Journal of Information Systems Engineering and Management
2025, 10(25s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

On-device Keyword Spotting of Odia Language on an Edge Device Using Quantization for Model Compression

Bikash Ranjan Bag[1], Manas Ranjan Patra[1, 2]
[1]*Department of Computer Science, Berhampur University, Bhanja Bihar, Berhampur, Ganjam, Odisha, India*
[2]*NIST University, Berhampur, Odisha-761008, India*
**Email: brb.cs@buodisha.edu.in, Author2*

ARTICLE INFO

Received: 18 Dec 2024

Revised: 05 Feb 2025

Accepted: 18 Feb 2025

ABSTRACT

Keyword spotting (KWS) in speech recognition refers to the task of detecting specific words or phrases by a system. KWS focuses on identifying and reacting to particular words or commands. There are various domains where KWS can be applied, such as voice assistants, security systems, automotive systems, healthcare, industrial systems, and various edge devices. In this research, a KWS system has been developed to recognize 24 frequently used words in the Odia language which includes the digits (0 to 9), seven colours and different directions. The dataset is recorded with Zoom H1n mic with 41Khz frequency. This paper proposes three models based on CNN, LSTM, and CNN+LSTM for KWS. Mel Frequency Cepstral Coefficients (MFCCs) are used as features for each keyword. The models are trained and tested with the dataset we prepared. It has been observed that the CNN model performs better than the other two models. The models are compressed using a quantization technique, resulting in a 3x reduction in model size after quantization. The accuracy of the original model (97%) is preserved after quantization. This enables such a model to be deployed on various edge devices. All the models are deployed and tested on the Raspberry Pi 3B board.

Keywords: Keyword Spotting, Odia Language, Speech Recognition, Model Compression, Quantization, Edge Computing.

INTRODUCTION

Speech recognition is the process of converting an audio signal from a human voice into text by a computer. There are various applications of speech recognition. Virtual assistants like Google Home, Alexa, Siri, etc., use speech recognition to respond to users' queries. Different electrical and electronic devices at home are controlled through voice commands in a smart home. In automotive systems, speech recognition is used for navigation, communication, and controlling entertainment systems, reducing the need for manual input by the driver. In most cases, the computation required for automatic speech recognition (ASR) takes place in the cloud, for which internet connectivity is necessary. However, in situations where there is no internet connectivity, the required computation for ASR must be done on the device itself, a process known as on-device computation. Nowadays, technology is shifting from cloud-based to on-device computation, making all necessary processes independent of the internet.

We generally operate any device with some commands, such as in a home automation system, where the command "open" will open the door and "close" will close the door of the room. Recognition of such commands by a device is called keyword spotting (KWS). In a KWS system, the recognition of a speech command takes place on the device itself rather than in the cloud, as shown in Fig. 1. Google has prepared a dataset of forty words for keyword spotting in the English language [1]. Keyword spotting helps hearing-impaired people communicate with others using voice commands. People with disabilities can operate any device or machine using voice commands. During Covid, people had to be isolated and were advised not to touch anything. In such contactless situations, any operation can be performed using voice commands. The keyword spotting method is very useful in such scenarios.

Odia is one of the classical languages of India, mostly spoken in the state of Odisha. There are KWS systems to recognize Odia words and digits [2], [3], [4] but these models are not designed for Edge devices. In this research, we

have developed deep learning models and compressed them using quantization to enable them to run on Edge devices. All our models are deployed and tested on the Raspberry Pi 3B board, which is widely used in the IoT ecosystem. This paper is organized as follows: Section 2 discusses related work, Section 3 presents the proposed methodology, Section 4 focuses on the analysis of the experimental results, and Section 5 concludes our findings.

