Design Of Emergency Keyword Recognition Using Arduino Nano BLE Sense 33 And Edge Impulse

Danyar N. Karim[1], Noraimi Shafie[2], Azizul Azizan[3]
[1,2,3]Univesiti Teknologi Malaysia
[1]danyar.nabaz@gmail.com, [2]noraimi.kl@utm.my, [3]azizulazizan@utm.my

Article history

Received:
28 Oct 2023

Received in revised form:
10 Nov 2023

Accepted:
16 Nov 2023

Published online:
18 Dec 2023

*Corresponding author
danyar.nabaz
@gmail.com

Abstract

This project focuses on Custom Keyword Voice Recognition (CKVR) for emergency response scenarios. A multilingual keyword spotting system is developed using the Arduino Nano 33 BLE Sense board and Edge Impulse. The system accurately recognizes the keyword "help" in English, Arabic, Kurdish, and Malay languages. The project utilizes Mel Frequency Cepstral Coefficients (MFCC) for feature extraction and employs deep learning techniques for model training. By optimizing the model through quantization and achieving 100% accuracy in training and testing phases, the system provides a reliable solution for identifying emergency keywords. The developed system has the potential to enhance safety in public spaces such as malls, hospitals, schools, and stations by quickly responding to individuals in distress. The project demonstrates the effectiveness of the chosen approach, highlighting the significance of MFCC processing, classification learning, and optimized model design in speech recognition. The successful development of this keyword spotting system opens doors for further advancements in the field and emphasizes the potential for innovative solutions that contribute to a safer and more responsive world.

Keywords: *Machine learning, edge impulse, Arduino nano BLE 33 sense, keyword spotting, keyword recognition*

1. Introduction

The field of speech recognition has seen tremendous growth in recent years, owing to its wide array of applications in modern technology. From powering AI assistants like Alexa and Siri to enabling voice-activated controls in smart homes, speech recognition is revolutionizing human interaction with electronic devices. An essential facet of this field is Custom Keyword Voice Recognition (CKVR), which focuses on identifying specific keywords within audio samples.

The current project is on capitalizing the potential of CKVR in emergency response scenarios. The development of a multilingual keyword spotting system using machine learning are focus on designed to identify a critical keyword "help" from audio inputs. This system is unique and able to recognize the keyword across four languages: English, Arabic, Kurdish, and Malay. The backbone of this project using

* Corresponding author.danyar.nabaz@gmail.com

Arduino Nano 33 BLE sense board and Edge Impulse, which involved data collection and model training.

CKVR typically involves the generation of an audio signal and identification of its properties using the Mel Frequency Cepstral Coefficients (MFCC) technique. The algorithm categorizes various keywords to train the neural network and is subsequently converted into a file deployable on an Arduino device, specifically the Arduino Nano 33 BLE sense in this project. Traditional approaches to CKVR often required significant processing power, making use of cloud servers for necessary hardware and advanced feature extraction techniques. However, these methods tend to result in high costs and inconsistent recognition rates due to ambiguous feature extraction and substantial processing board sizes.

This project addresses these challenges using a deep learning-based model trained on curated datasets, optimized for accurate and efficient feature extraction. A variety of techniques are used to interpret human speech, including frequency-time interpretations like spectrograms, power spectral densities, frequency-energy interpretations, cepstral analysis, and linear predictive coding. In addition, the use of Leaky RELU activation function provides the highest accuracy in the model.

In the pursuit of leveraging technology to increase safety in public spaces, the engagement of the development of a Keyword Voice Recognition system is focused. The primary motivation of the project is to create an accessible, practical solution that is attuned to the urgency of emergency situations. The target environments for system's deployment are heavily populated public areas such as malls, hospitals, schools, and stations, where quick response to individuals in distress is often critical.

