

Voice Keyword Spotting on Edge Devices

Khalid Saifullah

Department of Computer Engineering,
Zakir Husain College of Engineering &
Technology

Aligarh Muslim University
Aligarh, India

khalidsaif9045@gmail.com

Raunaque Mujeeb Quaiser

Group Manager, ST Microelectronics
India

Noida, India

raunaque.quaizer@st.com

Nadeem Akhtar

Department of Computer Engineering,
Zakir Husain College of Engineering &
Technology

Aligarh Muslim University
Aligarh, India

nadeemakhtar@zhecet.ac.in

Abstract— One of the most challenging subjects in edge device voice recognition is the difficulty of detecting a given set of words in fluent speech. Convolutional Neural Networks (CNN) are proven to be extremely successful for cognitive tasks and are simple to deploy on edge devices. In this article, a Convolutional Neural Network (CNN) model for Keyword Spotting (KWS) is trained and deployed. Unwanted components are eliminated from audio recordings, and voice characteristics are retrieved using the Mel-Frequency Cepstral Coefficient (MFCC). The model is trained with 84 thousand parameters. The model uses 10.34 Kilobyte of RAM and Flash memory of 122.32 Kilobyte. The model is successfully deployed on STM32 Nucleo G474RE Board. The proposed work outperforms the existing works.

Keywords—CNN, KWS, Voice Recognition, MFCC

I. INTRODUCTION

The ubiquity of smart devices and equipment that perform intelligent operations by autonomously perceiving, processing, and transmitting information through the Internet characterizes the next generation of information and communication technology. The Internet of Things (IoT) is meant to give mankind a simpler, safer, and more intelligent lifestyle, ranging from e-health, smart homes, and intelligent transportation to industrial production and supply chains. However, the fast proliferation of IoT applications has resulted in an unprecedented number of linked "Things." [1] Internet-of-things (IoT) platforms for visual data analytics have applications in a variety of fields, including residential, commercial, and military surveillance, robots, and autonomous vehicles [2]. It connects many sensors and devices in cloud data centers. However, dealing with this massive amount of data is soon becoming a severe challenge for the clouds. It must provide tremendous computational power, which is a major concern. Furthermore, rising data transmission demand is pressing up against the world's maximum capacity. Devices that need data at short interval of time like keyword spotting (KWS), automatic speech recognition (ASR) employing audio data, and object detection with a large number of image sensor data come to mind. This data may contain sensitive personal information, and sending private data to the cloud poses serious issues on privacy. Edge computing proves to be a fantastic answer to the stated problem. In its place of transmitting data to the cloud, it processes it locally, solving challenges like bandwidth cost, privacy, and scalability [3][4].

Keyword spotting on edge devices has already shown to be an efficient method of engaging with electronic devices. These devices frequently receive speech data as input and transmit it to the cloud to identify the keyword; the cloud analyses the data and sends the result back to the device, and the device acts upon receiving the response. Steady connectivity between the cloud servers and the device not

possible due to the unpredictability of cellular networks. To address the issue of communication failure, edge devices containing microcontrollers were created. Instead of transmitting data to the cloud, these interfaces take control of the input process and execute the desired action.

In challenging applications like these, Convolutional Neural Networks (CNN) provide extraordinarily high accuracy. Because the Keyword Spotting application is primarily concerned with data sets, a good response is received with recurrent neural networks (RNN). However, these network needs a significant quantity of computing and storage. An RNN neuron has eight times the weight and complexity of a normal Convolutional Neural Network (CNN) [5].

Tensorflow lite is used to reduce computational complexity and memory footprint. TensorFlow Lite is free to use and multiple platforms supported deep learning framework that provides optimized results for performance and storage by converting a pre-trained TensorFlow model to a new format. Google Speech Command dataset version 2 is used to train the proposed model [6]. Extracting feature from an audio stream is also a computationally demanding task. Fewer filters will be employed to build the energy spectrum to reduce computation expenses. Feature extraction consists of complex mathematical techniques like matrix multiplication, Discrete Cosine Transform (DCT), Fast Fourier Transform (FFT) and many others. To write these computations in low-level C language is also a challenging task.

