Al-Qadisiyah JournalFor Engineering Sciences 18 (2025) 320 – 326

Contents lists available at: http://qu.edu.iq

Al-Qadisiyah Journal for Engineering Sciences

Journal homepage: https://qjes.qu.edu.iq

Research Paper

Design and implementation of upper limb controlled by human voice and artificial intelligence

Ahmed M. A. Ahmed [1]✉, ***Mithaq N. Raheema*** [2], ***and Jabbar S. Hussein*** [2]

[1] *College of Engineering, Department of Electrical and Electronic Engineering, University of Kerbala, Kerbala, Iraq.*
[2] *College of Engineering, Department of Prosthetics and Orthotics Engineering, University of Kerbala, Kerbala, Iraq.*

ARTICLE INFO

Article history:
Received 16 May 2024
Received in revised form 05 August 2024
Accepted 20 August 2025

keyword:
Automatic Speech Recognition
Multi-Layer Feed Forward Neural Network
Mel-frequency cepstral coefficients
Root Mean Square
Standard Deviation

ABSTRACT

Given the restrictions and complexity of EMG-based hand prostheses, the present work chose to investigate the possibility of a much simpler method based on voice-operated. Voice commands are simpler to understand than EMG signals to regulate the prosthetic arm. Therefore, the first and most important benefit of this method is that the related prosthesis is simple to use. Speech recognition is a vital focus in artificial intelligence, serving as a prominent mode of human interaction. Researchers have developed speech-driven prosthetic hand systems using traditional speech recognition frameworks and neural network models. This work intends to employ the Raspberry Pi 4 Model B – 4GB RAM embedded inside the prosthetic limb instead of Arduino without requiring a computer to lower hand weight; this offers simplicity of usage. The proposed system captures and transforms speech input into features resembling spectrograms. It processes them using the MLFFNN to categorize speech as signals (words) and forward it to the prosthetic hand, fingers, and wrist control system involving six servo motors. 3D printers created a light and sturdy prosthetic locally. This work stands out for the freedom of usage amputees of this limb have, with the ability of the researcher to increase the movements by adding more signals (words). The entire system is implemented in Python using Keras and a deep learning framework with a TensorFlow backend. The simulation results demonstrate an accuracy of 99.4%, real-time test accuracy of 96.05%, and operational validation efficiency of 97.6%. These findings indicate that the MLFFNN can be effectively utilized for real-time control of prosthetic hands.

1. Introduction

The newest technical developments have helped scientists and engineers to create more creative, realistic, and agile hands during the last few decades. To control robotic hands, researchers have in the past used push buttons, keyboards, text, speech, electromyography (EMG) [1], vision, or a combination of these [2]. More easily, humans voice control is a popular concept, especially for gadgets made for persons with disabilities, since individuals communicate and express their thoughts through voice. In this work, speech recognition is the main topic. Automatic Speech Recognition (ASR), or translating raw speech into text, is a key component of artificial intelligence. Three major subsystems make up conventional ASR systems: preprocessing, feature extraction, and classification [3]. Typically, developers employ start and end point detection, normalization, noise reduction, and pre-emphasis for the preprocessing stage. These processed signals are then analysed using multiple features that help classify them into distinct classes while remaining resilient to noise interference. The feature extraction subsystem calculates specific characteristics from these processed signals. The success of traditional ASR systems heavily relies on effective feature extraction methods. Numerous valuable features have been proposed to improve performance [4]. Finally, a classifier assigns these extracted features to their corresponding text labels. Over the past years, ASR researchers have produced many open-source libraries by implementing them in Python, such as TensorFlow, PyTorch, Scikit-learn, the Natural Language Toolkit (NLTK), and the Open-Source Computer Vision Library (OpenCV) [5].

But while traditional ASR systems have improved over time, their performance remains poor in noisy environments. This is the reason that academics have proposed several approaches to reduce the effect of noise and improve the overall performance of ASR systems. Still, there are significant differences in mistake rates between actual situations (such as outdoor or office settings) and high Signal-to-Noise Ratio (SNR) environments (such as studios or laboratories). Said another way, resilience is a major problem with traditional ASR systems in real-world applications. Nowadays, mobile devices can do a wide range of concurrent real-time processing tasks thanks to the usage of modern general-purpose hardware, including Raspberry. They can operate several software packages and support a variety of functions without requiring specialised hardware [5]. Besides, a tonne of tagged voice data has recently been released to the public. Researchers may thus train deep neural networks and apply them in real-time scenarios by using large data and modern parallel computing processors. Smaller ANN networks should be explored for first to reduce size. In many instances, conventional ASR systems have been used to operate prosthetic limbs and other smart devices. This work focuses on designing a speech recognition system using (MLFFNN) to control upper limb prosthetics through speech-based input commands. It converts multiple extracted features such as MFCCs and RMS into textual speech input. Thus, this approach controls five-fingered artificial hands with the wrist. The proposed design is employed to achieve human-friendly robot systems that find applications, including upper limb prosthetics. Several steps have been taken to meet the requirements of building such a system, as shown in Fig. 1.