Designing the model for this recognition system focuses on the substantial knowledge of extracting features from human pitches. The discussion on the development phase led to an understanding of the importance of incorporating various activation functions during the model's training. By doing so, the aim of the project is to strike a balance between responsiveness and accuracy, resulting in an optimized model that can correctly interpret the urgency in the voice of a person in danger.

Once operational, the recognition model use the embedded microphone in the microcontroller to capture the user's speech. A significant aspect of ongoing discussion is the consideration of how to handle the various features extracted from the captured speech. Depending on these features, the system initiates different operations. The challenge lies in ensuring these operations correlate accurately with the urgency and the context of the situation.

This discourse is pushing towards creating a solution that doesn't just act as a reactionary system, but rather one that proactively contributes to a safer environment. The path towards the realization of this project is filled with

fascinating technical challenges and look forward to addressing them and continuing this important conversation.

2. Literature Review

The development of Custom Keyword Voice Recognition system, relying on a real-time dataset, progresses through five essential stages. The crux of the project is data collection and feature extraction from the acquired data. Unlike conventional methodologies that utilize readily available data for model training, the approach is a tad more hands-on. The engagement in rigorous data collection from a multitude of individuals with varied voice pitches. The features thus extracted from the collected voice data paved the way for the creation of a unique dataset for model training.

The work of Ramachandra A.C, Raghavendra Prasanna, and Prem Chowdary Kakarla [1] on "Implementation Of Tiny Machine Learning Models On Arduino 33 BLE For Gesture And Speech Recognition" stands out. The proposed model leveraged extracted features for training and created a required Neural Network model embedded into the Arduino Nano 33 BLE sense. Notably, Edge Impulse was employed as the development platform for model training and testing, in addition to data collection and pre-processing tools. In a similar vein, the work of Ahmad Dziaul Islam Abdul Kadir, Ahmed Al-Haiqi, and Norashidah Md Din [2] on "Age Classification Based on Voice Commands" used a deep learning model developed via Edge Impulse. The model tested data collected from various individuals.

Insights from Vikranth Singh Tomar's [3] proposal of a TinyML and Voice Recognition model, which articulated the extraction of features from the voice dataset using the MFCC technique, a key input for training the deep learning model. The work of Sam Myer and Vikrant Tomar [4] on "Speech Recognition on Low Power Devices" provided guidance on building a neural network with respect to microcontroller specifications. Meanwhile, a published model emphasizing that "Neural Networks and Deep Learning consist of both classical and modern models" [5] proposed the implementation of deep learning models that have been instrumental in improving the model's accuracy.

Other studies [6,7,8] focused on the application of deep learning models on low-power devices. The Neural network is trained using collected data via Edge Impulse, which auto-extracts features to train the model. Rusci, M, Capotondi, and Benini [9] suggested "quantized nns as the definitive solution for the interface on low-power arm mcus" which informed the specifications of microcontroller and facilitated model development.
Works by Han, Song, Xingyu Liu, Huizi Mao, Jing Pu [10], and Teerapittayanon, Surat, Bradley McDanel Lan [11] provided valuable insight into constructing datasets for training models according to the processing engine. The connection of microcontrollers to WiFi and the cloud for data transmission was a crucial aspect covered in other references [12,13,14].

In subsequent sections, the software tools used to develop Deep Learning Model. This will be followed by the display of Experimental Results, reflecting the Accuracy and Loss of the Training and Testing Data, and a discussion on the Confusion Matrix used in both datasets. The final section encapsulate the conclusion, and reference contributions to the project.

3. Design Approach and Methodology

The project unfolds across two critical phases: firstly, the development of neural networks utilizing a custom keyword dataset via the Edge Impulse platform; and secondly, the implementation of these neural networks into the Arduino Nano 33 BLE Sense environment.

