Detection of Human Voice Presence in Space Using Microphone and ESP32 Microcontroller

Askar Anafin, Merey Bissenbin, Rakhat Baltabekov, Dinmukhamed Issenov
Guided By - Dmitrios Zormpas, dimitrios.zorbas@nu.edu.kz
Astana, Kazakhstan
askar.anafin@nu.edu.kz, merey.bissenbin@nu.edu.kz, rakhat.baltabekov@nu.edu.kz, dinmikhamed.issenov@nu.edu.kz
April 19, 2024

***Abstract*—This project explores the development and performance of a voice recognition system implemented on an ESP32 microcontroller, utilizing a 1D Convolutional Neural Network (CNN) architecture. The system's objective is to detect human presence by recognizing individual vocal characteristics through real-time audio input. The research extends into quantization techniques, employing the EON compiler to optimize the CNN model for efficient execution on the constrained hardware, reducing memory and flash usage while maintaining accuracy.**

The system was evaluated on a dataset split into training and testing subsets, achieving a remarkable accuracy of 90.72% on the testing set, surpassing the initial accuracy target of 80% set during the project's inception. The integration of the MAX9814 microphone with the ESP32's Direct Memory Access (DMA) and built-in I2S protocols enabled high-fidelity audio recording without delays. This project not only confirms the feasibility of deploying machine learning models on low-resource microcontrollers but also provides a foundation for future enhancements in biometric-based security and personal identification systems.

***Index Terms*—Machine Learning, Model Compression, Voice Detection, Human Presence, Convolutional Neural Network, Microcontroller**

I. Introduction

Voice recognition is a biometric method that involves recognizing individuals based on their unique vocal characteristics. This widely used and advantageous biometric approach can be utilized for tasks such as verifying identity, ensuring security, and forensic voice verification for suspect detection. The working voice recognition system would improve different aspects of people's lives all over the world.

The system consists of both hardware and software components. The conceptual view of our system is as follows: A microcontroller device with a microphone takes an audio input to decide whether the particular human individual is present. The actual, detailed view of our system is that we have a microcontroller (ESP32) that will run code on the edge to detect a human individual's voice in space using input from a microphone in real time. The system sends the result to the serial monitor of a laptop. The major part of such a system is the choice to construct a system due to the memory and computational constraints of ESP32. The architecture of our system including interaction between hardware and software parts is:

1) The trained machine-learning model and code for the digital signal processing are stored on the microcontroller.
2) The system converts audio input to an MFCC spectrogram.
3) The spectrogram is fitted to the CNN model.
4) The probabilities of audio belonging to the target person are calculated and sent to the serial monitor.

II. Project Development

A. Background and Related Work

Before getting into work, it was important to review the literature to learn what already has been done and what we can use. We started by getting familiar with the ESP32 and the MAX9814 microphone. We used Espressif's official documentation to explore ESP32[1] and Maxim's documentation to explore MAX9814[2]. ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility, and reliability in a wide variety of applications and power scenarios and it has 520Kb RAM. We learned about the ESP32's Direct Memory Access (DMA) to record audio without delay and built-in I2S protocols to record high-frequency audio from tutorials of the GitHub user named atomic14[3]. The starting point of our project was a public project on YouTube that performed keyword detection with ESP32 and TensorflowLite. [4] However, there is no project on public access identical to our project. For the feature generation part of our project, we used the Edge Impulse platform. [5] Edge Impulse is a leading development platform designed specifically for edge machine learning (ML). It enables developers to create intelligent device solutions that can process and react to real-world data at the location where it is captured, minimizing latency and bandwidth use.

B. Project Approach

We can divide our system into three parts: audio recording, feature extraction, and inferencing. The audio recording part was performed using the built-in ESP32 I2S config with a 20KHz frequency, and 16-bit depth. Since MAX9814 is an analog microphone the audio recording includes Analog-to-Digital conversion (ADC). The DMA allowed us to perform

```
static int i2s_init(uint32_t sampling_rate) {
  i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX | I2S_MODE_ADC_BUILT_IN),
    .sample_rate = sampling_rate,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_STAND_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 16,
    .dma_buf_len = 1024,
    .use_apll = false,
    .tx_desc_auto_clear = false,
    .fixed_mclk = 0};

  adcSampler = new ADCSampler(ADC_UNIT_1, ADC1_CHANNEL_7, i2s_config);
  adcSampler->start();

  esp_err_t ret = 0;

  return int(ret);
}
```