II. RELATED WORK

In [7], Ruchi et al. successfully investigated and implemented a cross-correlation-based speech recognition system, Pitch for two distinct persons, with MFCCs utilized as audio characteristics. For the audio extraction method, they employ MATLAB. They also investigated fluctuations in pitch and MFCCs, concluding that these characteristics can be utilized for the improvement of accuracy in speech recognition systems.

[8] Saleh et al. investigate wheelchair control using CNN-LSTM as a control mechanism given by their combination. Wheelchair commands are divided into following categories: forward, reverse, right, left, and stop. Two CNN and CNN-LSTM tests were performed to assess the performance of the proposed technique. Voice instructions applied to wheelchairs were developed to deliver excellent performance and accuracy. Certain voice commands, however, encountered label recognition difficulties. For voice instructions such as forward, backward, left, right, and stop, there are no notable changes between CNN and CNN-LSTM.

Md Naim Miah et al. construct, train, and implement a Deep Neural Network-based model on two separate edge devices [9]. They tested their algorithm on six keywords and found it to be 91% accurate. Their model makes use of 11.52 KB of RAM and 169.63 KB of flash memory.

In [10], Robert et al. created a smart speaker for the German language. Due to the lack of a training dataset, they did not achieve a high recognition rate. More than 3000 hours of training data are required for English instructions; however, Mozilla Common Voice only provides 483 hours of training data; even after adding more data through data augmentation, the training data is still insufficient for English instructions.

The authors of [11][12] did substantial research on a variety of neural networks, including CNN, LSTM, RNN, and others. They assessed the memory footprint, computational complexity, and accuracy of all networks. They ran the model on a kit using a 32-bit ARM CPU. They came to the conclusion that the CNN model and Depth wise Separable CNN are the best.

A. G. Howard et al. in [13] found that for applications like photo categorization which requires classification of 2D spatial data, MobileNet model is very effective. They demonstrated a considerable reduction in computational needs, making it a great choice for edge devices.

Rahaman et al. investigated the performance of isolated speech recognition systems using MFCC and cross-correlation in [14]. They displayed many recognition approaches as well as recognition accuracy. Out of four separate recognition methodologies, with the help of FFT they achieved 92% recognition accuracy. Their research will contribute to the progress of speech recognition technologies.

Chiu et al. [15] propose employing sequence-to-sequence models for voice recognition. Their strategy addresses voice recognition improved performance as a result of working on a 12500-hour voice search assignment. Compared to the normal WER of 4%, their technique yielded a WER of 4.1 percent.

Afouras et al. [16] present a method for audio-visual speech recognition. They attempted to identify speech by analyzing facial expressions while listening to audio or not. They used lip movement to distinguish sounds, especially in environment with lots of noise. They worked on two data sets which is available on TED, British television, YouTube, and TEDx.

Toshniwal et al. [17] created a Single edge-to-edge Multilingual Speech Recognition Model. They included a variety of languages in their model, and it improved by 21% when compared to earlier models.

III. PROPOSED WORK

A. Audio Feature Extraction

The initial or fundamental stage in every speech or voice recognition program is to process audio extract characteristics

from the audio. It just simply picks the relevant information from the audio and discards the rest. Before training, raw data is subjected to a series of augmentations, including Shift, Pad zeros, Amplify, Change play speed, and Superpose noise.

The prominent feature extraction method for speech recognition applications is Mel Frequency Cepstral Coefficient (MFCC) [18]. A 2D feature matrix that is equivalent to a grayscale image is generated by MFCC, making it an excellent choice for use in keyword detection system based on Neural Network. It is just like human hearing system. When the frequencies are low, about 1kHz, humans are adept at discerning minor fluctuations in pitch. However, higher frequencies are harder for humans to detect. Given this, the speech characteristics may be retrieved similarly to an ear of human being, as proposed in [18], [19].

Step 1: Raw audios in the Speech Commands Dataset [6] are all around one second long and contain very minimal background noise. It can easily lead to overfitting. Data augmentation techniques are required to prepare for a wide range of real-world circumstances, including noise, fluctuation in audio duration, and loudness. The raw audio is processed with some augmentations like amplify, change play speed and superpose noise.