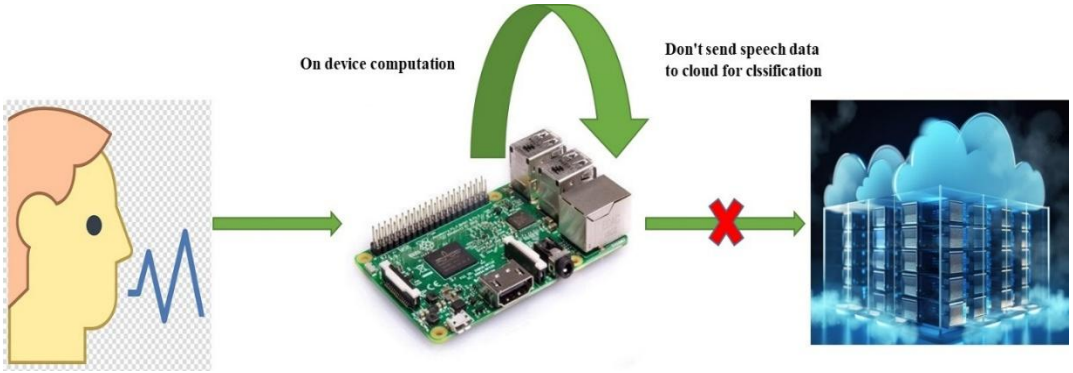

Fig.1 On-device computation for KWS

LITERATURE REVIEW

Individuals with intellectual disabilities are often eager to use information technologies, but the technologies are often not designed to align with their skill sets [5]. Keyword spotting helps people with disabilities to operate the technology-enabled machine using voice commands. A keyword-spotting algorithm has been developed by [6] using CNN. They implemented the algorithm in the Movidius Myriad 2 Vision Processing Unit. They have used 10 words from the Google speech commands dataset [1] for training, validation and testing the model by splitting 80:10:10. In this work they have improved the power consumption, throughput and latency. [7] have developed an isolated word recognition system for people with speech disordered using CNN. The model was trained and tested with an Italian dataset with 13 labels. They deployed the model on the Raspberry Pi board. They achieved a WER of 15.6%. [8] have designed a 65-nanometer CMOS with 32 KB of memory for keyword spotting. The chip is designed to support the LSTM model for training and classification. They have trained the model using the TIMIT dataset. They compressed the LSTM model using quantization where the 32-bit floating-point operation is represented by an 8-bit fixed point operation. [9] have developed an algorithm to search for the efficient architecture of the DNN model for keyword spotting. They achieved an accuracy of 97.04% with 184 thousand parameters. [10] have developed a voice user interface for speech-impaired people that recognizes the 79 Italian speech commands. They have fine-tuned the different variants (tiny, base, small) of the whisper model. They train the model with 65 thousand datasets collected from 208 speech-impaired users. The word recognition accuracy they achieved is 95.9% using the whisper-small pre-trained model. [11] have developed a contactless elevator operation system using CNN-based keyword spotting. The CNN model is deployed in the Arduino 33 BLE Sense Board using Tiny ML. They trained the model with a dataset of six labels and achieved an accuracy of 83.5%. [12] have developed a lightweight speaker-independent speech recognition model for keyword spotting in the French language to control the robotic arm. They have used transfer learning by using the model trained on the google dataset [1]. The accuracy they achieve is 94.7%. [13] have developed WAV2KWS using transfer learning which uses the WAV2VEC 2.0 [14] model. They have used the google dataset [1] for training, validation and testing of the model. They achieve an accuracy of 97.9%. [15] have developed a KWS system using the Swin-Transformer [16] model. Speech communication in the worksite plays an important role. The adoption of KWS makes it suitable for enhancing communication and reducing errors in decision-making on construction sites. [17] have developed a framework for keyword identification on construction sites. The model they have used consists of a 1-D CNN model. The model is trained with 12 hours of raw audio data consisting of crane signalman speech commands, referred to as 'keywords'. The dataset includes speech commands from 45 volunteers representing 13 different ethnolinguistic backgrounds and various accents. The percentage of accuracy they achieved is 93.8%.

PROPOSED SYSTEM

In KWS, the speech command is captured in .wav format. The captured voice is pre-processed by trimming to remove the silent parts at the beginning and end of the speech signal. The Pydub library is used to trim the speech signal. Then, the speech is converted into a 16-bit signal and from stereo to mono. After preprocessing the speech signal, the

MFCC feature is extracted using the Librosa library. The process of predicting the command in the KWS system is shown in Fig. 2.

Fig. 2 KWS Workflow.