*Corresponding Author.
E-mail address: ahmedmohii81@gmail.com ; Tel: (+964) 771-883 6174 (Ahmed Ahmed)

https://doi.org/10.30772/qjes.2024.149892.1241

Nomenclature

2-DOF	two degrees of freedom
ANN	Artificial Neural Network
API	Application Programming Interface.
ASR	Automatic Speech Recognition
CNN	Convolution Neural Network
DT	Decision Tree
EMG	Electromyography
HMM	Hidden Markov Model
K-NN	K-Nearest Neighbour
LPC	Linear Predictive Coding
LSTM	Long Short-Term Memory
MFCCs	Mel-frequency cepstral coefficients
MLFFNN	Multi-Layer Feed Forward Neural Network
NLTK	Natural Language Toolkit
OpenCV	Open-Source Computer Vision Library
RBF	Radial Basis Function
ReLU	Rectifier Linear Unit
RMS	Root Mean Square
RNN	recurrent neural networks
SC	Spectral Contrast
SF	Spectral flatness
SNR	Signal-to-Noise Ratio
SSCD	Synthetic Speech Commands Data set
STD	Standard Deviation
TC	Temporal Centroid
TDuf	Twice-Defined user frequency
ZCR	Zero-Crossing Ratio
Greek Symbols	
x	Value of a feature
μ	Mean value
σ	Standard deviation
Subscripts	
N	Number of samples

ata preparation involving recording speech signals and downloading datasets, preprocessing, feature extraction, and recognition. Next, in the subsequent work phase, recognition parameters will pass into the decoding circuit to control servo motor movement of fingers and wrist. Finally, hand movements mimicking spoken words will be generated.

2. Literature review

Recent studies have explored various approaches to control robotic devices using speech commands. One study focused on using vertical inclination frequency-based features in combination with convolutional neural networks for controlling speech-controlled prosthetic hands [6,7]. Another work involved the development of speech-controlled upper limb exoskeletons using machine learning-based voice control integrated with Raspberry Pi. Application of the classification algorithms K-Nearest Neighbor (K-NN) and Decision Tree (DT) is made for this purpose. The mobile robot also does left and right movement, clockwise and anticlockwise rotation, opening and closing, and stop command. This was accomplished by the application of methods including Hidden Markov Model (HMM) and Linear Predictive Coding (LPC) [8]. Moreover, researchers have investigated voice instructions for recurrent neural networks (RNN)-generated robotic systems [9]. Particularly well-performing artificial neural network architecture they created is Long Short-Term Memory (LSTM). The two degrees of freedom (2-DOF) exoskeleton robot "RehaBot"has been modified using this method for rehabilitation of the upper limb [10]. Moreover, scientists operated a five-fingered robotic hand using a radial basis function (RBF) and a spoken character recognition system [11]. These publications provide a range of well researched methods for voice-commanding robotic systems.

3. Methodology

This work proposes a system that captures and transforms speech input from a microphone into features resembling spectrograms. As seen in Fig. 1, specific actions have been taken to satisfy the criteria for designing such a system: Data preparation includes recording voice signals and downloading datasets, preprocessing, feature extraction, and recognition. Precise control of a prosthetic hand with five fingers and wrist motions is achieved by combining an MLFFNN-based speech recognition system with a control system including an embedded Raspberry Pi 4 and six servo motors. The MLFFNN algorithm trains the prosthetic hand to precisely understand pre-defined spoken instructions for certain motions, such as moving the wrist or opening individual fingers. This improves the prosthetic hand's ability to perform complex movements based on spoken instructions by allowing real-time processing of commands and smooth interaction between the user and the prosthetic hand. It processes them using the MLFFNN to speech classification relaying to the prosthetic hand motion control. All work was implemented using the last version of Python (13.12.0) (64-bit), a personal computer (11th Gen Intel(R) Core (TM) i3-1115G4 @ 3.00GHz) and a microphone. Figure 2 demonstrates the overall system, which includes both the training and testing stages [12].

3.1 Dataset

The dataset utilized for both training and testing purposes was acquired from two distinct sources. The initial Synthetic Speech Commands Data set (SSCD) was sourced from the internet.

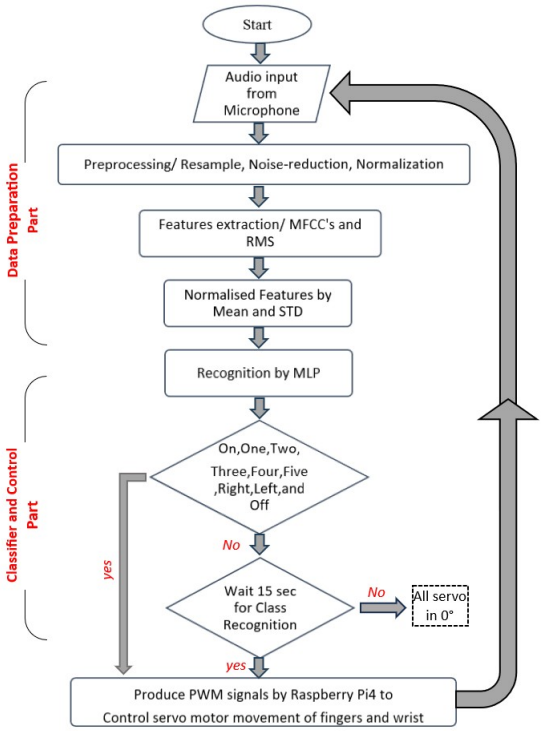


Figure 1. General flow chart diagram of present work.