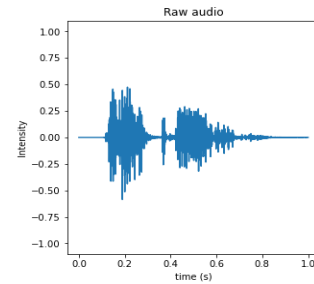


Fig. 1 Raw audio sample for keyword “backward”

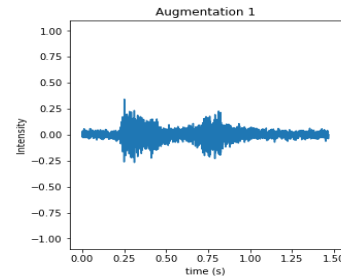


Fig. 2 Voice sample after voice data augmentation

Step 2: An audio signal is continually changing, we presume that it does not change considerably over short periods, so we divide the audio signal into frames of 20ms at 8kHz. Number of samples = $0.020 * 8000 = 160$ samples

Step 3: Calculating Fast Fourier Transform (FFT) of each frame. A signal is deconstructed into two units: imaginary and real units by Fast Fourier Transform (FFT). FFT can be used for the analysis of frequency, resulting in simplifying voice processing as it is related to the human hearing process. The discrete Fourier transform (DFT) is implemented in the FFT algorithm. DFT is an N-sample transformation of each frame. Transitioning from the time domain to the frequency domain. The DFT is calculated using the Equation 1.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n / N} \quad (1)$$

For $0 \leq k \leq N-1$, where, $x(n)$ represents the signal frame in time domain, $X(k)$ is the frequency domain signal, and n is the number of particular frames.

Step 4: The FFT signal those results is bundled into a triangle filter known as Mel-spaced Filter. The grouping here serves the purpose of multiplying each FFT value by the associated filter gain and summing the result. Then, for each group, a particular quantity of signal energy weight is assigned.

Step 5: A power spectrum is generated by Mel filter bank. The sensitivity of humans to small energy changes at high energy levels is lower than their sensitivity to large energy changes at low energy levels. It is actually logarithmic. The log will thus be extracted from the Mel filter bank output in this stage. This also lowers acoustic variances that are unimportant for voice recognition.

Step 6: The final stage in the fundamental process of MFCC feature extraction is the Discrete Cosine Transform (DCT). The essential idea behind DCT is to correlate the values of the mel spectrum in order to generate a fair representation of the property spectral local.

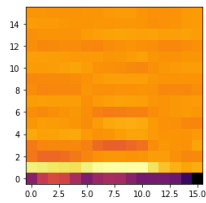


Fig. 3 Mel Frequency Cepstral Coefficient (MFCC) representation for keyword “backward”

B. Convolutional Neural Network Model

A convolutional neural network (CNN) is divided into two sections: convolutional and classification. The convolutional portion includes one or more layers of neural networks that conduct the "convolution" operation on the input picture, where "convolution" refers to filtering the image to find features. An image filter is a window that sweeps through the whole picture, collecting a sample of pixels (2x2 pixels in the first layer), multiplying them by certain weights, and then summing the results to form a pixel on the next image. This is referred to as a kernel window.

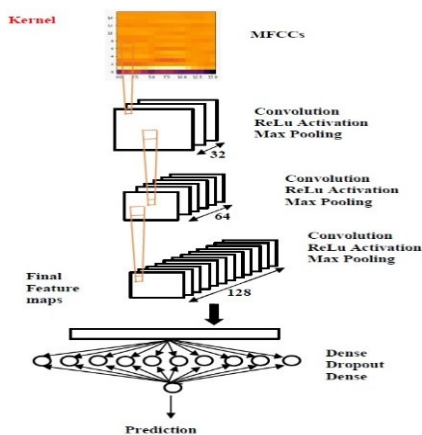


Fig. 4 Proposed CNN_VKS Model

The first convolutional layer has 32 nodes, resulting in 32 smaller pictures after the first round of calculations. These

smaller pictures are filtered outputs of the input, and they frequently emphasise minute elements of the original image, such as borders and corners. Following convolution, we apply the rectified linear unit (ReLU) activation function, which simply turns all negative values to 0. This provides a degree of nonlinearity to the neural network, allowing it to be trained for classification and feature extraction.

The Max Pooling procedure is then done, in which another window glides over each filtered picture and drops everything but the highest pixel value in that window. This helps to maintain the most significant characteristics and information about their approximate position. It also helps to reduce the computational load on the subsequent layers. The Convolution-ReLU-Max Pooling method is done two more times to obtain a final set of 64 and 128 feature maps. With each subsequent convolutional layer, we train the network to learn increasingly complicated information.