Dataset Preparation

The dataset is created by capturing the voices of 15 individuals, 10 males and 5 females, as detailed in Table 1. Each person utters the words listed in Fig. 3 ten times. The words include numbers (0 to 9), seven colours, and various directions. The speech is recorded at a sampling rate of 41,000 Hz using a ZOOM H1n recorder. Table 2 outlines the specifications of the recorded speech for the training dataset. There is a total of 3,600 audio samples (15 * 24 * 10). The recorded audio files contain silence at the beginning and end, which is removed using the PRAAT tool.

Table 1: Dataset details

Male	10
Female	5
Age range	20 to 30
Total audio sample	3600

Table 2: Speech specification

Sampling Frequency	44 KHz
Recording Device	Zoom H1N1
Quantization	16 bits
Channel	Mono
File format	.wav
Language	Odia

Table 3: Selected words for KWS

In Odia Script	In English	Pronunciation in Odia	In Odia	In English	Pronunciation in Odia
ଶୂନ	Zero	Suna	ଉତ୍ତର	North	Uttara
ଏକ	One	Eka	ଦକ୍ଷିଣ	South	Dakshina
ଦୁଇ	Two	Dui	ନେଳି	Green	Neli
ତିନି	Three	Tini	ଲାଲ	Red	Lal
ଚାରି	Four	Chari	ସବୁଜ	Blue	Sabuja
ପାଞ୍ଚ	Five	Pancha	ହଳଦିଆ	Yellow	Haladia
ଛଅ	Six	Chha	ନାରଙ୍ଗି	Orange	Narangi
ସାତ	Seven	Sata	ବାଇଗଣି	Violet	Baigani
ଆଠ	Eight	Aatha	ଧଳା	White	Dhala
ନଅ	Nine	Na	ଡାହାଣ	Right	Dahana
ପୂର୍ବ	East	Purba	ଆଗ	Front	Aaga
ପଶ୍ଚିମ	West	Paschima	ବାମ	Left	Bama

Feature Extraction

The MFCC method is commonly used in speech and audio processing for feature extraction. The spectral characteristics of sound are captured by MFCC, making them suitable for machine learning applications like speech recognition and music analysis. MFCCs represent the shape of the power spectrum of a sound signal. They are obtained by initially converting the raw audio signal into the frequency domain using techniques such as the Discrete

Fourier Transform (DFT). Then the mel-scale is applied to approximate human auditory perception of sound frequency. Cepstral coefficients are then computed from the mel-scaled spectrum, as demonstrated in equation 1. MFCCs are valuable because they highlight important features of the audio signal for human speech perception while filtering out less relevant information. Consequently, they are effective for tasks such as speaker recognition, emotion detection, and speech-to-text conversion. Fig. 4 shows the complete MFCC feature extraction process. The speech signal, the Fourier transform of the speech signal, and the MFCC features of digits zero to three are shown in Fig. 5.

$$MFCCs = DCT(\log(Mel\ Filter\ Bank\ (|FFT(Windowed\ Frame)|^2))) \quad (1)$$

Fig. 4 . MFCC feature extraction process

Model Architecture

In this research, three different models are implemented and the results of different models are studied. The architecture of different models is as follows:

CNN

The model comprises three fully connected two-dimensional convolution layers, each utilizing a max pooling function to reduce input dimension. The input and hidden layers employ the RELU activation function, while the output layer uses the softmax function. Batch normalization is applied to the input and hidden layers. A 3x3 kernel with a stride of two is utilized. The first two convolutional layers have 16 feature maps, and the last convolutional.