The SSCD software is open-source and distributed under the Creative Commons BY 4.0 license [13]. Synthetic data generated by combining noise samples from AURORA-including airports, talk, automobile, exhibition, restaurant, street, subways, and trains with additional synthetic noise samples like ocean, white, brown, and pink. These noise samples were blended with voice using varying speech volume and offset and varying noise source and loudness using scripts called addnoise.py and addnoise2.py. Addnoise2.py was otherwise similar addnoise.py, except having lower voice volume and higher noise level. Every audio file was defined in WAV format, 16-bit, mono, with a 16000 Hz sampling rate, as one second long. It consists of 30 words and 41,849 utterances. The training incorporates these ambient sounds to enhance the system's ability to withstand and adapt to environmental noise. The dataset consists of nine classes, namely ön,one,two,three,four,five,right,left,änd öff.İt includes 30 words and 1914 utterances. Additionally, a second source of data is available, which comprises 200 audio files. These files were recorded by the researcher, with each file lasting one second and having a sampling rate of 16 kHz. The audio files are encoded in linear 16-bit single-channel format and cover the same nine classes as the original dataset. Two individuals are responsible for recording it. One individual was an adult woman who had undergone an amputation in her upper limb. The project aims to design and develop an advanced upper prosthetic limb with intelligence capabilities for

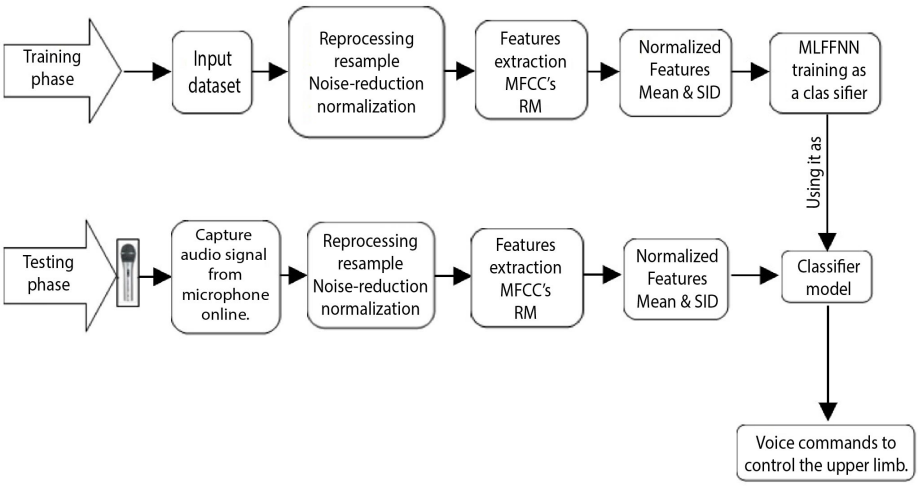
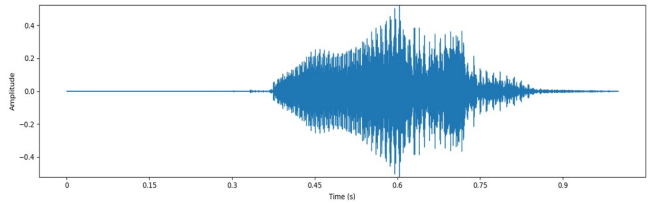
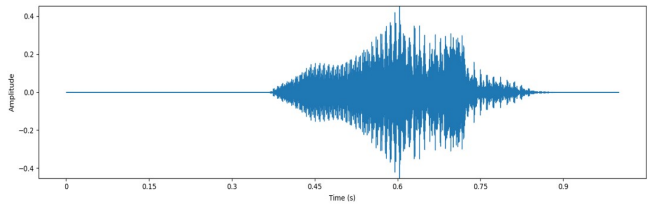


Figure 2. Proposed system model.

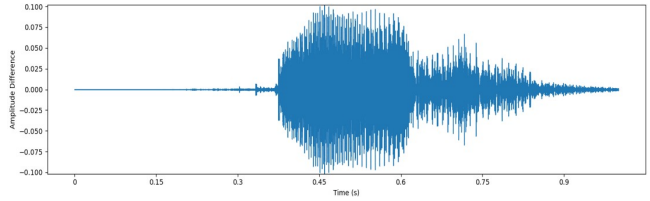
her utilization. The audio dataset for processing and training consists of 2114 voices in total. A vast and varied dataset of 2114 utterances allow the model to learn from different speech patterns and dialects, improving its generalization.



(a) Before noise reduction.



(b) After noise reduction.



(c) Difference between a and b

Figure 3. Audio signal before and after noise reduction.

3.2 Offline training model

The following steps are the proposed workflow for offline audio feature extraction, preprocessing, and building a neural network model for classification. Various Python libraries, such as Librosa and Keras, are used to perform these tasks efficiently.

