Conversational AI: Speech Processing and Synthesis

(UCS749)

Recognize My Voice Commands

Submitted by:

Shatakshi Saxena (102117165)
4CS6

Submitted to:

Dr Raghav B. Venkataramaiyer

BE Fourth Year

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Thapar Institute of Engineering and Technology, Patiala

September 2024

Abstract:

This project focuses on developing and fine-tuning an advanced audio recognition system utilizing a deep learning approach. Initially, a pre-trained model from TensorFlow's Speech Commands dataset was employed to handle keyword recognition tasks. The model was fine-tuned using a custom dataset, organized into 35 classes of audio commands, to enhance its performance for specific applications. Given the large size of the dataset, model quantization techniques were applied to reduce the computational load and model size while maintaining accuracy. Quantization-aware training (QAT) was utilized to adapt the model during the fine-tuning process, simulating quantization effects to ensure robustness. The project demonstrates the effectiveness of integrating model quantization in handling extensive audio datasets and optimizing deep learning models for real-world applications.

Introduction:

In recent years, advancements in deep learning have significantly enhanced the capabilities of audio recognition systems, particularly in the realm of keyword and command recognition. Audio recognition systems are crucial in various applications, from virtual assistants and automated customer service to accessibility technologies and personal voice-controlled devices. Voice recognition technology has many real-world applications that can significantly benefit society. It can assist medical professionals in healthcare by transcribing patient notes and enabling hands-free interactions with electronic health records. In smart homes, voice recognition powers devices that control lighting, heating, and security systems, offering convenience and accessibility, especially for individuals with disabilities. For elderly care, voice-activated systems can provide medication and emergency assistance reminders. In customer service, voice recognition improves user experience by allowing efficient and intuitive interaction with support systems. Additionally, it facilitates hands-free operation in vehicles, enhancing safety by enabling drivers to control navigation and communication systems without distraction. These applications enhance productivity, accessibility, and safety, contributing to a more inclusive and efficient society.

This project uses deep learning techniques to develop an audio recognition system that explicitly recognizes user-defined voice commands. The initial phase of this project involves utilizing a pre-trained model from TensorFlow’s Speech Commands dataset, which provides a robust foundation for handling general keyword recognition tasks. This dataset includes a wide range of common commands, serving as a useful benchmark for evaluating model performance. However, to address the specific needs of recognizing personalized voice commands, the project transitions to fine-tuning the pre-trained model using a custom dataset. This dataset is organized into 35 classes, each representing distinct voice commands, including those unique to the user’s vocal patterns.

Given the substantial size of the dataset, model quantization becomes a critical consideration. Quantization is a technique that reduces the precision of the weights and activations in a neural network, converting them from floating-point numbers to lower-bit integers. This process aims to decrease the model's computational footprint and memory requirements while maintaining performance. The project employs Quantization-Aware Training (QAT), a sophisticated approach that simulates quantization effects during the training phase. By incorporating QAT, the model becomes more resilient to the precision loss introduced by quantization, thereby preserving its accuracy and reliability.

Integrating model quantization with fine-tuning allows for efficient handling of extensive audio data while optimizing the model for real-world applications. This approach enhances the model's ability to recognize a broad spectrum of voice commands and ensures that it remains computationally feasible for deployment in resource-constrained environments.

Thus, the project bridges the gap between general audio recognition and personalized command recognition through advanced deep-learning techniques and model optimization strategies. By leveraging a pre-trained model, fine-tuning with a custom dataset, and applying quantization techniques, the project demonstrates a comprehensive methodology for developing and deploying an effective voice command recognition system.

Related Works:

[Reference Paper summary in around 50 words]

The paper "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition" by Pete Warden (2018) discusses a speech commands dataset comprising 105,829 utterances recorded from 2,618 speakers optimized for training and evaluating keyword spotting models. It provides detailed evaluation metrics, including Top-One Error and Streaming Error Metrics. The dataset's applications range from optimized neural network operations on ARM microcontrollers to adversarial attack testing and improvements in noise tolerance. The second version of the dataset demonstrates enhanced results compared to the original. [For more details, refer to the paper: Warden, P. (2018). Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. arXiv:1804.03209v1 [cs.CL]. Google Brain, Mountain View, California. Available at: arXiv.]