These picture characteristics are then flattened into a single long vector (rather than a collection of 2D matrices) and input into a fully-connected neural network for classification. A 0.25 dropout layer is added, which implies that 25% of the previous layer's node outputs are randomly discarded while entering the next layer. This adds some unpredictability to the network, which can assist with overfitting.

IV. HARDWARE AND SOFTWARE USED

A. STM32CubeMX code genrator

STM32CubeMX is a software utility that allows to setup numerous STM32 microprocessors and microcontrollers. It produces low-level C code for ARM-Cortex CPUs. This utility has an easy-to-use interface that allows to select a microprocessor and then adjust its GPIO or clock. This utility may also be used to manage peripherals [20].

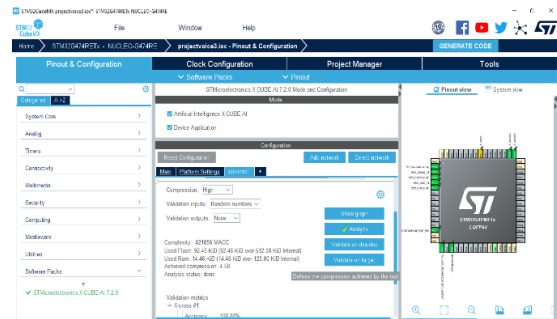


Fig. 5 STM32CubeMX Code Generator Interface

B. X-Cube-AI Expansion Pack

X-CUBE-AI is an STM32Cube expansion pack. This upgrade enables the STM32 Boards to implement AI algorithms like as neural networks and traditional machine learning techniques. This enhancement allows the methods to be analysed on the desktop and target, as well as the simulation results to be reviewed [21].

C. STM32 Nucleo G474RE Board

The STM32 Nucleo-64 board offers an economical and flexible approach for users to experiment with new concepts and construct prototypes by combining the STM32 microcontroller's performance and power consumption advantages. This board is powered by a 32-bit ARM cortex M-4 processor. It has 512 kilobytes of Flash memory and 128 kilobytes of SRAM [22].



Fig. 6 STM32 Nucleo G474RE Board

V. EXPERIMENTAL RESULTS

This section includes experimental results. The model is compiled utilizing the Adam optimizer and a batch size of 32, with 80% of the training dataset used for training and 20% utilized for validation. The model is trained for 40 epochs. Accuracy, Flash needed and Complexity are calculated for different keywords.

A. Training and Validation Accuracy

The accuracy gained when the model is applied to training data is referred to as training accuracy. Validation accuracy is determined by applying the model to a few randomly selected data points from a random class. Fig. 6 depicts a graph of the model's training and validation accuracy.

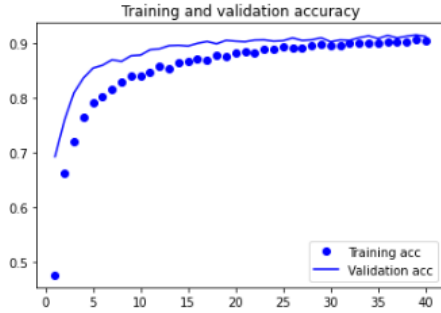


Fig. 6 Training and validation accuracy graph

B. Training and Validation Loss

The inaccuracy on the training data-set is referred to as training loss. The error that happens after passing the validation data-set through the trained network is referred to as validation loss. Fig. 7 depicts a graph containing the model's training and validation losses.

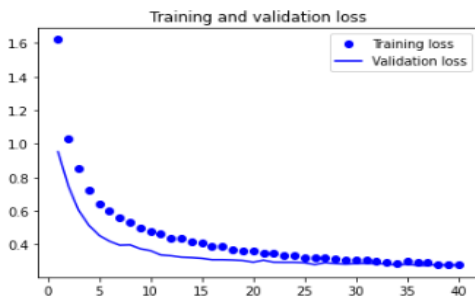


Fig. 7 Training and validation losses

C. Training Model Outputs

Fig. 8 depicts the model's performance using a confusion matrix. With the help of test data confusion matrix is generated. The predicted outputs are represented on x-axis and actual output represented on y-axis. The majority of the keywords are identified correctly.