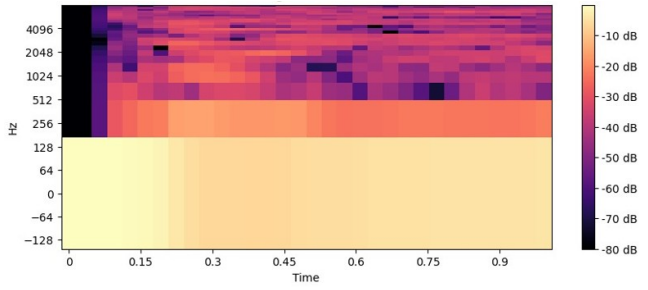
3.2.1 Load and preprocess the data

The desired sampling rate for the audio samples in this project is 16000 Hz. Systems typically employ 16000 Hz sampling to capture various speech frequencies and achieve optimal performance. The audio signals will undergo processing at a rate of 16000 samples per second. Additionally, each audio sample has a goal duration of 1 second. Short input segment guarantees coherence and enable standardized handling of the input data, and let the system collect temporal information and analyze it in real time. Audio signals are improved by implementing noise reduction techniques. The Python module 'noisereduce' employs a spectrum subtraction method. This algorithm utilizes spectral analysis to approximate the power spectrum of ambient noise. Then it eliminates it from the original signal in order to amplify desired speech or other auditory signals. Spectrum subtraction offers plenty of simplicity and efficiency to reduce background noise in audio transmissions. This approach offers, among other advantages, processing efficiency, frequency domain analysis capability, suitability for real-time applications, and dynamic noise estimation capability. The Short-Time Fourier Transform (STFT) converts an input audio signal from the time domain to the frequency domain by breaking down the sound into discrete frequency bands across time. Following Short-Time Fourier Transform (STFT) computation of the magnitude spectra for both the noisy input and estimated noise signals, subtracting the estimated/noise magnitude spectra from the corresponding noisy input magnitude spectra element by element yields spectral subtraction. The modified magnitude spectra then reverse their Short-Time Fourier Transform (STFT) operation to generate a denoised or filtered output in the time domain. As illustrated in Fig. 3, an audio signal example is presented both before and after undergoing noise reduction. Framing and windowing techniques are utilized to partition the audio signal into manageable segments for analysis [14]. The frame length is the number of samples contained in each analysis frame, while the hop length is the number of samples between consecutive frames. Frame extraction is a process that takes an audio signal as input and produces a matrix where each column represents an analysis frame. The audio stream is divided into overlapping frames using frame length and hop length. Each column represents successive samples from the audio source. Determining the quantity of frames taken from each audio source is essential in audio signal processing. This calculation entails examining signal length, frame length, and hop length. A sample rate of 16000 Hz and a signal duration of 1 second per audio sample were selected. A frame length of 1024 samples and a hop length of 512 samples were chosen. The signal length has been calculated to be 16000 samples. Afterward, the number of frames obtained from each audio signal is computed, reduced to 32 according to the provided settings. Hence, the resulting output array obtained from the process of extracting frames is characterized by dimensions of (2114, 32), indicating that thirty-two frames have been recovered from each audio signal in the dataset.

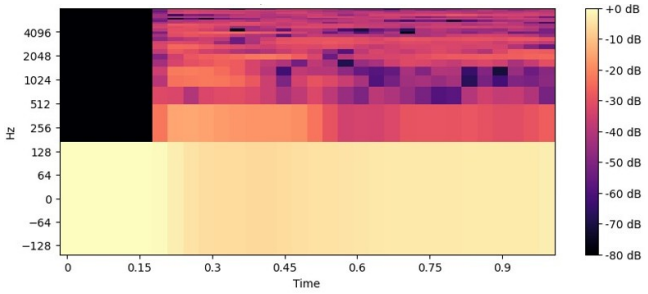
3.2.2 Features extraction

Extracted features iterate over all audio files in a dataset after frame extraction, load them using Librosa's load function by Python [5], and apply resampling if necessary to match the target sample rate(16 KHz). If an audio file is shorter than the target duration, it pads it with zeroes to precisely match its duration. If it is longer, it trims it accordingly. Then, features are extracted from each preprocessed audio sample using Python's librosa functions. These features provide valuable insights into various aspects of speech signals, such as: Mel-frequency cepstral coefficients (MFCCs) [15, 16] are widely used in speech signal processing to capture essential spectral characteristics by encoding key information about the spectral content of speech signals [17]. Because this encoding lets recognition systems focus on the necessary frequency components, speech recognition systems become more powerful, and background noise has less of an effect. Figure 4 shows the MFCCs of the audio signal both before

and after preprocessing. The Root Mean Square (RMS) [18] also calculates the overall energy levels in the speech signal since it makes it easier to detect changes or transitions in speech and hence distinguish between voiced and non-voiced parts.

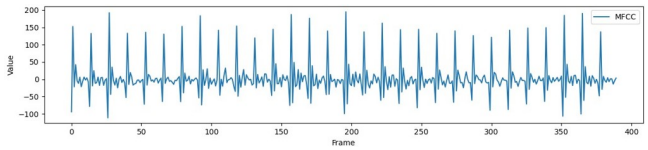


(a) Original MFCC's

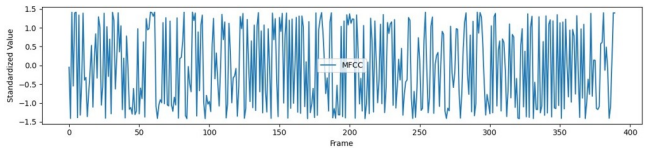


(b) Preprocessed MFCC's

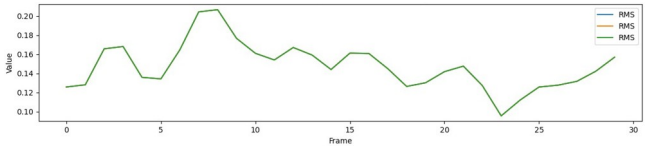
Figure 4. MFCC's audio signal data.



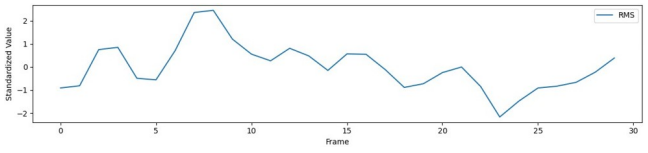
(a) MFCC feature (before standardization)



(b) Before standardization MFCC feature



(c) RMS feature (Before standardization)



(d) Standardization RMS feature)

Figure 5. Standardization for MFCC and RMS features.