Dataset Description:

This project utilizes two distinct datasets for training and fine-tuning the model. The first dataset is the widely recognized "Speech Commands" dataset introduced by Pete Warden (2018) in his paper Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. This dataset consists of 105,829 utterances recorded by 2,618 speakers and 35 distinct voice commands. The dataset was designed to train and evaluate keyword spotting models and features a range of metrics such as Top-One Error and Streaming Error Metrics for assessing performance across various environments, including adversarial attacks and noise-tolerance improvements.

The second dataset is a custom-built collection, explicitly capturing the user's voice. It is created using personal voice recordings of the exact keywords as in the original dataset and recorded using a personal device to simulate real-world conditions which hereby allows the model to adapt to the specific voice characteristics of the user. This dataset was curated to enhance the model's performance in recognizing personalized voice commands. The custom dataset contains the same 35 command categories as the original dataset, including commands such as 'yes', 'no', 'up', 'down', 'left', 'right', 'stop', 'go', ‘ 'zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'backward', 'bed', 'bird', 'cat', 'dog', 'follow', 'forward', 'happy', 'house', 'learn', 'marvin', 'sheila', 'tree', 'visual', 'wow', and others, providing a comprehensive set of keywords for training. This dataset enables the fine-tuning of the model to accurately recognize the user's vocal patterns, thereby personalizing the audio recognition system. By combining the general-purpose dataset from Warden's research with the custom dataset tailored to the user's voice, the project ensures both broad keyword recognition capabilities and precise identification of personalized commands. The data is then splitted as per training, testing and validation set.

Model Used:

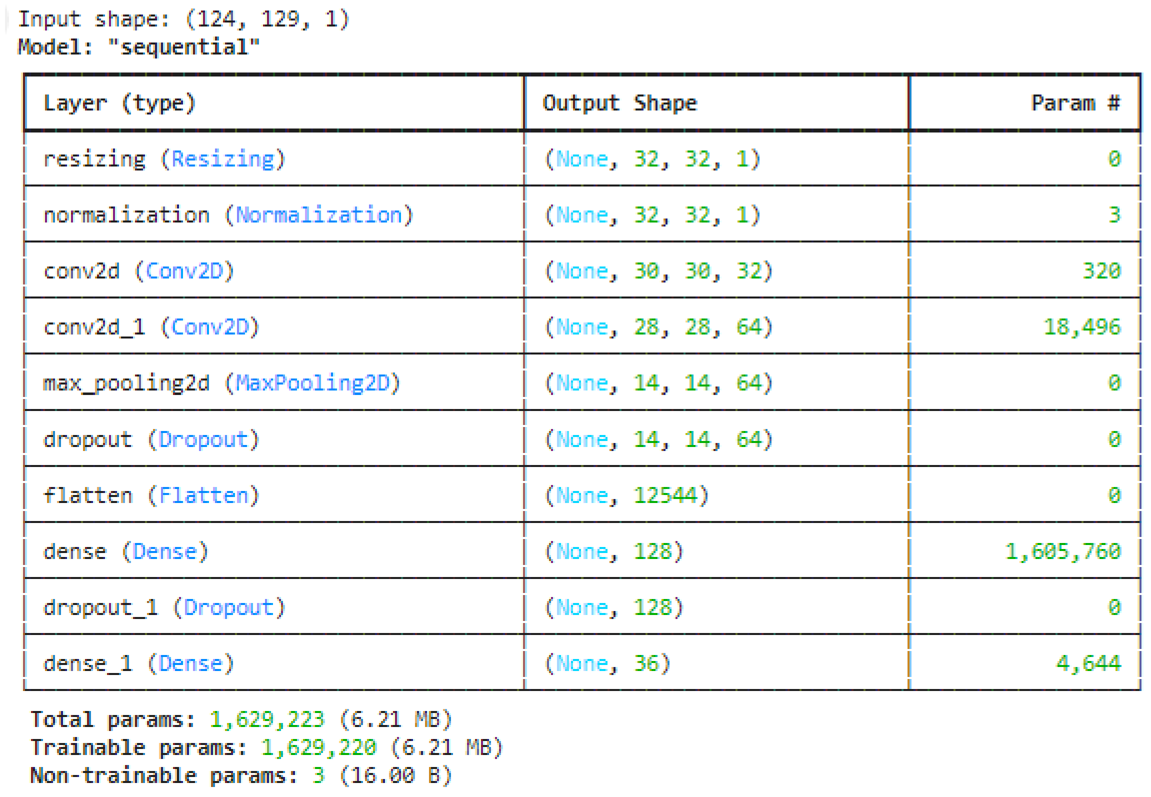
This project employed a sequential **Convolutional Neural Network (CNN)** for keyword recognition. CNNs are particularly effective for audio signal processing as they are well-suited for handling time-series data, such as waveforms and spectrograms, due to their ability to capture spatial hierarchies. The sequential model architecture, in this case, consists of a series of convolutional layers followed by pooling layers, which progressively extract high-level features from the raw audio input.