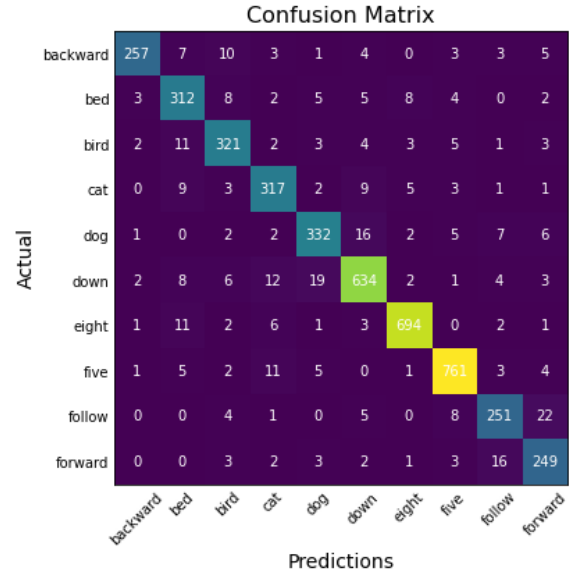


Fig. 8 Confusion matrix

The computed precision, recall, accuracy, and f1-score are shown in table 1. The average accuracy is 0.912, the average recall is 0.906, and the average f1-score is 0.909, indicating that model performance is satisfactory.

TABLE I: Precision, Recall, Accuracy and F1-score

Metric	Value
Precision	0.912
Recall	0.906
Accuracy	0.920
F1-score	0.909

D. Accuracy, Flash Usage and Complexity

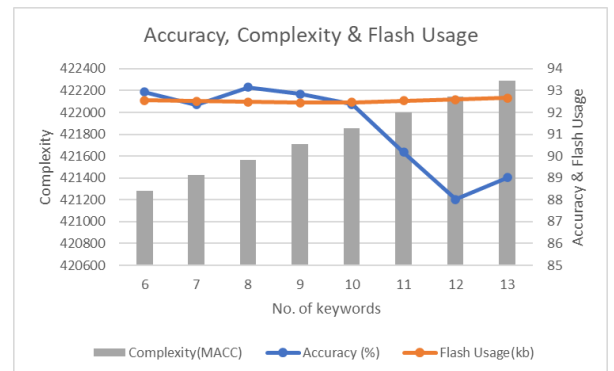


Fig. 9 Accuracy, Flash Usage and Complexity graph for different number of keywords

According to the aforementioned findings, the model's accuracy is 93% for less than 11 keywords. The accuracy falls as the number of keywords increases. The flash usage is strongly related to the number of keywords; as the number of keywords grows, flash usage also grows. Complexity is also dependent on keywords, increase in keywords leads to increase in complexity.

E. Result Comparison

TABLE II: Results Comparison

Labels	M. N. Miah [9] (7 Keywords)	Proposed Work (7 Keywords)	Proposed Work (10 Keywords)
No. of layers in model	27	13	13
No. of trainable params	721,000	84,246	84,246
Accuracy	91%	92.36%	93%
Flash Usage (kb)	169.93	118.26	122.32
Ram Usage (kb)	11.52	10.32	10.34
Execution Time (ms)	6.941	6.071	6.259
Complexity (MACC)	287,763	421,424	421,856
Kit Used	STM32F769NI	STM32 Nucleo G474RE	STM32 Nucleo G474RE

According to Table II, the suggested work exceeds the prior work, the accuracy rose by 1.36% with 7 keywords and 2% with 10 keywords, and the proposed model requires less flash memory. The model also has fewer layers and trainable parameters than earlier studies. The kit utilized in the suggested model is significantly less expensive than the present works kit.

CONCLUSION

In this study, a CNN_VKS model is proposed, developed, and deployed on the STM32 Nucleo G474RE board, and the results are discussed. Voice data augmentation and MFCC are used for feature extraction. To make the model more noise resistant, several sorts of noise such as air conditioner, fan, and ambient noise are superimposed on the speech signal. A CNN_VKS model is trained and deployed on the board. The trained model is 93% accurate. The model uses 122.32Kb of Flash and 10.34Kb of RAM. The trained model is put to the test in the real world and produces astounding results. Using this technique, devices with limited processing resources will be able to reliably detect spoken words for a wide range of small-scale applications with a substantial margin of error without the need of trained models and complicated learning algorithms.