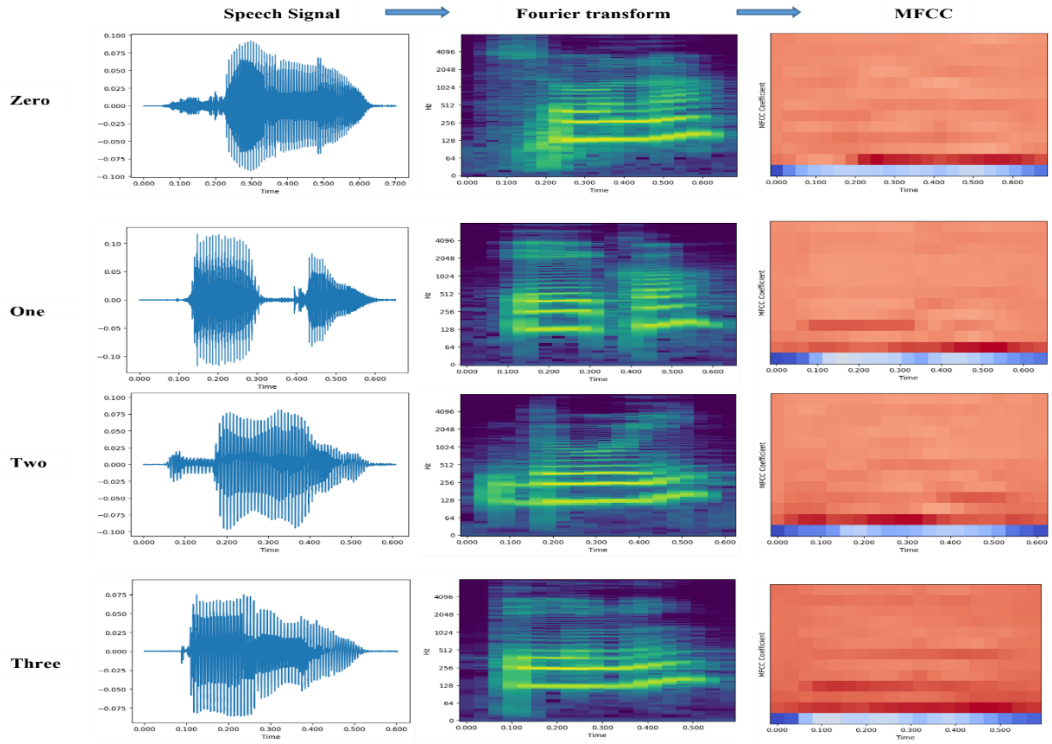

Fig. 5 Speech Signal, Fourier Transform and MFCC feature of digit Zero to Three

layer has 64 feature maps. The model's hyperparameters include a batch size of 32, no dropout, and a learning rate of 0.001. Training is carried out using the Adam optimizer over 100 epochs. The model's architecture is depicted in Fig. 6.

Fig.6 CNN model architecture used for KWS

LSTM

LSTM is widely used for speech recognition. It is also used in low-powered devices like FPGA-controlled devices [18]. Our model consists of 3 LSTM layers of dimension 128. Softmax function is used in the output layer. The model has 128 feature maps in all the LSTM layers. The hyperparameters used in the model are Batch Size as 32, no dropout and Learning Rate as 0.001. We train the network using the Adam optimizer. The model is trained with 100 epochs. Fig. 7 represents the architecture of the model.

Fig.7 LSTM model architecture used for KWS

CNN and LSTM

Our architecture comprises two fully connected two-dimensional convolutional layers followed by two LSTM layers. Each convolutional layer employs a max pooling function to decrease the input dimensions. The RELU activation function is utilized in both the input and hidden layers, while the output layer employs the Softmax function. Batch normalization is implemented in the input and hidden layers. The convolutional kernels are sized at 4, with a stride of 2. The first convolutional layer contains 32 feature maps, and the second layer contains 64 feature maps. The hyperparameters for the model include a batch size of 32, no dropout, and a learning rate of 0.001. We train the network using the Adam optimizer. The model is trained with 100 epochs. Fig. 8 represents the architecture of the model.

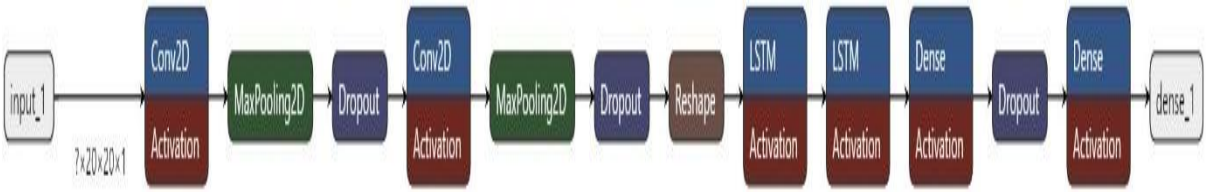

Fig.8 CNN+LSTM model architecture used for KWS

Model Compression

Generally, the size of deep learning models is too big and can only be deployed in the cloud or on a high-performance computer. The Edge devices have very little memory and storage space. In order to deploy the deep learning models on edge devices, the size of the model needs to be smaller. There are various techniques to compress the DL model. We have compressed our model using the quantization technique.