A. Deep learning using TinyMl:

TinyML is characterized as an operational neural network model designed to function on less than 1 mW power. This results in a compact, portable device capable of supporting any application that necessitates intricate machine learning algorithms [15] Warden. Machine learning operates by forming a system model based on the input data from the programmer, a phase known as 'training.' Once the training phase concludes, the predictive data is fed into the model, a process referred to as 'inference' [15]. TensorFlow Lite was conceived to cater to projects intended for smaller-scale execution. It's engineered to run on just a few kilobytes of space, thus, the libraries don't rely on any operating systems, C or C++ dependencies, floating-point hardware, or dynamically allocatable memory [15]. The deep learning algorithms feasible for TinyML implementation encompass Convolutional Neural Networks (CNN), Support Vector Machines (SVM), or a hybrid approach known as Transfer Learning [16]. As demonstrated in [16], different emotions were classified using a deep neural network grounded on Raspberry Pi. The utilized machine learning API was Keras, employing TensorFlow for model definition and training. The model, a Deep Neural Network (DNN), comprises multiple CNN layers, Pooling layers, and Fully Connected (FC) layers. The cascading of convolution layers enables the model to learn an array of features while keeping the filter size minimal. Figure 1 is a Block diagram of flow of implementation of TinyML.

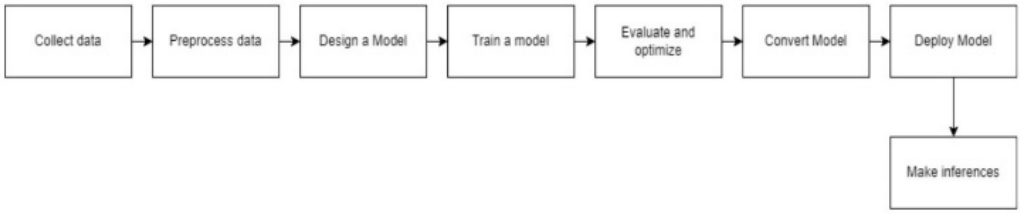

Figure 1 Block diagram of flow of implementation of TinyML.

1) Data Collection: This is the initial stage in the implementation of a TinyML model. It involves gathering a substantial amount of relevant data that the model can learn from. This data can be collected from various sources and in various forms such as images, audio files, text, or numerical data. The data collected should be diverse and representative of the real-world scenarios that the model is expected to encounter.

2) Data Processing: After collecting the data, the next step involves processing the data to prepare it for the model. This can involve various sub-steps including cleaning the data to remove any irrelevant or corrupt data, normalizing or standardizing data, handling missing data, and data augmentation. The goal here is to make the data more suitable for model training by converting it into a form that the model can understand and learn from.

3) Model Design: This stage involves selecting an appropriate machine learning model based on the problem at hand. This could be a neural network, decision tree, or other types of models. In TinyML, this typically involves designing a neural network architecture which is compact yet sufficiently capable to handle the complexity of the task at hand.

4) Model Training: Once the model has been designed, the next step is to train the model using the processed data. Training involves presenting the model with the input data and allowing it to make predictions. The model's predictions are then compared to the actual outcomes, and the model adjusts its internal parameters to improve its predictions. This process is repeated multiple times to minimize the prediction error.

5) Model Evaluation and Optimization: After training, the model's performance is evaluated using a separate validation dataset that the model has not previously seen. This gives an unbiased estimate of the model's performance. If the model's performance is not satisfactory, hyperparameters are tweaked or changes are made in the architecture of the model to optimize its performance.

6) Model Conversion: Once the model is optimized and satisfactory performance is achieved, the model is converted into a format that can be implemented on the target hardware, usually a microcontroller for TinyML applications. This often involves quantization or other techniques to reduce the size and complexity of the model.

7) Model Deployment: The converted model is then deployed onto the target hardware. This involves programming the hardware with the model and any necessary inference code. The model is now ready to make predictions in the field.

8) Making Inferences: This is the final stage where the deployed model is now used to make predictions or inferences on new, real-world data. The model processes the input data, making inferences based on its training, and produces output that can be used to make decisions or further processes.

