Based on STM32 of CNN Speech Keyword Command Recognition System

KUANG Wenbo [1], LUO Weiping [1,2]

(1. *School of Mechanical Engineering and Automation*, *Wuhan Textile University*, *Wuhan 430200*, *China;*
2. *Hubei digital textile equipment key laboratory*, *Wuhan 430200*, *China*)

Abstract: Speech recognition is a hot topic in the field of artificial intelligence. Generally, speech recognition models can only run on large servers or dedicated chips. This paper presents a keyword speech recognition system based on a neural network and a conventional STM32 chip. To address the limited Flash and ROM resources on the STM32 MCU chip, the deployment of the speech recognition model is optimized to meet the requirements of keyword recognition. Firstly, the audio information obtained through sensors is subjected to MFCC (Mel Frequency Cepstral Coefficient) feature extraction, and the extracted MFCC features are input into a CNN (Convolutional Neural Network) for deep feature extraction. Then, the features are input into a fully connected layer, and finally, the speech keyword is classified and predicted. Deploying the model to the STM32F429, the prediction model achieves an accuracy of 90.58%, a decrease of less than 1% compared to the accuracy of 91.49% running on a computer, with good performance.

Keywords: Speech Recognition, STM32, MFCC, Convolutional Neural Network

1 Introduction

In recent years, the development of artificial intelligence has been rapid, and speech recognition has been active in various fields of the industry, with its shadow everywhere. Speech recognition also has many applications in human-machine interaction. Using speech recognition technology, people can operate devices in a non-contact manner, which greatly improves safety and convenience for users. With the integration of artificial intelligence into people's lives, many electronic devices have embedded speech recognition functionality, which is particularly common in smartphones and smart home devices. People can use the speech recognition function to wake up their phone at any time and perform corresponding functions based on the conversation content[1]. Smart homes can operate various appliances and switches through speech recognition, providing maximum convenience for people. Therefore, the accuracy of speech recognition is also very important. Currently, common speech recognition systems run on high- performance AI chips, with low versatility but able to quickly and accurately recognize specific languages. This article introduces a method for recognizing speech keywords using a conventional ARM chip STM32[2]. This system is composed of STM32DSP library + STM32F429IGT6 + STM32CUBEMX + X-CUBE-AI, and realizes the function of speech recognition on a single-chip system [3]

2 Methods and Experiments

2.1 Dataset

In this experiment, the public dataset Speech

Commands Dataset is used for recognition. This dataset is a set of one-second WAV files, with each folder containing one spoken word. The audio data is saved as 16-bit encoded WAVE files with a sampling rate of 16000. The dataset includes twenty core commands, with most speech recorders saying each one five times. In this paper, four keywords are selected for speech recognition: "yes", "no", "stop", and "go". For audio with a sampling rate of 16000, 16000 data points are collected in one second. Therefore, a complete 1-second sample will have 16000 audio data points. After screening the dataset for the four categories mentioned above and removing samples that do not meet the requirements, the number of screened samples is shown in Table 1.

Table 1 Data Filtering

Category	Yes	No	Stop	Go
Number before Screening	4044	3941	3872	3880
Number after Screening	3692	3545	3563	3478

2.2 MFCC (Mel-frequency Cepstral Coefficient) Feature Extraction

The first step in speech recognition is feature extraction, which can effectively reduce interference signals and improve the accuracy of speech recognition. Generally, the speech data obtained by sensors is in the time domain. In the time domain, it is not easy to distinguish the characteristics of different sounds. Therefore, the characteristics of the sound are extracted by Fourier transform and certain processing to facilitate speech recognition.

MFCC[4](Mel-Frequency Cepstral Coefficient) is a common method for sound feature extraction. MFCC first extracts the frequency domain information of the sound. Since the recognition rate of the human ear to sound is not linear, the obtained frequency information needs to be filtered by the Mel filter bank, and then the result is subjected to DCT transformation [5] to obtain the MFCC coefficients, which are the features of the sound [6].