The zero-crossing ratio (ZCR) measurement [18] helps distinguish between non-acoustic sounds or noise, which seem more from random zero crossings, and acoustic sounds, which are typified by regular zero crossings, by measuring how often a signal crosses the zero axis. Moreover, by providing helpful information about energy concentration throughout time within different frequency bands, the Temporal Centroid (TC) feature aids the capture of temporal dynamics, such as transitions or duration differences between spoken or audio syllables. Spectral flatness (SF) is the measure of the energy dispersion or concentration across multiple frequencies that makes it easier to distinguish between tonal sounds with high spectral concentration (speech) and loud noises with widely dispersed spectral energy. Lastly, Spectral Contrast (SC) emphasizes critical acoustic cues that aid in discriminating between various phonemes or linguistic units by highlighting differences between peaks and bottoms within specific frequency bands. These distinct features are subsequently concatenated into a unified feature matrix, facilitating comprehensive analysis and processing of speech signals for various applications in speech recognition and related fields.

3.2.3 Normalization by mean and standard deviation

Normalization is commonly conducted before the training stage. The purpose of normalizing the features is to standardize their scale, hence enhancing numerical stability, ensuring equal significance of features, and optimizing model performance [17]. The process of normalizing by mean and standard deviation is performed on the training data before training the model. It guarantees that all samples in the training set have a mean of zero and a variance of one across their characteristics.

Table 1. Label to index.

Class	Off	One	Two	Three	Four	Five	Left	Right	On
Index	0	1	2	3	4	5	6	7	8

Figure 5 displays the MFCC and RMS characteristics prior to and following standardization for a single audio sample. The process of normalizing by mean and standard deviation should be applied to characteristics in the following manner:
First, the mean (μ) of each feature is computed by summing up all values of that feature and dividing by the total number of samples (N), [17], Eq. 1:

$$\mu = (X_1 + X_2 + ... + X_n)/N \tag{1}$$

Next, the standard deviation (σ) is calculated for each feature using the formula, [17], Eq. 2:

$$\sigma = \sqrt{((x_1 - \mu)^2 + (x_2 - \mu)^2 + ... + (X_n - \mu)^2)/N} \tag{2}$$

This equation measures the spread of values within a feature. Finally, each sample in the dataset and for each feature is normalized using the following Eq. 3:

$$x_{normalized} = (x - \mu)/\sigma \tag{3}$$

Where "x"is the original value of a feature, "μ"is its mean value calculated in Eq. 1, and "σ"represents its standard deviation calculated in Eq. 2.

3.2.4 Prepare data for training

The filenames are used to extract a collection of distinct labels, which are then used to define all the potential classes in the dataset. A dictionary is established to associate each distinct label with an index for one-hot encoding, as depicted in Fig. 6 and Table 1. The audio sample labels are transformed into one-hot encoding using the np.eye function from the Python NumPy library. A binary matrix is generated, with each row representing a sample and each column representing a class label. The presence of a class in a corresponding sample is indicated by the value 1 in the matrix. The data is subsequently divided into training and testing sets via the scikit-learn function in Python. Through the specification of various test sizes (10%, 15%, and 20% of the data), random samples are chosen for testing, while the remaining samples (90%, 85%, and 80% of the data) are utilized for training.

```
Array ([1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.])
```

Figure 6. Labels one hot array.

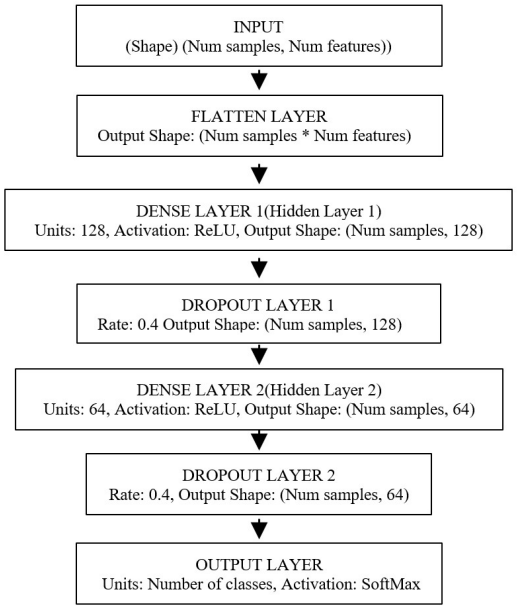


Figure 7. Labels one hot array.

3.2.5 Build MLFNN model

The MLFNN model is built using Kera's Sequential API, which enables the step-by-step construction of models [19], as illustrated in Fig. 7. The model starts with a Flatten layer, which transforms the input feature matrix into a one-dimensional array. This enables the subsequent dense layers to handle the data. The dense layers in this context employ Rectifier Linear Unit (ReLU) activation functions. They are characterized by complete connectivity, whereby each neuron is connected to every neuron in the preceding layer [20]. Dropout layers are incorporated following each dense layer to mitigate overfitting by randomly deactivating a portion of input units to zero during the training process [21]. The last dense layer consists of a number of units that is equal to the number of distinct labels in the dataset. It employs a softmax activation function to produce probability scores for each class [22]. This architecture facilitates efficient processing and categorization of the data.