B. Arduino nano BLE sense

The hardware component employed in this development project is the Arduino Nano 33 BLE Sense, as illustrated in Figure 2. This advanced platform is designed for the utilization of cutting-edge AI models. Under its hood, it houses a 32-bit ARM Cortex-M4F microcontroller operating at 64MHz, equipped with 1MB of system memory and 256KB RAM. This compact controller delivers sufficient capability for the deployment of Tiny ML models.

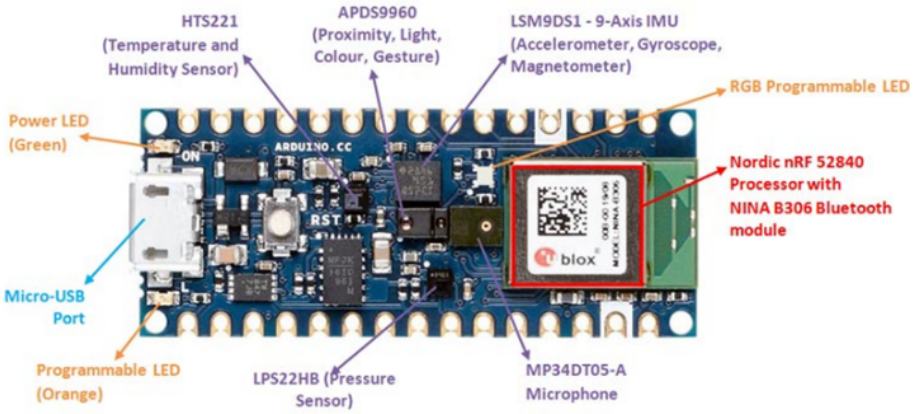

Figure 2 Arduino Nano 33- BLE

Moreover, the Arduino Nano 33 BLE Sense is integrated with a plethora of sensors that measure variables such as temperature, brightness, proximity, touch, motion, and vibration. The comprehensive sensor suite embedded within this device makes it suitable for a wide array of applications.

C. Edge Impulse

Edge Impulse is an integrated development platform specifically designed for developing machine learning models for edge devices. It offers a comprehensive set of tools and features that facilitate the entire development workflow, from data collection and preprocessing to model training and deployment. Edge Impulse provides a user-friendly interface where it can import and preprocess audio data, choose and configure signal processing blocks such as MFCC, and select learning blocks like Transfer Learning or Classification. It allows to train and optimize the model using curated datasets, evaluate its performance, and convert it into a format

suitable for deployment on the Arduino Nano BLE Sense board. Edge Impulse greatly simplifies the process of developing machine learning models for edge devices, providing a seamless and efficient experience for creating robust and accurate keyword spotting systems on the Arduino Nano BLE Sense.

D. Phases of the project

These phases guide the development process of the keyword spotting project, leveraging Edge Impulse and Arduino Nano BLE Sense to create an efficient and accurate system for real-time keyword recognition. Figure 3 shows the project flow chart based on the steps and phases.

a) Data Collection: Utilizing Edge Impulse's built-in recorder connected to Arduino Nano BLE 33, the sound is captured as 15-second audio samples, after that the sound is segmented.

b) Data Segmentation: Each 15-second audio sample is split into smaller segments. Segmentation helps improve the efficiency of processing and allows for a more fine-grained analysis of the audio data.

c) Impulse Creation and Configuration: to create an impulse in Edge Impulse, defining the processing block, learning block, and feature extraction settings. Here, set the window size and window increase, window Size: this parameter defines the length of each audio segment that will be processed. In other words, it determines how much audio data is considered at once. The window size is typically measured in milliseconds. Window Increase: This parameter, also known as hop size or stride, determines the overlap between consecutive windows. It defines how much the window should move forward in the data for the next segment. For example, if a window size of 1000 ms and the window increase of 200 ms, it means that every second window starts 500 ms (half a second) after the first one. This means there is a 20% overlap between consecutive windows. Processing Block specifies the chosen audio processing technique (e.g., MFCC, MFE). Learning Block determines the learning algorithm or approach (e.g., Transfer Learning or Classification) and feature Extraction extracts relevant features from the audio segments for training and analysis.