The audio waveforms are first transformed into spectrograms, representing the signals' frequency content over time. These spectrograms are then passed through the CNN, where convolutional layers apply filters to learn local patterns in the data. Pooling layers are used to downsample the feature maps, reducing the spatial dimensions and computational complexity while retaining essential information.

The sequential CNN architecture is chosen for its simplicity and efficiency, making it highly suitable for real-time applications like voice command recognition. By stacking convolutional and pooling layers, the model can effectively learn from large datasets, such as the Speech Commands dataset, while fine-tuning to recognize personalized voice commands from the custom dataset. The model’s output is processed through dense layers and a softmax activation to predict the probability distribution across the 35 possible commands.

This approach allows for efficient training and inference while also being flexible enough to accommodate the large amount of data and the fine-tuning process needed to recognize personalized commands.

The model used for this project is a sequential Convolutional Neural Network (CNN) designed for efficient audio recognition tasks. It takes an input shape of (124, 129, 1) and begins with a resizing layer to normalize the input to a fixed size of (32, 32, 1). The architecture includes two convolutional layers: the first with 32 filters and the second with 64 filters, each followed by ReLU activations. These layers capture essential features from the spectrograms generated from the audio signals. After each convolutional block, max-pooling and dropout layers are applied to reduce overfitting and downsample the data. The features are then flattened and passed through two fully connected (dense) layers, the first with 128 neurons and the second with 36 output neurons, corresponding to the 35 command labels plus one background noise class. The model is compact, containing approximately 1.6 million trainable parameters, making it suitable for real-time recognition tasks while handling large datasets efficiently. Hence, the model architecture begins with an input layer that accepts the spectrogram representation of audio, followed by preprocessing layers for resizing and normalization to enhance processing efficiency. It incorporates multiple convolutional layers with progressively increasing filters to capture complex patterns within the audio data. Pooling and regularization layers are employed to reduce spatial dimensions and mitigate overfitting. Finally, fully connected layers process the extracted features to provide the final classification. The output layer highlights and classifies the spoken word, enabling effective voice command recognition.



```
Input shape: (124, 129, 1)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ resizing (Resizing)             │ (None, 32, 32, 1)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ normalization (Normalization)   │ (None, 32, 32, 1)      │             3 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d (Conv2D)                 │ (None, 30, 30, 32)     │           320 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 28, 28, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 14, 14, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 14, 14, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 12544)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │     1,605,760 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 36)             │         4,644 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 1,629,223 (6.21 MB)
 Trainable params: 1,629,220 (6.21 MB)
 Non-trainable params: 3 (16.00 B)
```

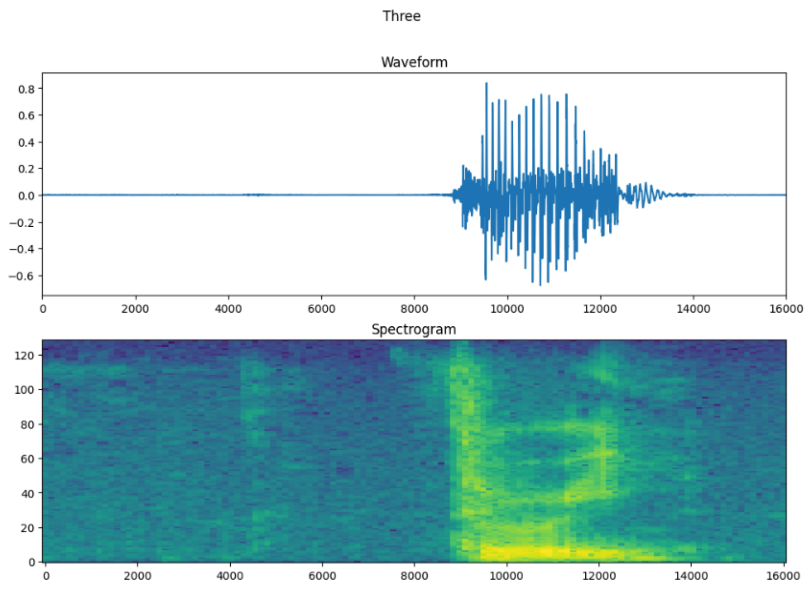
Model Finetuning:

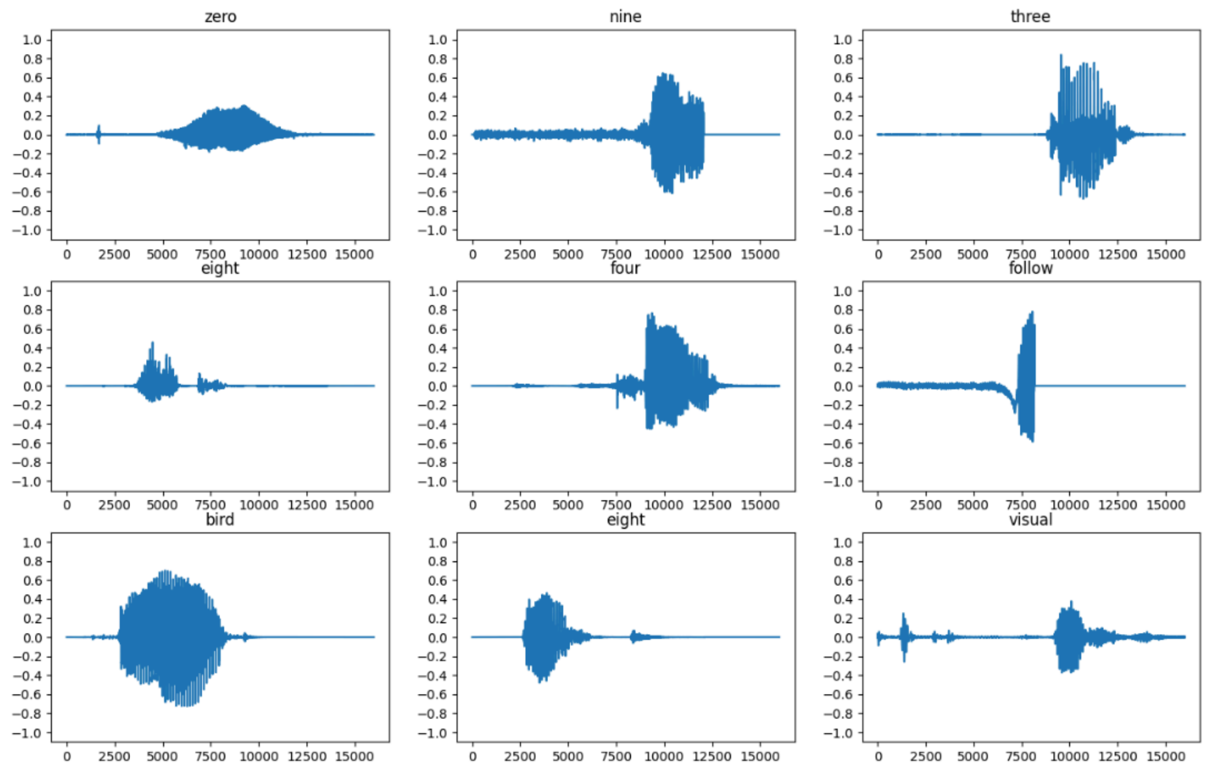
To fine-tune the pre-trained sequential CNN model for recognizing personalized voice commands, the custom dataset was extracted from a .tar file and loaded into training and validation subsets using TensorFlow's audio_dataset_from_directory. Quantization-aware training (QAT) was applied by wrapping the convolutional and dense layers with QuantizeWrapper, optimizing the model for deployment on resource-constrained devices. The model was then compiled with a reduced learning rate and fine-tuned on the custom dataset for 10 epochs, utilizing callbacks for model checkpointing and early stopping. After fine-tuning, the model was evaluated and saved in a quantized format to ensure efficient inference while maintaining performance.