Fig. 1. MAX9814 microphone configuration

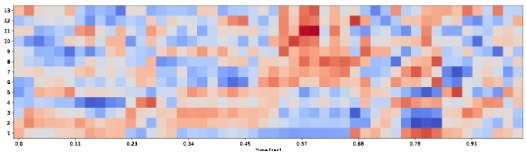

Fig. 2. Cepstral Coefficients of audio

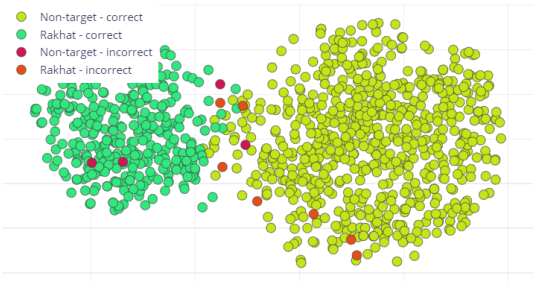

Fig. 3. Training Data Distribution

```
# model architecture
model = Sequential()
model.add(Reshape((int(input_length / 13), 13), input_shape
    =(input_length, )))
model.add(Conv1D(8, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Conv1D(16, kernel_size=3, padding='same', activation='relu'
    ))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(classes, name='y_pred', activation='softmax'))
```

Fig. 4. 1D CNN model Architecture

ADC without delay and avoid data loss. For the audio processing part, we used the MFCC features (See Fig.2). Because the MFCC uses the Mel scale, it approximates human voice behavior well.[6]

For the classification model, we chose a 1D Convolutional Neural Network with 2 layers(see Fig. 4 and Table I). The neural network has two output classes Rakhat and Non-target. Rakhat is the target person that our model is trained to recognize.

The next step was to quantize our model to make it suitable for microcontrollers. We used a new EON (Edge Optimized Neural) compiler that runs a neural network with 25-55% less RAM and up to 35% less flash, while retaining the same accuracy, compared to TensorFlow Lite for Microcontrollers.[7] This approach provided us the opportunity to save our high accuracy.

C. *Project Execution*

During the model testing part, we faced problems with the volatility of the human voice. The human voice depends on many factors such as emotion, attitude, the language spoken, the content of speech, and many others. After training the model we couldn't perform live testing because we didn't understand does our system performed poorly or if the voice of the speaker Rakhat differed due to such reasons as sore throat.

Input Layer (650 features)
Reshape Layer (13 columns)
1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)
Dropout (rate 0.25)
1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)
Dropout (rate 0.25)
Flatten Layer
Output Layer (2 classes)

TABLE I
1D CNN MODEL ARCHITECTURE

For some time, we couldn't record high-quality audio due to a mistake related to the baud rate (the data transfer capability of the USB cable). We thought that this was the restriction of the analog microphone or this was due to the delay of analog to digital conversion. However, after we figured out how DMA works and even programmed audio recording to the single core of ESP32 there was still a problem of discontinuous audio and the low quality. Then, we calculated the actual stream of data coming through the USB cable and we understood that the current baud rate is much less than the actual data stream. Therefore, we lost part of our data. After setting up the maximum baud rate we solved that problem.

Initially, when we didn't how to perform audio feature extraction on ESP32 we thought that we would fit the raw data into a deep learning model. After the literature review, we understood that this approach leads to low accuracy[8]. Then, we found the Edge Impulse platform and used the existing libraries to perform feature generation on the microcontroller. Actually, feature generation was the hardest part of our project.

III. PROJECT EVALUATION

The primary evaluation metric utilized for our system was accuracy, which was crucial for determining the reliability of voice recognition under real-world conditions. The final accuracy achieved on the testing dataset was an impressive 90.72%, which significantly exceeds the initial goal of 80%

	Non-target	Rakhat
Non-target	98.5%	1.5%
Rakhat	1.4%	98.6%
F1 Score	0.99	0.98

TABLE II
ACCURACY OF TRAINING DATA

	Non-target	Rakhat	Uncertain
Non-target	92.0%	5.0%	2.9%
Rakhat	6.7%	89.3%	4.0%
F1 Score	0.93	0.92	

TABLE III
ACCURACY OF TESTING DATA

accuracy set at the project's outset(**see Table** III). This high level of accuracy demonstrates the model's robustness and its capability to effectively differentiate between the target and non-target voices based on the processed audio inputs.

To measure this accuracy, the data was split into training and testing subsets, providing a controlled environment to evaluate the model's generalization beyond the data it was trained on. The model's performance was further quantified using the F1 score, a harmonic mean of precision and recall, which was calculated for each class (target and non-target) in the testing data. The results showed F1 scores of 0.99 for the target and 0.98 for non-target classes, indicating a highly balanced and effective classification system(**see Table** II).