d) Model Training: The model is trained using the curated dataset, and the training results are obtained. Training involves optimizing the model parameters using the labeled audio segments to learn and identify keyword patterns.

e) Model Testing: The trained model is tested using a separate dataset to evaluate its performance and accuracy. Testing provides insights into the model's ability to correctly identify keywords in unseen audio samples.

f) Project Deployment: The project is deployed as an Arduino library, allowing for easy integration and use with Arduino Nano BLE Sense. The deployment ensures that the developed model and associated code are readily accessible and compatible with the target hardware. It's worth mentioning that optimization can be achieved through either quantized or unoptimized approaches. Quantization refers to the process of reducing the precision of numerical values, such as converting floating-point numbers to fixed-point numbers with a limited range. On the other hand, unoptimized methods involve performing optimization without any specific adjustments or enhancements. The reason for choosing quantization is its ability to provide 100% accuracy after thorough testing. By reducing the precision of numerical values, quantization enables efficient storage and computation, resulting in optimized performance. This technique is particularly useful in scenarios where computational resources are limited, as it allows for faster and more efficient execution.

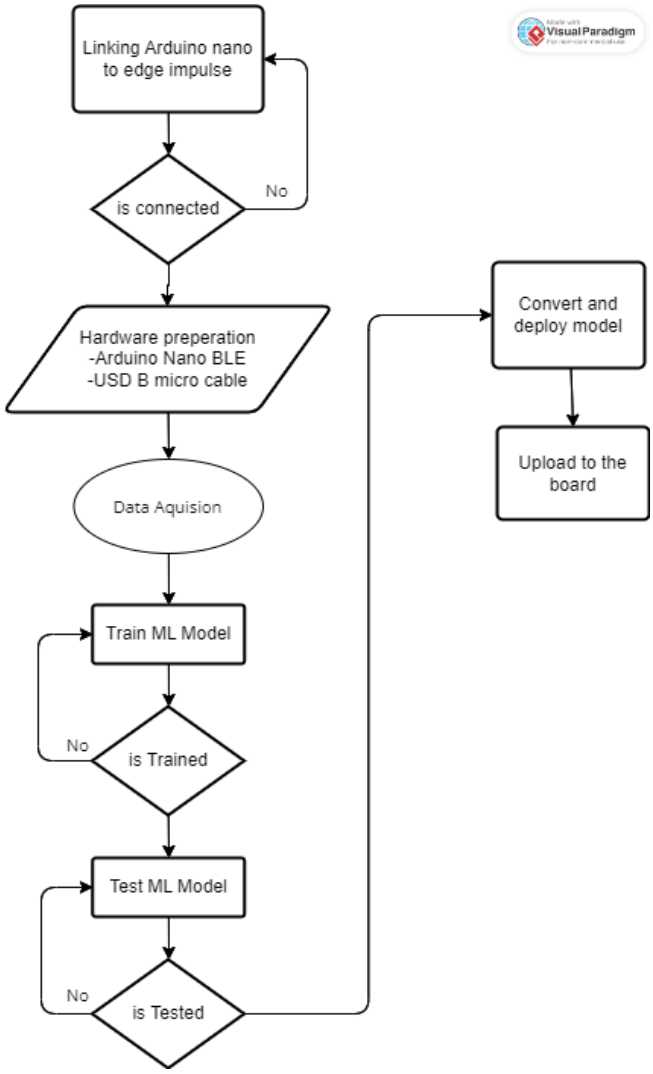

Figure 3 shows the project flow chart.

4. Results and Discussion.

After successfully collecting data, the extensive tests using different processing blocks and learning blocks to determine the optimal configuration for machine learning project. The results of these tests are presented in Table 1, which summarizes the performance metrics of each configuration.