Quantization refers to the process of reducing the precision of the numbers used to represent an ML model's parameters such as weights and activations [19]. In this research, the model is quantized by representing float32 by INT8 (Fig. 9). The models are quantized using TensorFlow. A fractional number is quantized using Equation 2 and de-quantized during inference using Equation 3.

$$X_{quant} = round(scale \times X + zeropoint) \quad (2)$$

$$X_{dequant} = \frac{X_{quant} - zeropoint}{scale} \quad (3)$$

$$scale = \frac{255}{max(X) - min(X)} \quad (4)$$

$$zeropoint = -round\big(scale \times min(X)\big) - 128 \quad (5)$$

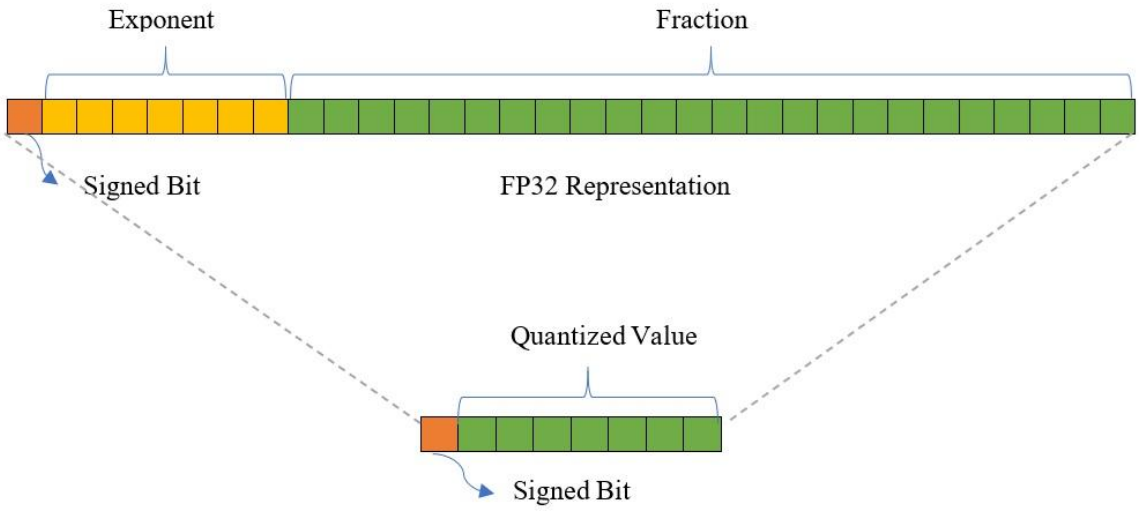

Fig. 9 Float32 to INT8 Compression

RESULTS

The dataset is split into 80%, 10% and 10% for training, testing and validation. Table 3 represents the accuracy, precision, recall and F1-score for our CNN, LSTM and CNN+LSTM models. It is observed that CNN performs better than other models for our dataset. The accuracy and loss graph of CNN, LSTM and CNN+LSTM is shown in Fig. 10. The Fig. 11 represents the confusion matrix of CNN, LSTM and CNN+LSTM models for fifteen classes. The size of these three models before quantization and after quantization is shown in Table 4. It has been observed that the size of the models is reduced by 3 times after quantization. Table 5 represents the accuracy of models before quantization and after quantization. The accuracy of the models is preserved after quantization. We have performed the memory profiling of all the quantized models. The peak memory utilization for CNN and LSTM is 12.35 KB and 12.92 KB for the CNN+LSTM model as shown in Fig. 12. The memory size of most of the edge devices is more than 100 KB. So, our models can easily be deployed on any edge device for on-device inference.