Finally, as shown in Fig. 5, the accuracy of the 1D CNN model in real time scenario is around 90%. It takes about 300 ms to generate features for real-time audio input and only 4 ms for the ML model to classify this as target or non-target voice. The size of the optimized model obtained through the EON compiler is 31.5Kb which is more than the non-optimized float32 model which is 27.1Kb. However, the quantized model uses 3.8Kb of RAM than 7Kb of the non-optimized model. The RAM usage is more important for us than the actual size of the model and we sacrificed some memory in order to achieve less RAM memory and execution time.

IV. CONCLUSION

In conclusion, this project has effectively shown that it's possible to implement a voice recognition system on an ESP32 microcontroller using a 1D Convolutional Neural Network. The system exceeded the initial accuracy target set at the beginning of the project by achieving an impressive 90.72% accuracy on the test dataset, underscoring the strength of the machine learning model used.

The EON compiler was instrumental in quantizing the model, significantly reducing memory and computational resource usage without compromising performance. This not only makes the system compatible with low-resource hardware but also upholds the high operational standards necessary for real-time applications. The optimization strategies and incorporation of superior audio processing techniques have successfully overcome the hardware limitations of the ESP32.

Furthermore, the project tackled key challenges such as real-time audio processing and data loss due to baud rate discrepancies, demonstrating the team's capacity to innovate despite technical constraints. The solutions put in place have boosted the system's reliability and efficiency, rendering it a practical choice for use in security systems, personal identification, and voice-interactive devices.

Possible Future Work

```
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.851562
    Rakhat: 0.148438
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.996094
    Rakhat: 0.003906
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.964844
    Rakhat: 0.035156
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.886719
    Rakhat: 0.113281
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.996094
    Rakhat: 0.000000
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.929688
    Rakhat: 0.070312
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.031250
    Rakhat: 0.968750
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.035156
    Rakhat: 0.964844
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.695312
    Rakhat: 0.304688
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.015625
    Rakhat: 0.984375
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.027344
    Rakhat: 0.972656
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.988281
    Rakhat: 0.011719
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.996094
    Rakhat: 0.000000
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.984375
    Rakhat: 0.015625
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.996094
    Rakhat: 0.000000
Predictions (DSP: 302 ms., Classification: 4 ms., Anomaly: 0 ms.):
    Non-target: 0.980469
    Rakhat: 0.019531
```

Fig. 5. Accuracy of Real-Time Model

In the future, we can construct a more complex neural network model to achieve better accuracy. Our current model weighs less than we initially expected and we are sure that ESP32 is capable of running bigger neural networks. The area of possible future improvements may include:

- **Increasing the dataset**. It is a common practice to increase the dataset to obtain better results. Since human voice is volatile to different life situation, it would be of significant addition to include target voice in different scenarios(e.g. different emotional states, during sickness).
- **Trying out other ML models**. Current results are based on MFCC method for feature extraction and CNN ML architecture. It would be a good practice to try newer machine or deep learning models to acquire similar or better accuracy score while maintaining model's weight within the power and memory constraints of ESP32.
- **Classifying multiple targets**. Current model is set to binary classification resulting in only in true or false output. By training ML model for bigger dataset that include several people, we could have implement multi-class ML model that is able to identify the number of people in the room and who are they by their voice.

REFERENCES

[1] M. Araya-Salas and G. Smith-Vidaurre, "Espressif," accessed on November 22, 2023. [Online]. Available: https://www.espressif.com/

[2] librosa development team, "Microphone amplifier with agc and low-noise microphone bias," accessed on November 24, 2023. [Online]. Available: https://www.analog.com/en/index.html

[3] C. Greening, “Esp32 audio input,” 2020, accessed on November 25, 2023. [Online]. Available: https://www.atomic14.com/2020/09/12/esp32-audio-input.html
[4] C. Greening, “Diy alexa: Create your own voice assistant with esp32 & tensorflow lite,” 2020, accessed on December 22, 2023. [Online]. Available: https://www.youtube.com/@atomic14
[5] “Edge impulse,” accessed on March 25, 2024. [Online]. Available: https://edgeimpulse.com/
[6] J. Martinez *et al.*, “Speaker recognition using mel frequency cepstral coefficients (mfcc) and vector quantization (vq) techniques,” in *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, 2012, pp. 248–251.
[7] J. Jongboom, “Introducing eon: Neural networks in up to 55
[8] N. H. Tandel, H. B. Prajapati, and V. K. Dabhi, “Voice recognition and voice comparison using machine learning techniques: A survey,” in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 459–465.