Table 1 Performance Comparison of Different Processing and Learning Blocks

Processing Block	Learning Block	Training Accuracy	Loss	Test Accuracy	Created features
Audio (MFE)	Transfer learning (Keyword Spotting)	%100	0.40	%83.33	5710
Audio (MFE)	Classification	%88.8	0.23	%100	5160
Audio (MFCC)	Classification	%93.8	0.21	%90.1	845

The table provides a summary of the training accuracy, loss, test accuracy, and the number of created features for each combination of processing and learning blocks. The first configuration, utilizing the Audio (MFE) processing block with Transfer Learning (Keyword Spotting) as the learning block, achieved a training accuracy of 100%, a loss of 0.40, and a test accuracy of 83.33%, with 5710 features created. The second configuration, using Audio (MFE) with Classification as the learning block, achieved a training accuracy of 88.8%, a loss of 0.23, and a perfect test accuracy of 100%, with 5160 features created. Lastly, the third configuration, which shows the best results out of them, employing Audio (MFCC) with Classification as the learning block, achieved a training accuracy of 93.8%, a loss of 0.21, and a test accuracy of 90.1%, with 845 features created.

Table 2. Impact of Training Cycles and Learning Rate on Model Accuracy

Number of training cycles	**Learning rate**	**Model Training Accuracy**	**Loss**	**Model Testing Accuracy**
100	0.005	%80	0.41	%84
120	0.005	%94.7	0.14	%91.7
150	0.006	%92.4	0.18	%90
200	0.008	%100	0.00	%100
220	0.009	%94.33	0.12	%93.33
250	0.009	%100	0.06	%83.33

The table illustrates the influence of different training cycles and learning rates on the model's performance. As the number of training cycles increased from 100 to 250, its observed varying levels of accuracy and loss. Notably, the best overall accuracy was achieved with 200 training cycles and a learning rate of 0.008,

resulting in 100% accuracy in both model training and testing. Figures 4 and 5 show the model training and testing results.

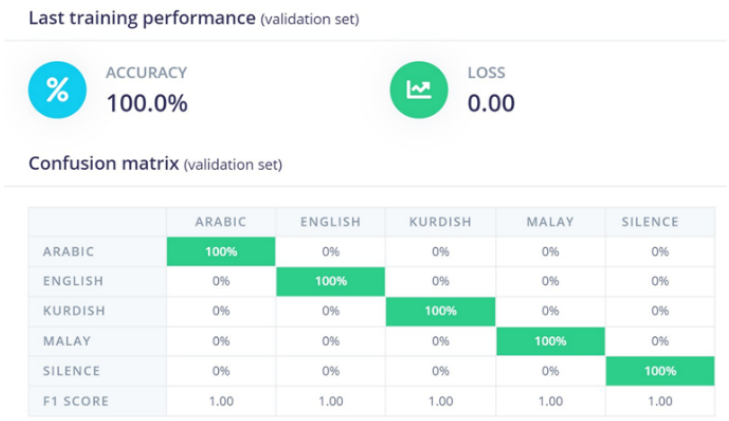

Figure 1. Model Training Results

These results highlight the importance of optimizing the training cycles and learning rate to maximize the model's accuracy. Through iterative adjustments, the identification for ideal combination that yielded the highest level of accuracy for keyword spotting project.