Table 3: Accuracy, Precision, Recall, F1-Score of CNN, LSTM and CNN+LSTM model

Model	Accuracy	Precision	Recall	F1-Score
CNN	97	97	97	97
LSTM	85.6	87	86	86
CNN+LSTM	96	97	96	96

Table 4: Size of models before and after quantization

Model	Model size without quantization (in KB)	Model size with quantization (in KB)
CNN	1599	521
LSTM	4040	1358
CNN+LSTM	3107	1035

Table 5: Accuracy of model before and after quantization

Model	Model Accuracy without quantization	Model Accuracy with quantization
CNN	97	97
LSTM	85.6	86
CNN+LSTM	96	96

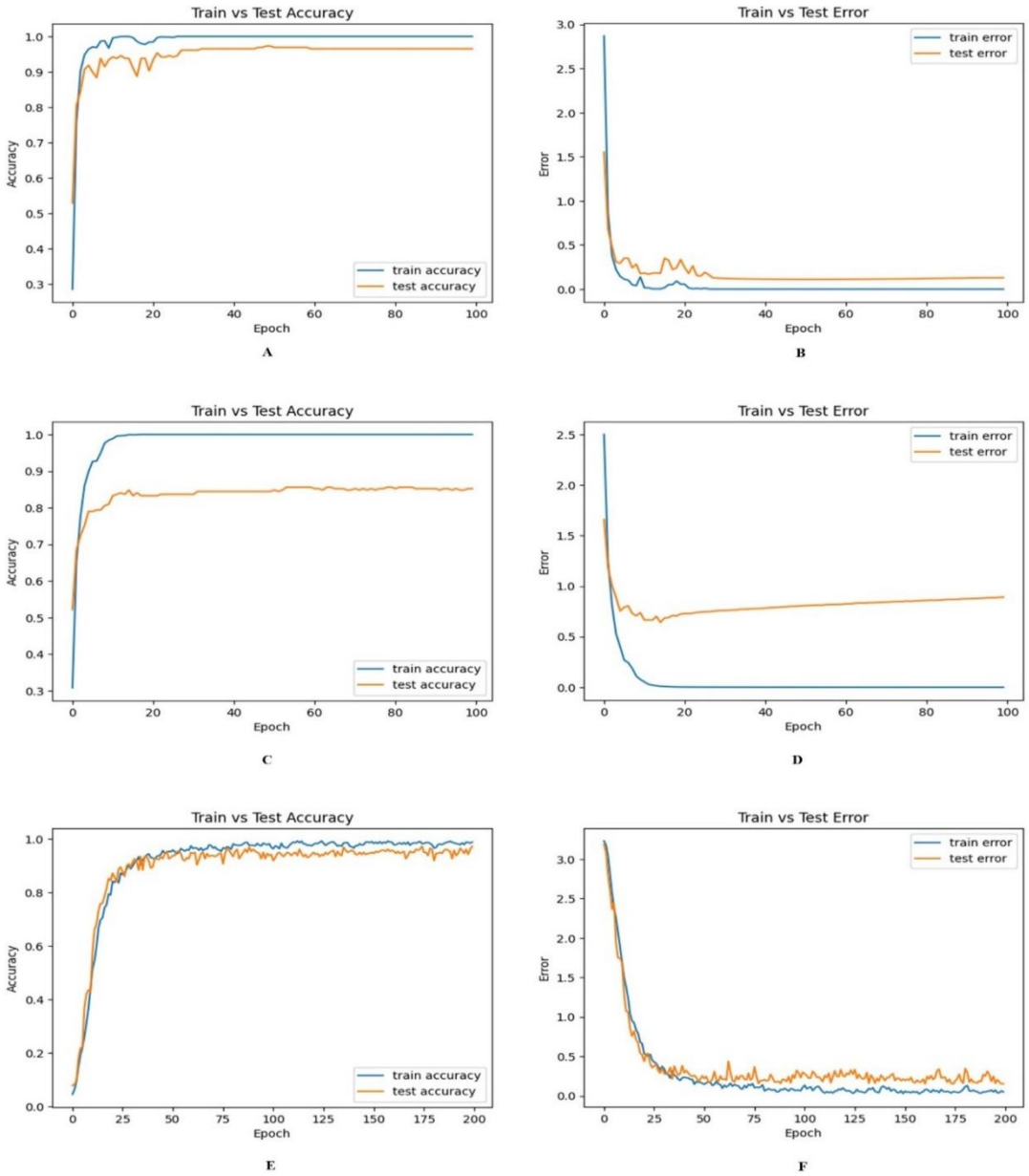

Fig. 10 (A) CNN model Accuracy; (B) CNN model Loss; (C) LSTM model Accuracy; (D) LSTM model Loss; (E) CNN+LSTM model Accuracy; (F) CNN+LSTM model Loss

