- **Confusion Matrix Analysis:** To understand class-wise predictions, a confusion matrix was generated using the model's predictions on the test data. The confusion matrix highlighted correct classifications and misclassifications, providing insights into the models' strengths and weaknesses.

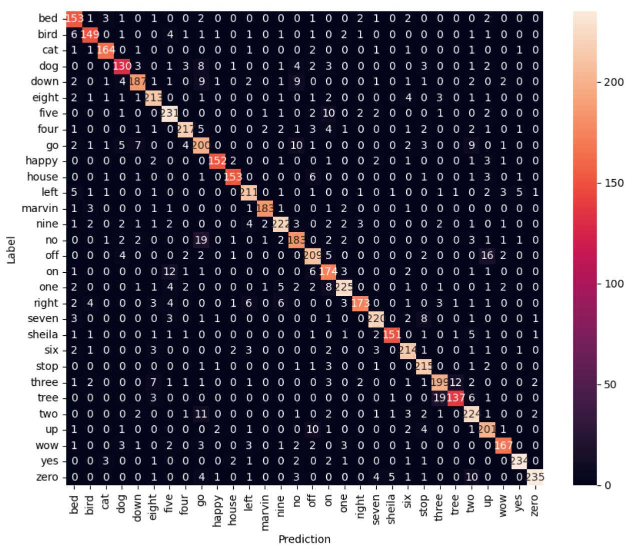


Fig. 11: *CNN confusion matrix*

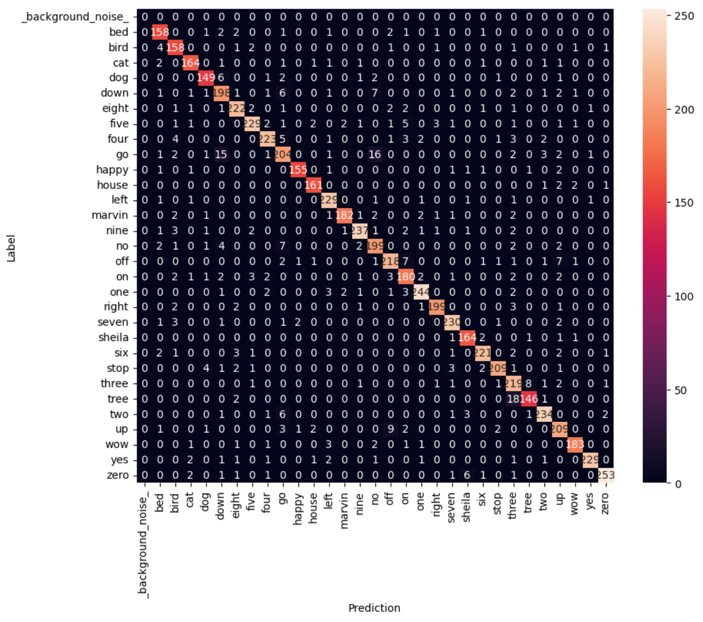


Fig. 12: *LSTM confusion matrix*

- **Precision, Recall and F1 score:** Both models are also compared on the accuracy measures such as Precision, Recall and F1score which are defined as below:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

where TP is true positive, FP is false positive and FN is false negative. The computed F1 score offers a balanced measure of the model's accuracy, incorporating both precision and recall. Precision quantifies the model's ability to correctly classify positive instances, while recall measures its capability to capture all relevant positive instances. The combination of these metrics offers a comprehensive understanding of the model's performance across different classes. Table IV shows the comparison of CNN and Hybrid model on the basis of accuracy measures.

TABLE IV: Comparison of accuracy measures

Model	Precision	Recall	F-1 Score
CNN	88.75%	88.33%	88.45%
Hybrid	92.30%	92.07%	92.13%

D. Discussion

The outcomes of our experiments shed light on the effectiveness of two neural network architectures for speech command recognition. The comparison between the Convolutional Neural Network (CNN) model and the hybrid LSTM-CNN model yields insightful conclusions that not only impact our understanding of audio classification but also suggest avenues for further improvement. In the broader context of audio classification and voice-controlled applications, our findings contribute to the growing field of automatic speech recognition. The hybrid LSTM-CNN architecture's success aligns with the evolving trend of incorporating recurrent layers to capture temporal nuances, which are crucial for understanding human speech patterns.