REFERENCES

- [1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," in IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015, doi: 10.1109/COMST.2015.2444095.
- [2] H. Liu, S. Chen and N. Kubota, "Intelligent Video Systems and Analytics: A Survey," in IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1222-1233, Aug. 2013, doi: 10.1109/TII.2013.2255616.
- [3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.
- [4] B. A. Mudassar, J. Hwan Ko and S. Mukhopadhyay, "Edge-Cloud Collaborative Processing for Intelligent Internet of Things: A Case Study on Smart Surveillance," 2018 55th ACM/ESDA/IEEE Design

- Automation Conference (DAC), 2018, pp. 1-6, doi: 10.1109/DAC.2018.8465862.
- [5] M. H. Samavatian, A. Bacha, L. Zhou, and R. Teodorescu, "Rnnfast: An accelerator for recurrent neural networks using domain-wall memory," ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 16, no. 4, pp. 1-27, 2020.
- [6] Google, Speech commands dataset version 2, 2018. [Online]. Available: http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz.
- [7] R. Gupte, S. Hawa and R. Sonkusare, "Speech Recognition Using Cross Correlation and Feature Analysis Using Mel-Frequency Cepstral Coefficients and Pitch," 2020 IEEE International Conference for Innovation in Technology (INOCON), 2020, pp. 1-5, doi: 10.1109/INOCON50539.2020.9298320.
- [8] Sutikno, K. Anam and A. Saleh, "Voice Controlled Wheelchair for Disabled Patients based on CNN and LSTM," 2020 4th International Conference on Informatics and Computational Sciences (ICICoS), 2020, pp. 1-5, doi: 10.1109/ICICoS51170.2020.9299007.
- [9] M. N. Miah and G. Wang, "Keyword Spotting with Deep Neural Network on Edge Devices," 2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC), 2022, pp. 98-102, doi: 10.1109/ICEIEC54567.2022.9835061.
- [10] R. Peinl, B. Rizk and R. Szabad, "Open Source Speech Recognition on Edge Devices," 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), 2020, pp. 441-445, doi: 10.1109/ACIT49673.2020.9208978.
- [11] Zhang, Yundong, et al. "Hello Edge: Keyword Spotting on Microcontrollers." *arXiv.Org*, arxiv.org, 20 Nov. 2017, <https://arxiv.org/abs/1711.07128>.
- [12] L. Lai and N. Suda, "Enabling Deep Learning at the LoT Edge," 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2018, pp. 1-6, doi: 10.1145/3240765.3243473.
- [13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", 2017. arXiv: 1704.04861 [cs.CV].
- [14] M. E. Rahaman, S. M. S. Alam, H. S. Mondal, A. S. Muntaseer, R. Mandal and M. Raihan, "Performance Analysis of Isolated Speech Recognition Technique Using MFCC and Cross-Correlation," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1-4, doi: 10.1109/ICCCNT45670.2019.8944534.
- [15] Chiu, Chung-Cheng, et al. "State-of-the-Art Speech Recognition With Sequence-to-Sequence Models." *arXiv.Org*, arxiv.org, 5 Dec. 2017, <https://arxiv.org/abs/1712.01769>.
- [16] T. Afouras, J. S. Chung, A. Senior, O. Vinyals and A. Zisserman, "Deep Audio-visual Speech Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2018.2889052.
- [17] S. Toshniwal et al., "Multilingual Speech Recognition with a Single End-to-End Model," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 4904-4908, doi: 10.1109/ICASSP.2018.8461972.
- [18] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 28, no. 4, pp. 357-366, August 1980, doi: 10.1109/TASSP.1980.1163420.
- [19] Wei Han, Cheong-Fat Chan, Chiu-Sing Choy and Kong-Pang Pun, "An efficient MFCC extraction method in speech recognition," 2006 IEEE International Symposium on Circuits and Systems (ISCAS), 2006, pp. 4 pp.-, doi: 10.1109/ISCAS.2006.1692543.
- [20] STM32CubeMX code generator (<https://www.st.com/en/development-tools/stm32cubemx.html>)
- [21] X-CUBE-AI Expansion Package (<https://www.st.com/en/embedded-software/x-cube-ai.html>)
- [22] STM32 Nucleo G474RE Board (<https://www.st.com/en/evaluation-tools/nucleo-g474re.html>)