This article mainly uses the following five steps to obtain the MFCC features of the audio: 1. Pre-emphasis and frame segmentation of the audio signal. 2. Calculation of the power spectrum of each frame. 3. Filtering with a Mel filter bank. 4. Discrete cosine transform (DCT). 5. Retain the 2-13 coefficients after DCT as the features of the sound. The process of MFCC extraction is shown in Fig.1

Firstly, pre-processing is performed on the original signal by applying pre-emphasis, and the calculation formula is shown as Equation (1):

$$Y_t = x_t - 0.97 * x_{t-1} \quad (1)$$

The 1-second audio signal is divided into frames of 25ms, and to ensure the continuity of the signal, each frame is shifted 10ms backwards so that adjacent frames have overlapping parts. Then, a window is applied to the data of each frame, but this article does not use a window function and keeps the original segmented signal. For each frame of the signal, FFT operation is performed as shown in Equation (2), where k is the length of the DFT.

$$S_i(k) = \sum_{n=1}^{N} s_i(n) h(n) e^{-j2\pi kn/N}, 1 \leqslant k \leqslant N \quad (2)$$

The energy spectrum is calculated as shown in Equation (3):

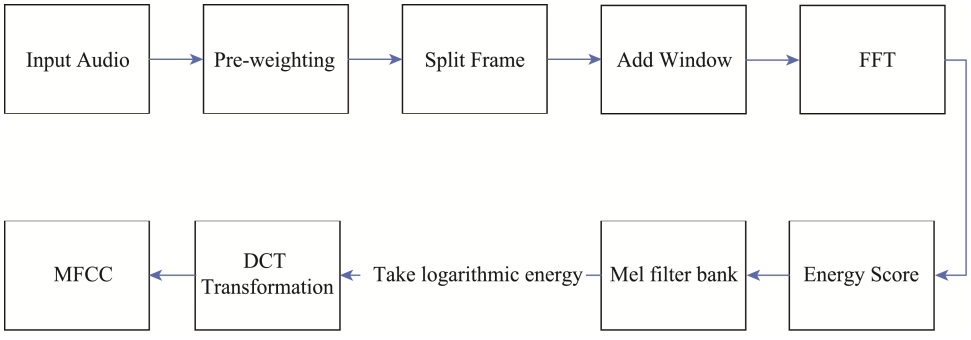


Fig.1 Flow Chart of MFCC Features

$$P(k)=\frac{1}{N}|S_i(k)|^2 \tag{3}$$

The Mel filter bank is calculated by dividing the frequencies uniformly on the Mel scale, and 26 triangular filter banks are used to filter the power spectrum. The conversion formula for Mel frequency is shown in Equation (4):

$$M(f)=1125\ln(1+f/700) \tag{4}$$

The conversion formula from Mel frequency to frequency is shown in Equation (5):

$$M^{-1}(m)=700(e^{\frac{m}{1125}}-1) \tag{5}$$

The formula for creating the Mel filter bank is shown in Equation (6):

$$\begin{cases} 0, k<f(m-1) \\ \dfrac{k-f(m-1)}{f(m)-f(m-1)}, f(m-1)\leqslant k\leqslant f(m) \\ \dfrac{f(m+1)-k}{f(m+1)-f(m)}, f(m)\leqslant k\leqslant f(m+1) \\ 0, k>f(m+1) \end{cases} \tag{6}$$

After passing the signal through the Mel filter bank, the logarithm of the 26 energy data is taken, and then the discrete cosine transform (DCT) is performed on the 26 points of the signal to obtain the MFCC feature of the audio. This article takes the 2nd to 13th coefficients as the MFCC feature of the frame.

2.3 Model and Processing

After extracting the MFCC features from the above 1-second dataset, we can obtain 99 frames of data with 13 speech features, which means an audio data can be represented as an array of shape (99, 13). Combining the MFCC features of different frames together, we can obtain a feature spectrogram as shown in Fig.2

Convolutional Neural Networks (CNNs) have wide applications in the field of image recognition. The convolutional kernel in a CNN can extract deeper features from images with fewer parameters, and thus perform well in image classification tasks. Based on the previous operations, which converted audio signals into image information, the problem of speech recognition and classification can be transformed into the same type of problem as traditional image recognition. In this paper, a CNN is used to classify and recognize the MFCC feature maps of speech. The network is designed to be implemented on a microcontroller and constructed using TensorFlow [7]. It has five layers: the first layer is a convolutional layer, which uses 6x3x3 convolutional kernels to perform a convolution operation on the input image with a stride of 1. The second layer consists of 16x3x3 convolutional kernels with a stride of 1. The third layer is a fully connected layer with 120 nodes, the fourth layer has 84 nodes, and the fifth layer is the output layer, which has 4 nodes corresponding to the four output categories. A max-pooling layer is added to the output of each convolutional layer to reduce the size of the feature maps by half, and a ReLU activation function is applied to the output of each layer in the network.