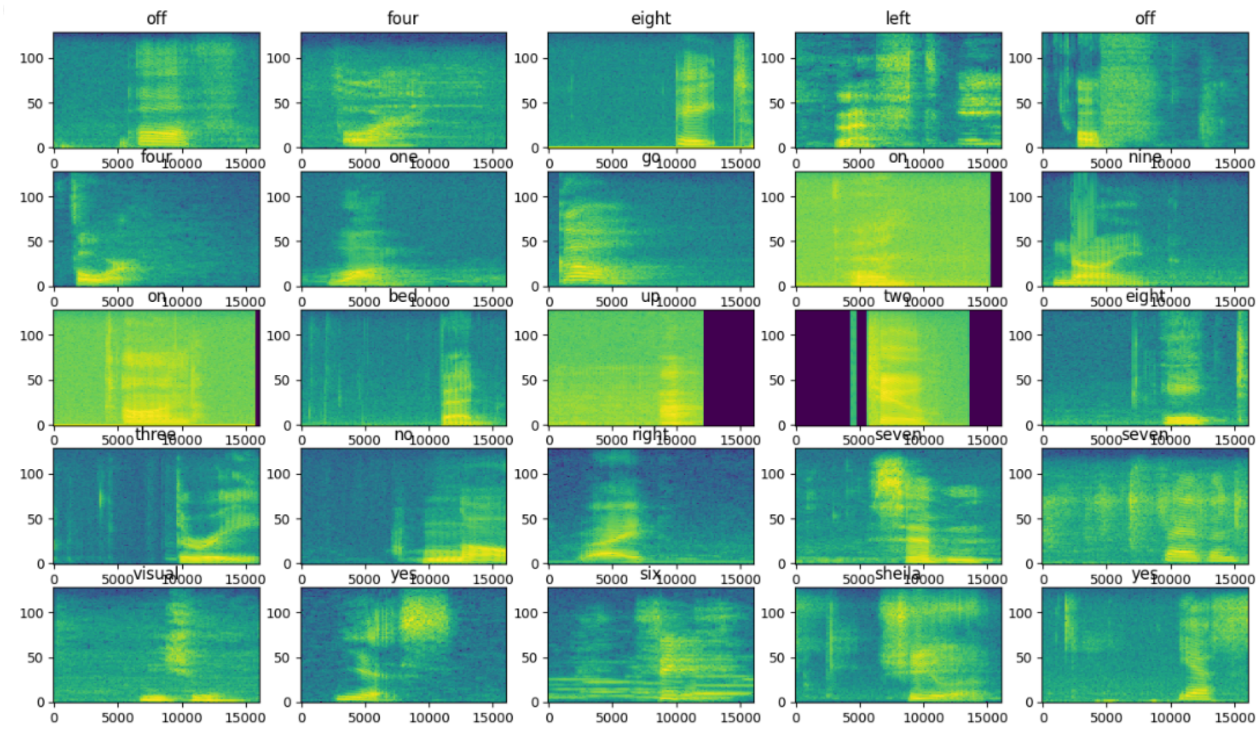
The fine-tuning parameters for the model were configured to optimize performance on the custom dataset. The training was performed on 2 epochs with a focus on leveraging the custom dataset for improved accuracy. Key parameters included the use of tf.keras.callbacks.ModelCheckpoint will save the model's best version during training, ensuring the preservation of the highest-performing weights. Additionally, tf.keras.callbacks.EarlyStopping was employed to halt training early if no improvement in validation loss was observed for three consecutive epochs, thus preventing overfitting and optimizing training efficiency.

Data Analytics and Visualization:

Following is the statistical and data analytics of the project:





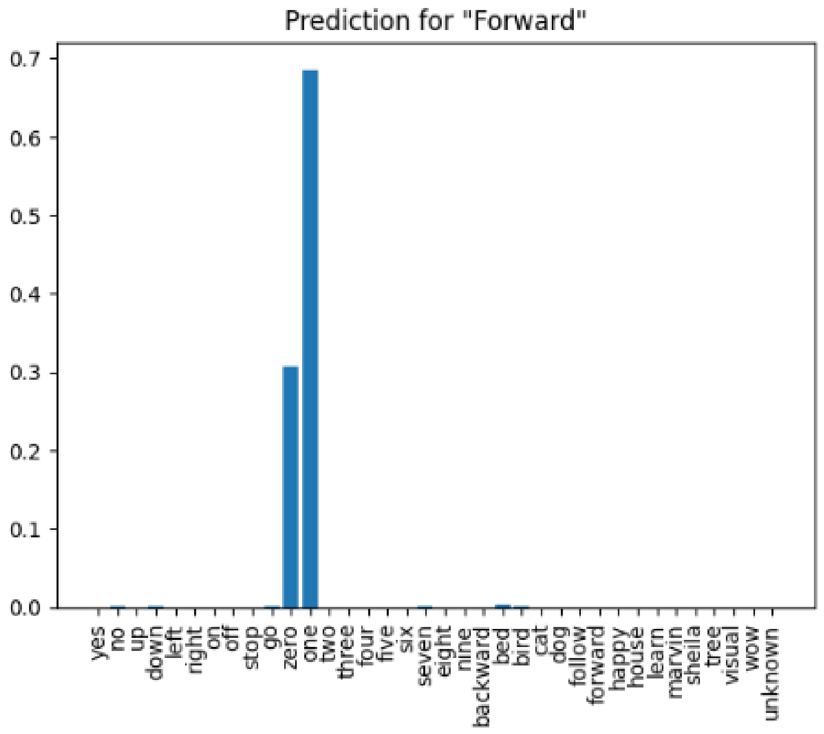


Evaluation Metrics:

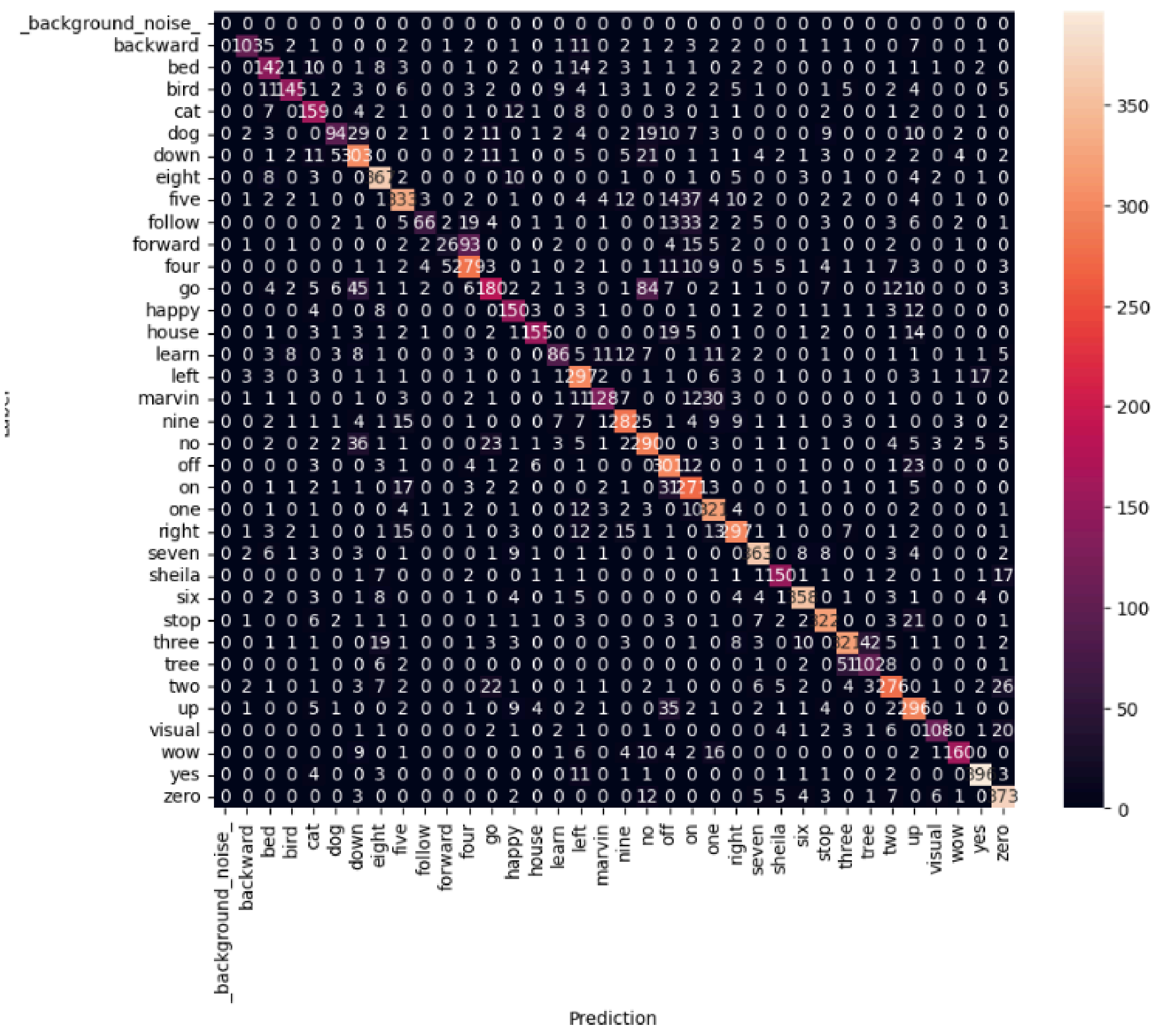
The model's performance was evaluated using several key metrics: accuracy, which measures the proportion of correctly classified instances among all predictions, and loss, which quantifies the difference between the predicted and actual values. The accuracy metric directly assesses the model's effectiveness in recognizing and classifying voice commands. In contrast, the loss metric helps monitor the model's learning progress and detect potential issues such as overfitting. These metrics collectively ensure a comprehensive evaluation of the model's performance on the training and validation datasets.

