MULTIDISCIPLINARY SCIENCE JOURNAL

RESEARCH ARTICLE
Published Online: August 9, 2023
https://doi.org/10.31893/multiscience.2024021

Breaking the silence: Innovation in Wake word activation

Sagar Shinde[a] ✉ | **Akash Kathole**[b] | **Lalitkumar Wadhwa**[a] | **Armaan Suhel Shaikh**[a]

[a]Department of Electronics & Telecommunication, Dr. D. Y. Patil Institute of Technology, India.
[b]Department of Artificial Intelligence, Nutan College of Engineering and Research, India.

Abstract Trigger term, also referred to as a heat phrase or activation phrase, is a particular expression or wording that activates a voice-controlled system like a virtual aide when spoken. These trigger terms are widely employed in various devices, including mobile phones, intelligent domestic appliances, and automobiles. The precise identification of trigger terms is crucial to guarantee that the voice-controlled system only responds when the trigger term is uttered, disregarding other sounds. Numerous methods, such as regulation-based approaches and machine learning algorithms like support vector machines, concealed Markov models, and profound neural networks, have been proposed for identifying trigger terms. Elements like ambient noise, pronunciation variations, and accents may impact the accuracy of trigger term identification, and researchers are continuously exploring ways to enhance the resilience of these systems. As the utilization of voice-controlled systems continues to grow, the development of reliable and accurate methods for detecting trigger terms is increasingly imperative to ensure the seamless and efficient operation of these systems.

Keywords: hotword, voice-activated system, virtual assistant, rule-based detection, hidden Markov models, deep neural networks

1. Introduction

The wake word, also known as the hotword or trigger word, is a fundamental element in any voice-activated system, such as AI assistants, and plays a critical role in initiating the system's response to the user's requests. Developing an accurate and robust wake word detection model is a challenging task that requires collecting diverse and high-quality data and training a precise model that can identify the wake word accurately under various conditions (Shinde et al 2023).

This article explores various methods for identifying wake words, such as rule-driven and artificial intelligence (AI) algorithms, and explores the difficulties and possibilities linked with each technique. The growing adoption of voice-activated systems in mobile devices, home automation devices, and cars has amplified the need for precise and dependable wake word identification techniques. The wake word models must be trained on a diverse dataset that encompasses different languages, accents, and speaking styles to ensure that the system works seamlessly across different regions and demographics.

Moreover, the increasing usage of voice-activated systems raises concerns about privacy and security. It is imperative to design wake word detection algorithms in a way that ensures activation only when the wake word is spoken, while also preserving user privacy. Addressing these privacy and security considerations is critical for instilling user confidence in voice-activated systems.

This paper provides an overview of the wake word detection problem, highlights the latest developments and trends in wake word detection research, and proposes future directions for research in this field. Chatbots are designed to receive audio input from users and provide responses based on their programmed functionality. For instance, consider the study by Terzopoulos and Satratzemi (2020), which explores innovative wake word detection techniques in the context of smart home devices. Furthermore, we discuss the implications of wake word detection in chatbots, such as Google Assistant. Google Assistant is a prime example of a chatbot that starts recording audio input as soon as it detects the wake word. The recorded audio is then sent to the cloud for speech synthesis to generate a response. The use of wake word prevents unnecessary cloud processing of irrelevant audio fragments, thus saving computational power. Essentially, wake word acts as a trigger for speech synthesis, indicating the point at which to start processing audio input (Quantum 2020).

2. Background and Literature Review

The notion of stimulus terms in voice-operated technology traces its origins to the early stages of computing, when researchers commenced exploring the potential of voice-operated technology. Throughout the 1960s and 1970s, the initial voice-operated systems were developed, utilizing basic commands such as "hi", "hear," or "commence" to initiate interactions

 Received: May 26, 2023 | Accepted: June 26, 2023

with the devices. Over time, significant advancements in natural language processing (NLP) and machine learning (ML) algorithms have led to the evolution of stimulus terms into more sophisticated forms.

The lexicon employed for simulation has grown progressively advanced, as contemporary electronic assistants like Siri, Alexa, and Google Assistant exhibit the capacity to comprehend and reply to intricate sentences and phrases. These advancements have been facilitated by enhancements in natural language processing (NLP) and machine learning (ML) algorithms.

The evolution of stimulation terminology has been driven by the necessity to render voice-activated technology more convenient and accessible, empowering individuals to engage with their devices without any tangible contact. This has streamlined tasks such as playing music, setting reminders, and managing intelligent household devices.

Despite these advancements, the detection of trigger words in noisy environments remains a challenge. Researchers have therefore developed advanced ML algorithms that can recognize and eliminate background noise, enabling trigger words to be detected even in difficult environments (Kathole et al 2023).