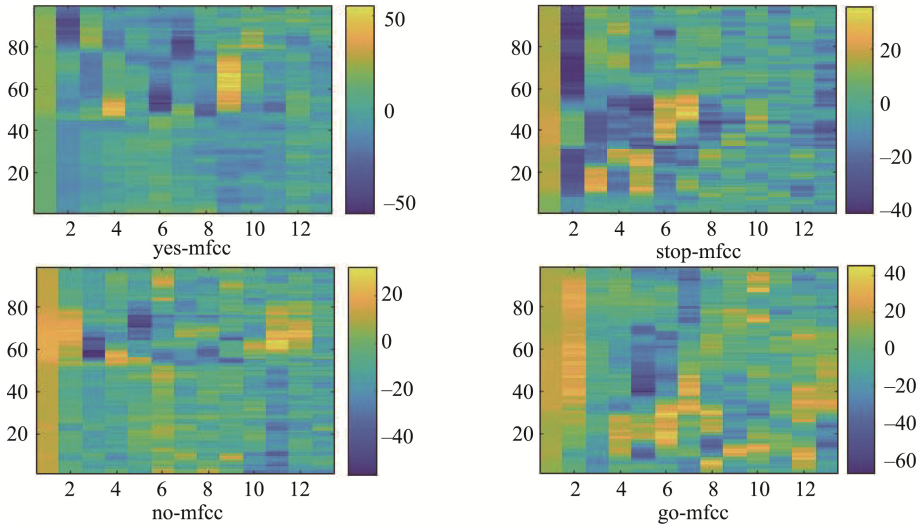


Fig.2 Spectrum of Different Phonetic Features

This model uses the commonly used cross- entropy loss function in classification models, expressed as formula 7:

$$\varepsilon=-\frac{1}{X}\sum_i\sum_{c=1}^{Y} y_{ic}\log(p_{ic}) \quad (7)$$

Here, X represents the total number of samples, Y represents the number of final classification categories, y_ic represents the symbol function, which is 1 if the classification result is correct and 0 otherwise. p_ic is the probability that sample i belongs to category c

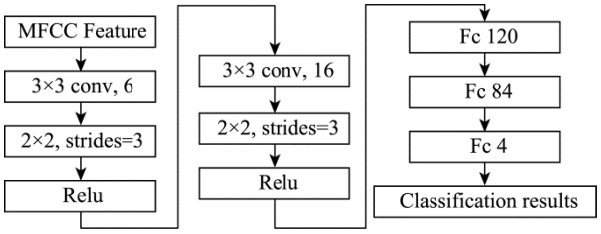


Fig.3 CNN Network Model Diagram

The experimental dataset in this study has four categories: yes, no, go, and stop, with a total of 14,278 audio data. The dataset files were shuffled and divided into training set, validation set, and test set. The training set accounts for 60%, the validation set accounts for 20%, and the test set accounts for 20%. There are 8,566 training data, 2,856 validation data, and 2,856 test data. TensorFlow was used to build the model, and its parameters are shown in Fig.4.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(4, 97, 11, 6)	60
max_pooling2d (MaxPooling2D)	(4, 48, 5, 6)	0
re_lu (ReLU)	(4, 48, 5, 6)	0
conv2d_1 (Conv2D)	(4, 46, 3, 16)	880
max_pooling2d_1 (MaxPooling2	(4, 23, 1, 16)	0
re_lu_1 (ReLU)	(4, 23, 1, 16)	0
flatten (Flatten)	(4, 368)	0
dense (Dense)	(4, 120)	44280
dense_1 (Dense)	(4, 84)	10164
dense_2 (Dense)	(4, 4)	340

Total params: 55,724
Trainable params: 55,724
Non-trainable params: 0

Fig.4 Model Structure

2.4 Hardware and Software Design for Microcontroller

This design uses the STM32F429ZGT6 microcontroller, which has 1024KB of Flash and 256KB of RAM. It also has timers, counters for controlling timing, and various communication interfaces such as AD/DA modules for interacting with external devices. The STM32F4 series microcontrollers can be programmed using Keil 5 software and common C language code, allowing it to work according to our own wishes. This microcontroller has a clock multiplier circuit, which, after being multiplied internally, can reach a clock speed of 180M using an external 8M crystal oscillator and software programming, ensuring fast system operation. ST has a rich software library, and in order to facilitate the implementation of DSP functions, ARM has specifically created a DSP library CMSIS-DSP, which is mainly used for digital signal processing. It includes common FFT operations, and using the optimized DSP library code provided by the manufacturer can greatly improve computing efficiency and accuracy. The system structure for this experimental design is shown in Fig.5.