Result Analysis:

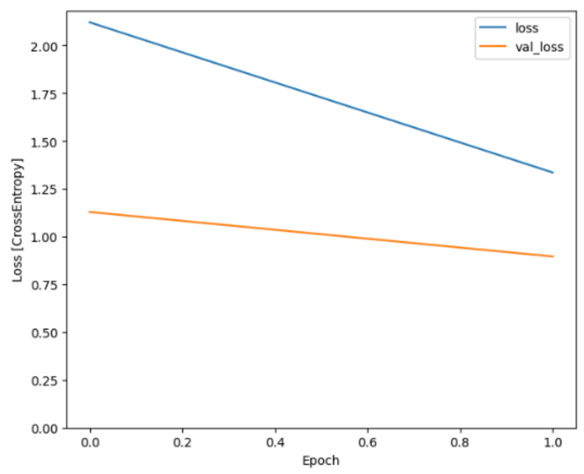
Following is the result analysis of the project:

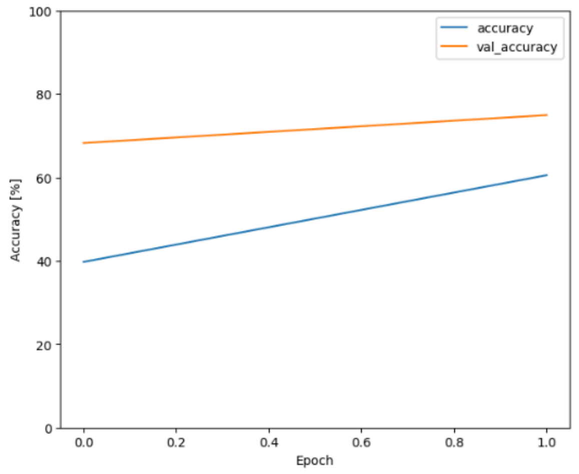


(Tested on a speech containing the term ‘forward’)



```
Text(0, 0.5, 'Accuracy [%]')
```





Conclusion:

In conclusion, the project successfully adapted a pre-trained sequential CNN model for custom voice command recognition by fine-tuning it on a personalized dataset of voice recordings. The integration of quantization-aware training facilitated optimizing the model for deployment on resource-constrained devices, achieving a balance between performance and computational efficiency. The fine-tuning process, enhanced by strategic callbacks for model checkpointing and early stopping, ensured robust model training and prevented overfitting. The final quantized model demonstrated effective recognition capabilities on both the original and custom datasets, highlighting its practical applicability in real-time, resource-efficient voice command systems. Future work could explore expanding the system's use to various applications such as smart home automation, where voice commands could control devices like lighting and temperature, or in wearable technology for hands-free interaction with digital assistants. Additionally, integrating the system with natural language processing could enable more complex voice interactions and commands.