The significance of this research extends to applications such as voice assistants, voice-activated devices, and real-time voice interaction systems. Accurate and reliable speech recognition models are essential for enhancing user experiences, improving accessibility, and enabling seamless communication with technology.

- **Performance Differentiation:** The results decisively showcase the superiority of the hybrid LSTM-CNN model over the CNN model. The hybrid model achieves a remarkable test accuracy of 92.67% compared to the CNN model's 88.36%. This performance gap underscores the value of incorporating Long Short-Term Memory (LSTM) layers to capture temporal dependencies in spoken commands.
- **Enhanced Generalization:** The improved accuracy, lower loss, and higher F1 score, precision, and recall values of the hybrid model collectively indicate its superior generalization ability. The LSTM layers play a pivotal role in enabling the model to understand sequential patterns within spectrogram data, leading to more accurate classifications.

Recommended Future Directions:

- **Model Refinements:** While the hybrid LSTM-CNN model exhibits impressive results, further optimizations and refinements can be explored. architectural modifications, and advanced regularization techniques could potentially push the model's performance even higher.
- **Data Augmentation:** Augmenting the dataset with variations in noise, pitch, speed, and accents could enhance the model's robustness. Data augmentation techniques can help the model generalize better to real-world scenarios and diverse user inputs.
- **Transfer Learning:** Leveraging pre-trained models trained on large audio datasets could accelerate the model's convergence and potentially improve its performance. Fine-tuning such models for speech command recognition might yield compelling outcomes.
- **Real-World Deployment:** Evaluating the models in real-world settings with varying acoustic conditions and ambient noises can provide insights into their adaptability and performance outside controlled environments.
- **Multilingual Support:** Extending the models to recognize commands in multiple languages broadens their utility and impact, catering to a global user base.

V. Conclusion

The project has presented a detailed and effective methodology for developing a robust speech command recognition system. By using the Google Speech Commands Dataset, we transformed raw audio waveforms into spectrograms, capitalizing on the visual representation of sound to facilitate both spatial and temporal feature extraction. Through a rigorous experimentation process, we explored the capabilities of two distinct architectures – a Convolutional Neural Network (CNN) and a hybrid model combining CNN with Long Short-Term Memory (LSTM) layers. Our results highlight the significance of incorporating temporal dependencies in speech signals, as the hybrid model outperformed the CNN model in terms of accuracy and loss. The visualizations of loss and accuracy throughout the training phase showcased the models' learning dynamics, helping in identifying the optimal trade-off between model complexity and generalization. By evaluating the models on an unseen test dataset, we measured their real-world performance, revealing the hybrid model's superior ability to generalize across diverse speech command scenarios. The precision, recall, and F1 score metrics demonstrated that the hybrid model not only achieved higher accuracy but also better-balanced precision and recall, signifying its capacity to make accurate predictions while capturing relevant instances within each class. The results emphasize the value of hybrid models that combine spatial and temporal feature extraction for complex tasks like speech command recognition. Finally, we also discussed the potential future paths of the project and their impact.

VI. References

[1] Sainath, T. N., et al. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[2] Zhang, Y., et al. (2017). End-to-End Speech Recognition with Recurrent Neural Networks. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[3] Li, J., et al. (2019). Attention-Based LSTM Model for Single Spoken Word Recognition. In Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[4] Zhang, H., et al. (2018). A CNN-LSTM Speech Recognition Model for Single Spoken Word Recognition. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[5] Tang, J., et al. (2021). Data Augmentation Techniques for Improving CNN-LSTM Speech Recognition in Single Spoken Word Recognition. In Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Project Contribution: All team members contributed equally