3.2.6 Compile and train the model

Prior to training, it is necessary to assemble the model by specifying an optimizer, a loss function, and measurements. The optimizer is responsible for determining the manner in which the weights of the model are adjusted during the training process. The Adam optimizer is employed in this work [23]. Adam adjusts the learning rate for each network parameter, which speeds convergence and improves performance . The loss function quantifies the model's performance on individual training samples and directs the optimization process. The categorical cross-entropy loss function is employed in the context of multi-class classification problems [22]. Metrics are employed to assess the efficacy of the model. Accuracy is selected as the metric in this instance. The Keras fit function is utilized to train the model by supplying input features (X) train and associated labels (y) train, along with varying numbers of epochs and batch sizes. The model progressively modifies its weights by considering the training data in order to minimize the loss function throughout the training process. Once the training process is over, the final trained model is saved, which includes the structure of the neural network as well as its learned weights and biases. It enables the loading and utilization of the trained model at a later time to make predictions on new data.

3.3 Online prediction

The same steps are followed in the online prediction process as in the offline training process. The audio is captured from the microphone for 1 second and then resampled to a target sample rate of 16000 Hz using the librosa resample function in Python. Feature extraction involves pre-emphasis, noise reduction, zero padding, frame generation with matching parameters to offline training, and computation of MFCCs (13) and RMS features. Then, the extracted features are normalized using the mean and standard deviation. The trained model predicts class probabilities for all unique classes in Table 1. Finally, the np. argmax function in Python determines the class with the highest probability.

4. Result and discussion

It is worth noting that calculating and monitoring the validation accuracy is crucial for assessing how well MLFFNN performs on unseen data during training. It aids in preventing overfitting, optimizing hyperparameters, selecting models, and tracking progress throughout training to ensure better generalization capabilities. Many training sessions were conducted for the neural network using a different number of features and splitting the data to obtain optimized results in terms of accuracy and validation with minor complexity. Table 2 shows some optimized results obtained while training the MLFFNN for two hidden layers with a dropout of 40% and 32 samples for each input audio signal (2114 signal) and mfcc's 13. The best result is obtained using two features (mfcc's13 and rms), 10% test data for input data shape (2114,14) with an accuracy of 99.8% and validation of 97.6%. Figure 8 shows the accuracy of different models for training and validation. During the real-time experiment, the microphone was used to capture the audio signal for 1 second and then resampled to a target sample rate of 16000 Hz using the librosa resample function in Python. Feature extraction involves pre-emphasis, noise reduction, zero padding, frame generation with matching parameters to offline training, and computation of MFCCs (13) and RMS features. Then, the extracted features are normalized using mean and standard deviation.

Table 2. Optimized offline training results for prediction accuracy and validation accuracy.

No. of epochs	No. of features	Data test, %	Accuracy, %	Validation, %
100	8	20	95.0	86.8
200	8	10	98.0	86.2
100	6	20	97.0	89.7
200	6	10	97.7	90.0
100	3	20	98.0	93.5
200	3	10	99.0	90.7
100	2	10	99.8	97.6
200	2	20	98.8	95.6

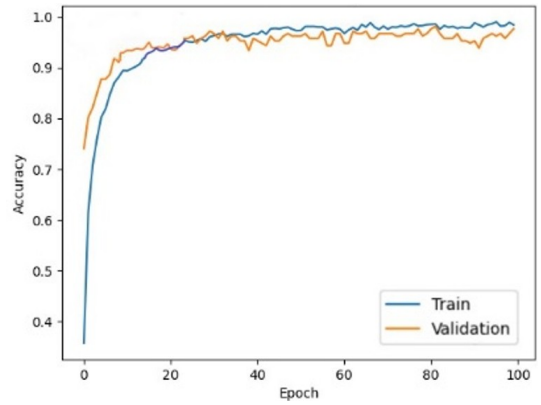
Table 3. Comparison with others' work.

Reference	Model	Offline classification	Experimental recognition	Validation accuracy
[24]	TDuf	88.30 %	88.00 %	—
[25]	RBFNN	99.175 %	94.80 %	—
[6]	CNN	91.00 %	—	91.00 %
Present work	MLFFNN	99.40 %	96.05 %	97.60 %

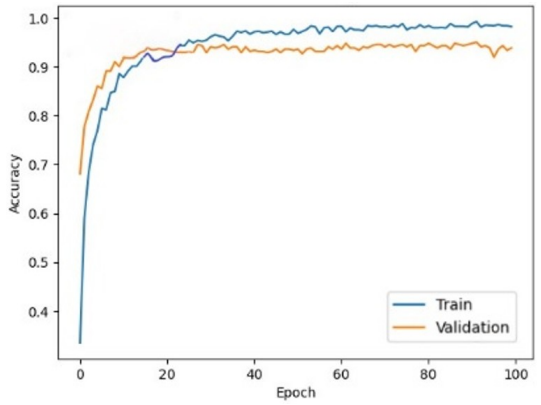
Table 4. Real-time recognition accuracy.

Class	Correct prediction	Wrong prediction	Recognition %, $(Correct/40)*100$
On	37	3	92.25
Off	38	2	95.00
One	38	2	95.00
Tow	40	0	100.0
Three	39	1	97.50
Four	40	0	100.0
Five	38	2	95.00
Right	39	1	97.50
left	37	3	92.25
Total accuracy	—	—	96.05