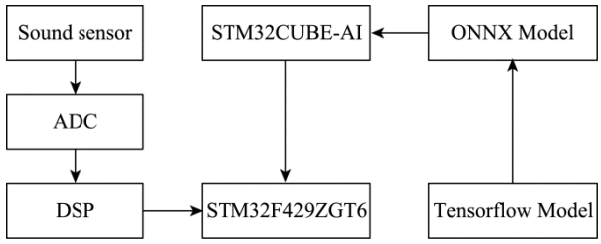


Fig.5 System Architecture

The STM32F429 collects sound sensor data through ADC and uses the DSP library to extract MFCC features from the collected sound data to obtain the sound feature vector. Using the Tensorflow2 framework, the trained model file is converted into an onnx file [8], which is then input into the STM32CUBE-AI toolbox and converted into C code and imported into the microcontroller. The sound feature vector is calculated by the model inside the microcontroller, and the speech is recognized accordingly.

On the computer, the Tensorflow-built code is trained and an onnx model file is generated, with a file

size of 219k. This model is imported into the STM32CUBE-AI[9] toolbox for analysis, which shows that running this model requires 48.54KB FLASH and 9.63KB RAM. The STM32F429ZGT6 fully meets the operating conditions and can run normally on the microcontroller. The analysis results are shown in Fig.6.

Complexity: 240900 MACC
Used Flash: 48.54 KiB (48.54 KiB over 1024.00 KiB Internal)
Used Ram: 9.63 KiB (9.63 KiB over 256.00 KiB Internal)
Achieved compression: 6.75
Analysis status: done

Fig.6 Analysis Results

2.5 Model Training

The deep learning model was built and trained using the Tensordlow2 framework. During training, the dataset was shuffled using the built-in function in TensorFlow to avoid the network learning the order information and to improve the final accuracy of the model. The Adam optimizer was used for gradient updates, which has the characteristics of simple implementation and high computational efficiency. It can automatically adjust the learning rate and control the step size, which is helpful for the model to quickly and accurately find the optimal parameters. The cross-entropy function was chosen as the loss function for the model, which is widely used in classification problems. The labels for the four categories: go, no, stop, yes were set as 0, 1, 2, and 3, respectively. With the above parameter settings, the model was trained for 100 epochs, and the training results are shown in Fig.7 and Fig.8.

From the training results shown in the figure, it can be seen that the model has achieved high accuracy after several iterations. The accuracy of the training set

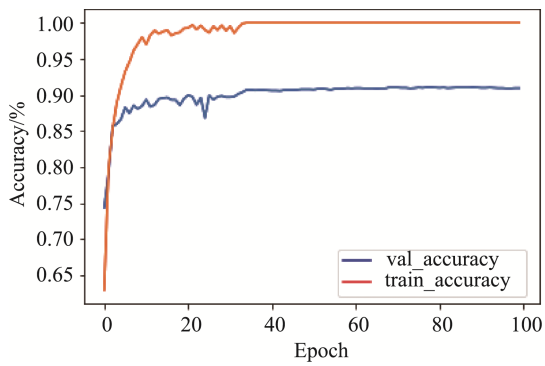


Fig.7 Diagram of Training Process

is close to 100% and the training set loss is close to 0. The validation set accuracy reaches 91.49%, indicating good classification performance of the model.

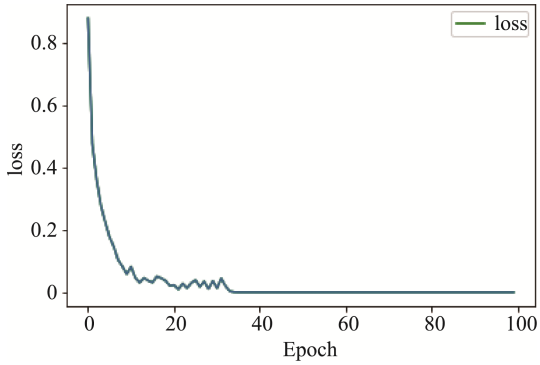


Fig.8 Training Loss

3 Experimental Results

The confusion matrix [10] of the classification results is presented below to evaluate the performance of the model. The values on the diagonal of the confusion matrix can be used to determine the classification accuracy. The model was tested on a previously unseen test set, and the resulting confusion matrix is shown in Fig.9.