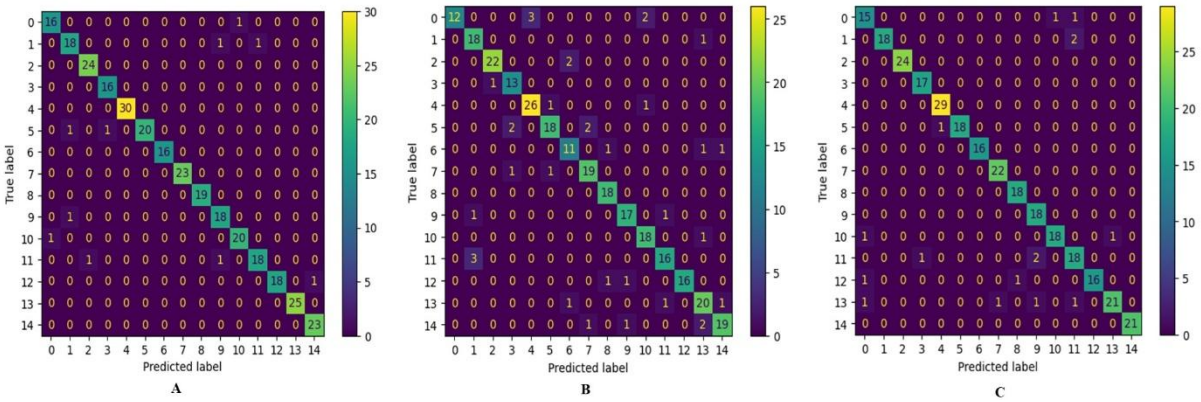

Fig. 11 (A) Confusion Matrix of CNN model; (B) Confusion Matrix of LSTM model; (C) Confusion Matrix CNN_LSTM model

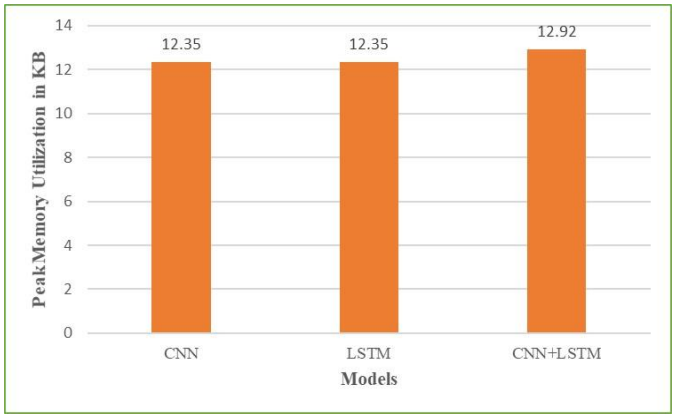

Fig. 12 Comparison of peak memory utilization of CNN, LSTM and CNN+LSTM models

CONCLUSION

This study demonstrates the feasibility of implementing an efficient keyword-spotting system for the Odia language on edge devices. Three different models are designed and tested on the Raspberry Pi 3B device. The models are trained and tested with the dataset developed by us as there is no Odia dataset available for KWS in the public domain. Among the three models, the CNN model performs better than the other two models. After quantization of the models, there is a 3x reduction in the model size. It is found in the memory profiling that the peak memory utilization is 12.35 KB for the CNN and LSTM model and 12.92 KB for the CNN+LSTM model which is the maximum as compared to the other two models. Generally, the memory provided by any edge device is around 256 KB. Thus, our model can run on a standard edge device without compromising the accuracy of inference.

REFRENCES

[1] P. Warden, “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition,” Apr. 09, 2018, arXiv: arXiv:1804.03209. Accessed: Jun. 13, 2024. [Online]. Available: http://arxiv.org/abs/1804.03209

[2] B. R. Bag, “Speech Recognition of Isolated Odia Digits Using Convolutional Neural Network,” in 2023 IEEE International Conference on Contemporary Computing and Communications (InC4), Bangalore, India: IEEE, Apr. 2023, pp. 1–3. doi: 10.1109/InC457730.2023.10263195.

[3] P. Mohanty and A. K. Nayak, “Isolated Odia Digit Recognition Using HTK: An Implementation View,” in 2018 2nd International Conference on Data Science and Business Analytics (ICDSBA), Sep. 2018, pp. 30–35. doi: 10.1109/ICDSBA.2018.00013.