Figure 2. Model Testing Results

4. Conclusion

The successfully developed a unique keyword spotting system that excels at identifying the critical keyword "help" in multiple languages, including English, Arabic, Kurdish, and Malay. This system harnesses the power of the Arduino Nano 33 BLE Sense board and leverages the capabilities of Edge Impulse for data processing and model training. In pursuit of the most effective approach, the combined the Mel Frequency Cepstral Coefficients (MFCC) processing block with the Classification learning block. This combination proved to be the optimal choice, resulting in outstanding performance. Furthermore, the employment of Quantized (int8) optimization techniques to ensure efficient utilization of resources without compromising accuracy. Through rigorous training and testing, the model exhibited

an exceptional accuracy rate of 100% in both the training and testing phases. This high level of accuracy is a testament to the effectiveness of the chosen approach and the robustness of the keyword spotting system. By accurately identifying the keyword "help" across multiple languages, the system is well-equipped to support emergency response scenarios and improve safety in various environments. Whether it is in bustling public spaces, hospitals, schools, or other critical areas, the keyword spotting system stands ready to provide immediate assistance. The successful development of this system highlights the effectiveness of the MFCC processing block, Classification learning block, and the careful optimization of the model. These achievements pave the way for further advancements in the field of speech recognition and demonstrate the potential for creating innovative solutions that contribute to a safer and more responsive world.

Acknowledgments

We like to generously thanks Universiti Teknologi Malaysia (UTM), and the Ministry of Higher Education Malaysia for all the supports to complete this research.

References

[1] Viswanatha V, Ramachandra A.C, Raghavendra Prasanna, Prem Chowdary Kakarla, Viveka Simha PJ, Nishant Mohan. Implementation Of Tiny Machine Learning Models On Arduino 33 BLE For Gesture And Speech Recognition.

[2] Ahmad Dziaul Islam Abdul Kadir; Ahmed Al-Haiqi; Norashidah Md Din. A Dataset and TinyML Model for Coarse Age Classification Based on Voice Commands, IEEE 2021.

[3] Vikranth Singh Tomar. TinyML and Voice Recognition Technology .

[4] Sam Myer and Vikrant Tomar. Speech Recognition on low power devices.

[5] Charu C.Aggarwal(2018). Neural Networks and deep learning consists both classical and modern models.

[6] Ian McGraw,Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, “Personalize d speech recognition on mobile devices. In Acoustics, Speech and Signal Processing”, IEEE 2016.

[7] Dania Maryam Waqar; Teddy Surya Gunawan; Malik Arman Morshidi; Mira Kartiwi. Design of a Speech Anger Recognition System on Arduino Nano 33 BLE Sense(IEEE-2021).

[8] Pete Warden and Daniel Situnayake. Machine Learning with tensorflow on Arduino and Ultra-Low-power Microcontrollers.

[9] Rusci, M., Capotondi,Benini, “quantized nns as the definitive solution for interface on low-power arm mcus”,IEEE 2018.

[10] Han, Song, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Perdram, Mark A. Horowiz,”EIE : efficient interence engine on compressed deep neural network.”

[11] Teerapittayanon, Surat, Bardley McDanel Lan, “Distributed deep neural networks over the cloud, the edge and end devices”.

[12] Manasa, T., Kadali, J., Syed, N., Raju, G. K., & Jamal, K. (2022, November). IoT based Coal Mine Safety Monitoring and Warning System. In 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 11-15). IEEE.

[13] Sudhakar Yadav, N., Ravikanth Motupalli, N., Jamal, K., & Usha Rani, Y. (2022). Predictive and Behavioral Analytics for Big Data Architecture. In Recent Trends in Product Design and Intelligent Manufacturing Systems: Select Proceedings of IPDIMS 2021 (pp. 595-605). Singapore: Springer Nature Singapore.

[14] Sumanth, P., Samiuddin, S., Jamal, K., Domakonda, S., & Shivani, P. (2022, April). Toxic Speech Classification using Machine Learning Algorithms. In 2022 International Conference on Electronic Systems and Intelligent Computing (ICESIC) (pp. 257-263). IEEE.

[15] P. Warden, D. Situnayake, "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers," O'Reilly Media, 2019.

[16] V. Srinivasan, S. Meudt, F. Schwenker, "Deep learning algorithms for emotion recognition on low-power single-board computers," In IAPR Workshop on Multimodal Pattern Recognition of Social Signals in Human-Computer Interaction, pp. 59-70, 2018.