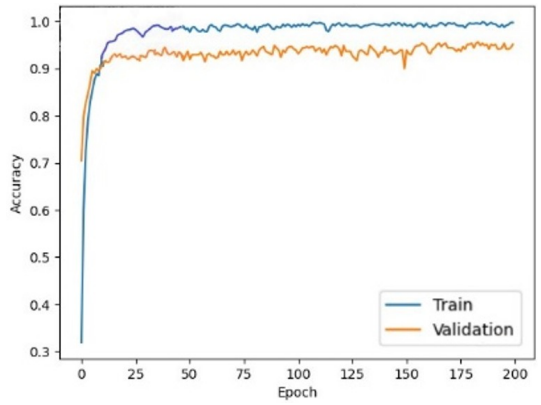
The trained model predicts class probabilities for all unique classes. The pronunciation of the training words (on, off, one, two, three, four, five, right, and left) was evaluated within a 1-second interval. The same subject repeated each word 40 times in a room environment with a noise level of 65 dB, as determined by a sound level meter. The findings collected showed a recognition accuracy of 96.05%, as depicted in Table 4 . Performing a comparative analysis in Table 3 is essential to compare the current findings with those of other researchers, some of whom did not explicitly mention the correctness of their validation. The comparison table exhibits offline classification, experimental recognition, and validation accuracy for many models. In offline classification and experimental recognition with 99.4% and 96.05% accuracy, MLFFNN exceeded TDuf, TMNN, RBFNN, and CNN correspondingly.



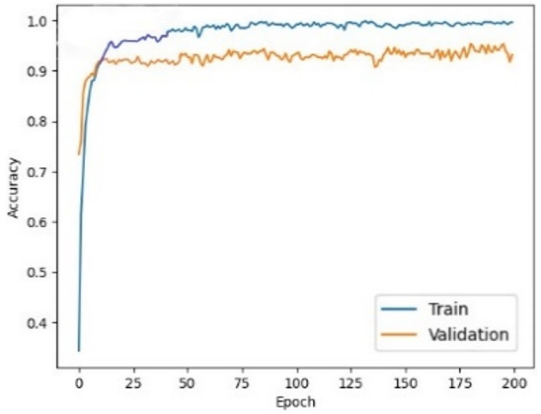
(a) 97.6% Validation accuracy with 100 epochs.



(b) 93.5% Validation accuracy with 100 epochs.



(c) 90.7% Validation accuracy with 200 epochs.



(d) 95.6% Validation accuracy with 200 epochs.

Figure 8. MLFFNN model accuracy for training and validation models

With a high validation accuracy of 97.6%, the MLFFNN model proved to be more effective in handling audio data than the mentioned models [8, 12, 24]. This comparison is crucial as it offers a valuable understanding of the model's capacity to recognize novel input accurately.

5. Conclusions

This work employs an MLFFNN model to perform automatic speech recognition in nine specific categories: on, off, one, two, three, four, five, right, and left. A dataset including 2114 audio signals was utilized, with each signal being recorded in distinct background noise conditions. The signal preprocessing procedure encompassed various stages, such as pre-emphasis, noise reduction, zero-padding, frame generation, and feature extraction, which entailed the utilization of MFCC's 13 and RMS computations. The process of feature normalization was carried out by utilizing the mean and standard deviation data. The neural network completed more than 150 training cycles, using various combinations of features taken from the audio signals. These characteristics ranged from eight to two, including MFCC and RMS. Several neural network models have undergone testing using varying bit sizes, data partitioning methods, and epoch numbers. Although certain experiments demonstrated encouraging outcomes, the actual time it took to recognize in real-time varied from 141 milliseconds for the 8-feature set to 40 milliseconds for the 2-feature set. Selecting a model that achieves a fine equilibrium between superior accuracy and validation in real-time recognition was of utmost importance. The offline model demonstrated performance with an accuracy rate of 99.4%, a validation accuracy of 97.6%, and a real-time recognition accuracy of 96.05%. The trained model will be utilized to pass recognition classes to a decoder circuit. This circuit will then be responsible for controlling servo motors in prosthetic hands that have five fingers and a wrist. This will enable the prosthetic hands to do movements that imitate spoken words. In future work, in order to enhance the promptness and precision of responses provided to consumers in real-time. The number of motions of the prosthetic hand may be enhanced using different data, and a more durable and lighter prosthetic limb can also be manufactured for practical usage by the patient. It is feasible to utilize a higher-performance CPU in the prosthetic hand by constructing a compact printed circuit board (PCB) that surpasses the capabilities of the Raspberry Pi 4 processor.

Authors' contribution

All authors contributed equally to the preparation of this article.

Declaration of competing interest

The authors declare no conflicts of interest.

Funding source

This study didn't receive any specific funds.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] S. Varshney, A. Suman, G. Varshney, and P. Suman, "Applications of ai in prosthetics: A critical analysis," *Computology: Journal of Applied Computer Science and Intelligent Technologies*, vol. 2, pp. 23–32, 2022. [Online]. Available: https://doi.org/10.17492/computology.v2i1.2203

[2] J. P. Ángel-López and N. Arzola de la Peña, "Voice controlled prosthetic hand with predefined grasps and movements," in *VII Latin American Congress on Biomedical Engineering CLAIB 2016, Bucaramanga, Santander, Colombia, October 26th -28th, 2016*, I. Torres, J. Bustamante, and D. A. Sierra, Eds. Singapore: Springer Singapore, 2017, pp. 520–523. [Online]. Available: https://doi.org/10.1007/978-981-10-4086-3_131

[3] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef, "Comparative study of automatic speech recognition techniques," *IET Signal Processing*, vol. 7, no. 1, pp. 25–46, 2013. [Online]. Available: https://doi.org/10.1049/iet-spr.2012.0151

[4] J. Hussein, "Speech recognition based on statistical features," *Computational Semantics*, 2022. [Online]. Available: https://doi.org/10.5772/intechopen.104671

[5] L. F. Tshimanga, F. Del Pup, M. Corbetta, and M. Atzori, "An overview of open source deep learning-based libraries for neuroscience," *Applied Sciences*, vol. 13, no. 9, 2023. [Online]. Available: https://doi.org/10.3390/app13095472