[4] P. Mohanty and A. K. Nayak, “CNN based keyword spotting: An application for context based voiced Odia words,” Int. J. Inf. Technol., vol. 14, no. 7, pp. 3647–3658, Dec. 2022, doi: 10.1007/s41870-022-00992-z.

[5] S. S. Balasuriya, L. Sitbon, A. A. Bayor, M. Hoogstrate, and M. Brereton, “Use of voice activated interfaces by people with intellectual disability,” in Proceedings of the 30th Australian Conference on Computer-Human Interaction, Melbourne Australia: ACM, Dec. 2018, pp. 102–112. doi: 10.1145/3292147.3292161.

[6] G. Benelli, G. Meoni, and L. Fanucci, “A low power keyword spotting algorithm for memory constrained embedded systems,” in 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Verona, Italy: IEEE, Oct. 2018, pp. 267–272. doi: 10.1109/VLSI-SoC.2018.8644728.

[7] D. Mulfari, L. Carnevale, and M. Villari, “Toward a lightweight ASR solution for atypical speech on the edge,” Future Gener. Comput. Syst., vol. 149, pp. 455–463, Dec. 2023, doi: 10.1016/j.future.2023.08.002.

[8] J. S. P. Giraldo and M. Verhelst, “Laika: A 5uW Programmable LSTM Accelerator for Always-on Keyword Spotting in 65nm CMOS,” in ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC), Dresden: IEEE, Sep. 2018, pp. 166–169. doi: 10.1109/ESSCIRC.2018.8494342.

[9] H. Mazzawi et al., “Improving Keyword Spotting and Language Identification via Neural Architecture Search at Scale,” in Interspeech 2019, ISCA, Sep. 2019, pp. 1278–1282. doi: 10.21437/Interspeech.2019-1916.

[10] D. Mulfari and M. Villari, “A Voice User Interface on the Edge for People with Speech Impairments,” Electronics, vol. 13, no. 7, p. 1389, Apr. 2024, doi: 10.3390/electronics13071389.

[11] A. S. Pimpalkar and D. V. Niture, “Towards Contactless Elevators with TinyML using CNN-based Person Detection and Keyword Spotting”, https://doi.org/10.48550/arXiv.2405.13051.

[12] S. Poirier, U. Côté-Allard, F. Routhier, and A. Campeau-Lecours, “Efficient Self-Attention Model for Speech Recognition-Based Assistive Robots Control,” Sensors, vol. 23, no. 13, p. 6056, Jun. 2023, doi: 10.3390/s23136056.

[13] D. Seo, H.-S. Oh, and Y. Jung, "Wav2KWS: Transfer Learning From Speech Representations for Keyword Spotting," IEEE Access, vol. 9, pp. 80682–80691, 2021, doi: 10.1109/ACCESS.2021.3078715.
[14] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," Oct. 22, 2020, arXiv: arXiv:2006.11477. Accessed: Sep. 09, 2022. [Online]. Available: http://arxiv.org/abs/2006.11477
[15] C. Sun, B. Chen, F. Chen, Y. Leng, and Q. Guo, "Speech Keyword Spotting Method Based on Swin-Transformer Model," Int. J. Comput. Intell. Syst., vol. 17, no. 1, p. 61, Mar. 2024, doi: 10.1007/s44196-024-00448-1.
[16] Z. Liu et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada: IEEE, Oct. 2021, pp. 9992–10002. doi: 10.1109/ICCV48922.2021.00986.
[17] A. Mansoor, S. Liu, G. M. Ali, A. Bouferguene, M. Al-Hussein, and I. Hassan, "Keyword identification framework for speech communication on construction sites," Modul. Offsite Constr. MOC Summit Proc., pp. 106–113, Sep. 2022, doi: 10.29173/mocs271.
[18] S. Han et al., "ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA," in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, in FPGA '17. New York, NY, USA: Association for Computing Machinery, Feb. 2017, pp. 75–84. doi: 10.1145/3020078.3021745.
[19] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," Dec. 15, 2017, arXiv: arXiv:1712.05877. Accessed: Oct. 16, 2024. [Online]. Available: http://arxiv.org/abs/1712.05877