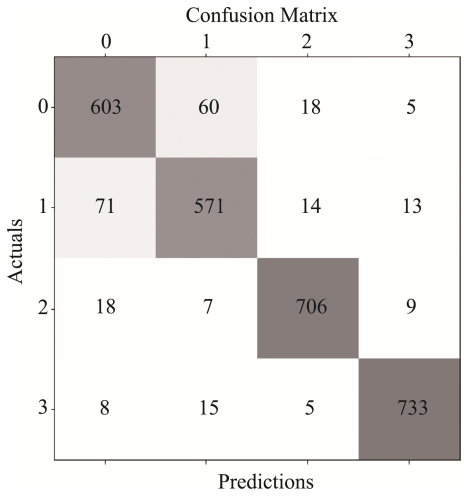


Fig.9 Tensorflow Classification Model Confusion Matrix

According to the confusion matrix, the recognition accuracy can be calculated, as shown in Table 2

Based on the confusion matrix, it can be seen that the neural network model running on the embedded STM32 microcontroller is basically consistent with the

results on the PC. The accuracy only drops by less than 1%. The loss in accuracy comes from converting the model into code that can be run on the microcontroller using the STM32CUBE-AI toolbox. In this process, the toolbox compresses the model, sacrificing some accuracy in exchange for faster running speed and smaller model size.

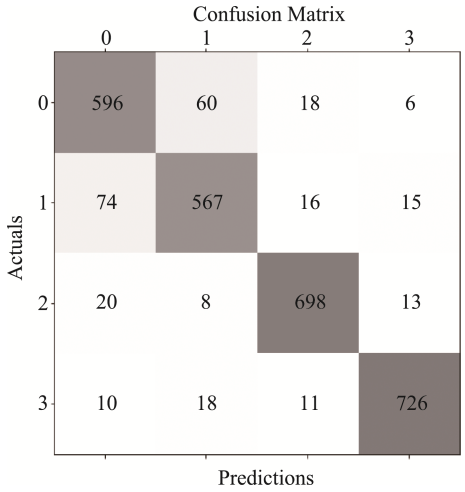


Fig.10 STM32F429 Classification Model Confusion Matrix

Table 2 Model Test Accuracy

Model	Tensorflow Model	STM32F429 Running Model
Test Accuracy	91.4%	90.58%

4 Conclusion

In this study, we used the STM32F429ZGT6 microcontroller to collect sound sensor data through ADC and extracted features using the DSP library. We then passed the feature data to C code converted by the STM32CUBE-AI toolbox for recognition of yes, no, stop, and go. The results showed that the model running on the STM32F429 has an accuracy of 90.58%, with less than 1% decrease in accuracy compared to the PC model.

In this experiment, the classification category is 4, which is a small number of classification categories. The speech categories can be continued to be added in future work to make the model richer in classification categories and used in a wider range of fields. This implementation provides a method to run a neural network model using a microcontroller with high recognition accuracy. Using more data sets, and building more complex network models, the prediction accuracy of the model can be further improved.

References

[1] Kim, Young Jong. (2015). *The Real-time Shopping System using Multipurpose Visual Language with Voice Recognize*. Journal of the Korea Academia-Industrial cooperation Society, 16(6).

[2] Khan, Isra. (2022). *An Intelligent Framework for Person Identification Using Voice Recognition and Audio Data Classification.* Applied Computer Systems, 27(2).

[3] Ming, L. (2022). *Method for implementing neural network on microcontroller chip. Modern Electronics Technique*. 43(22), pp.1-5+9.

[4] Pengfei, H. (2022). *Bird sound recognition based on MFCC-IMFCC and GA-SVM*. Computer Systems & Applications ,31(11), pp.393-399.

[5] Xin, Z. (2021). *Fourier Transformation of Pulse. Waves Based on STM32 and DSP Library.* Modern Computer ,27(28), pp.112-115

[6] Bin, L. (2022). *Fault diagnosis of wind turbine blades based on MFCC sound feature signal extraction*. Plant Maintenance Engineering, pp.148-149.

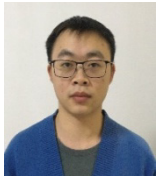
[7] Xiao, L. (2020). *Research on Image Classification Algorithms Based on Tensorflow2.0.* Modern Computer, pp.63-68+74.

[8] Kai, H. (2022). *Research and implementation of TensorFlow and ONNX model conversion. Xidian University.*

[9] STM32Cube software ecosystem empowers mass market embedded voice control technology applications. (2022) 22(08): pp.65.

[10] Ying Y. (2021). *Confusion Matrix Classification Performance Evaluation and Python Implementation.* Modern Computer, pp.70-73+79.

Author Biographies



KUANG Wenbo M.Sc. candidate, majored in big data processing, machine learning at Wuhan Textile University.

E-mail: 472013066@qq.com

LUO Weiping Master, Professor. Her main research Interests include big data processing, detection technology and automatic control.

E-mail: 651871236@qq.com