[6] M. Jafarzadeh and Y. Tadesse, "Convolutional neural networks for speech controlled prosthetic hands," *2019 First International Conference on Transdisciplinary AI (TransAI)*, pp. 35–42, 2019. [Online]. Available: https://doi.org/10.1109/TransAI46475.2019.00014

[7] T. Triwiyanto, S. D. Musvika, S. Luthfiyah, E. Yulianto, A. M. Maghfiroh, L. Lusiana, and I. D. M. Wirayuda, "Upper limb exoskeleton using voice control based on embedded machine learning on raspberry pi," in *Proceedings of the 2nd International Conference on Electronics, Biomedical Engineering, and Health Informatics*, T. Triwiyanto, A. Rizal, and W. Caesarendra, Eds. Singapore: Springer Nature Singapore, 2022, pp. 557–569. [Online]. Available: https://doi.org/10.1007/978-981-19-1804-9_42

[8] D. W. Thiang, "Limited speech recognition for controlling movement of mobile robot implemented on atmega162 microcontroller," in *2009 International Conference on Computer and Automation Engineering*, 2009, pp. 347–350. [Online]. Available: https://doi.org/10.1109/ICCAE.2009.26

[9] A. Hussein Shatti Alisawi and A. Qasim Jumaah Althahab, "Improve the performance of voice-excited lpc vocoder in speech coding application," *Al-Qadisiyah Journal for Engineering Sciences*, vol. 8, no. 4, pp. 558–568, 2015. [Online]. Available: https://qjes.qu.edu.iq/article_107872.html

[10] H. Herath, N. Nishshanka, P. Madhumali, and S. Gunawardena, "Voice control system for upper limb rehabilitation robots using machine learning," in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 729–734. [Online]. Available: https://doi.org/10.1109/WF-IoT51360.2021.9595827

[11] M. H. Asyali, M. Yilmaz, and M. Tokmakçi, "Design and implementation of a voice-controlled prosthetic hand," *Turkish Journal of Electrical Engineering and Computer Sciences*, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:16860496

[12] W. Ghai and N. Singh, "Literature review on automatic speech recognition," *International Journal of Computer Applications*, vol. 41, no. 8, pp. 42–50, March 2012. [Online]. Available: https://doi.org/10.5120/5565-7646

[13] A. Kuzdeuov, R. Gilmullin, B. Khakimov, and H. A. Varol, "An open-source tatar speech commands dataset for iot and robotics applications," in *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, 2024, pp. 1–5. [Online]. Available: https://doi.org/10.1109/IECON55916.2024.10905876

[14] L. Yuan and Z. Chen, "A novel statistical model for speech recognition and pos tagging," in *2006 IEEE International Conference on Video and Signal Based Surveillance*, 2006, pp. 61–61. [Online]. Available: https://doi.org/10.1109/AVSS.2006.9

[15] H. Ibrahim and A. Varol, "A study on automatic speech recognition systems," in *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, 2020, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ISDFS49300.2020.9116286

[16] A. Ahmed Abed and A. A. Jasim, "Design and implementation of a wireless voice-controlled mobile robot," *Al-Qadisiyah Journal for Engineering Sciences*, vol. 9, no. 2, pp. 135–147, 2016. [Online]. Available: https://qjes.qu.edu.iq/article_179828.html

[17] H. A. Abdulmohsin, H. B. Abdul wahab, and A. M. J. Abdul hossen, "A new proposed statistical feature extraction method in speech emotion recognition," *Computers Electrical Engineering*, vol. 93, p. 107172, 2021. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2021.107172

[18] C. Panagiotakis and G. Tziritas, "A speech/music discriminator based on rms and zero-crossings," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 155–166, 2005. [Online]. Available: https://doi.org/10.1109/TMM.2004.840604

[19] D. Nagajyothi and P. Siddaiah, "Speech recognition using convolutional neural networks," *International Journal of Engineering and Technology*, vol. 7, no. 4.6, p. 133–137, Sep. 2018. [Online]. Available: https://www.sciencepubco.com/index.php/IJET/article/view/20449

[20] "Ieee signal processing society," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. C3–C3, 2017. [Online]. Available: https://doi.org/10.1109/TASLP.2017.2684725

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, Jan. 2014.

[22] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *Computation and Language*, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1602.02410

[23] J. Kacprzyk, O. Kaynak, W. Pedrycz, M. M. Polycarpou, and J. Wang, *Lecture Notes in Networks and Systems Volume 430*. Springer Cham, 2016. [Online]. Available: https://link.springer.com/bookseries/15179

[24] K. Gundogdu, S. Bayrakdar, and I. Yucedag, "Developing and modeling of voice control system for prosthetic robot arm in medical systems," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 2, pp. 198–205, 2018. [Online]. Available: https://doi.org/10.1016/j.jksuci.2017.04.005

[25] J. Salman, T. R. Saeed, and A. H. Ali, "Design and implementation of a spoken letter recognition system for controlling the upper prosthetics," in *2018 Third Scientific Conference of Electrical Engineering (SCEE)*, 2018, pp. 174–179. [Online]. Available: https://doi.org/10.1109/SCEE.2018.8684098

How to cite this article:
Ahmed M. A. Ahmed, Mithaq N. Raheema, and Jabbar S. Hussein. (2025). 'Design and implementation of upper limb controlled by human voice and artificial intelligence', Al-Qadisiyah Journal for Engineering Sciences, 18(3), pp. 320-326. https://doi.org/10.30772/qjes.2024.149892.1241