Several studies have investigated the use of wake words and trigger words in voice-activated systems. One study proposed a novel approach to wake word detection using deep neural networks, achieving remarkably high accuracy rates even in noisy environments. Additionally, another study found that incorporating conversational trigger words significantly improved the overall user experience, making interactions with voice-activated systems more natural and intuitive (Holy 2018).

Privacy and security are also crucial considerations in wake word or trigger word technology. Researchers have proposed privacy-preserving wake word detection systems that use encryption and homomorphic filtering to detect wake words without exposing user speech data to potential eavesdroppers.

Furthermore, researchers have examined the potential implications of unintended wake words in voice-activated systems, highlighting the importance of designing wake word systems with strong privacy and security protections.

3. Proposed Methodology

Figure 1 illustrates the block diagram for the proposed system." as the first sentence in Proposed Methodology section.

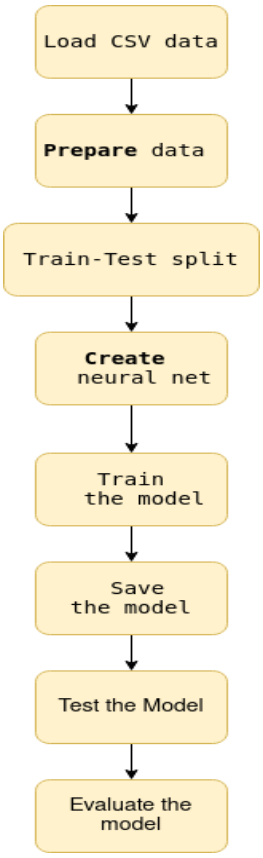

Figure 1 Block diagram of proposed system.

At a high level, the model first loads the CSV data, which is then prepared for training by separating the features and class labels. The data is then split into training and testing sets. Next, a neural network model is created using the Keras API, with three dense layers, each with 256 units and using the 'relu' activation function. The final layer has two units with the

'softmax' activation function. The model is compiled using the categorical cross-entropy loss function, the Adam optimizer, and accuracy as the metric. The model is then trained using the training data, with a total of 1000 epochs. Once training is complete, the model is saved to a file.

The model is then evaluated using the testing data, and a classification report is printed to the console. Table 1 lists the datasets used for the detection of wake words, along with their respective data collection techniques.

Table 1 Various datasets used for wake word detection and their data collection techniques.

Dataset Name	Data Collection Technique
VoxCeleb	Collected from public videos on YouTube
Google Speech Commands v1 & v2	Collected from native English speakers using scripts
Mozilla Common Voice and Speech Commands	Crowdsourced through community, surveys, etc.
Various datasets	Collected from various sources such as research papers
Google Speech Commands	Collected from native English speakers using scripts
Freesound	Collected from open-source audio sharing platform
Environmental Sounds Classification	Collected from various sources such as research papers
Data collected from crowdsourcing	Collected manually by users through a mobile app
Data collected from events	Collected through crowdsourcing platforms like Amazon Mechanical Turk
Referral-based data collection	Collected manually from events such as marriage, birthday party, etc. Collected by offering incentives and referral programs to individuals to collect wake word data

The first dataset mentioned is VoxCeleb (VoxCeleb 2023), which is collected from public videos on YouTube. The data collection technique involves using publicly available videos to extract audio samples from celebrity interviews and speeches. The next dataset is Google Speech Commands v1 and v2, which is collected from native English speakers using scripts. This dataset involves collecting spoken words from a standardized script read by participants (TensorFlow 2023). The Mozilla Common Voice and SpeechCommands datasets are crowdsourced through community, surveys, and other means. These datasets involve collecting speech data from a diverse set of individuals, including those with different accents, languages, and speech styles (Mozilla 2023). Various datasets are collected from various sources, such as research papers, to provide a diverse range of speech samples for wake word detection. For example, the Environmental Sounds Classification dataset contains various environmental sounds collected from research papers. Google Speech Commands is another dataset collected from native English speakers using scripts, similar to the Google Speech Commands v1 and v2 datasets (TensorFlow 2023). The Freesound dataset is collected from an open-source audio sharing platform, providing a wide range of audio samples for wake word detection.

Some datasets are collected manually through mobile apps, while others are collected through crowdsourcing platforms such as Amazon Mechanical Turk. Additionally, data can be collected from events like marriage or birthday parties. When gathering audio data for detecting the activation phrase, it is crucial to ensure that the recorded data is of excellent quality and has minimal background noise. The presence of ambient noise can significantly impact the accuracy of the activation phrase detection model and make it difficult to differentiate the activation phrase from other sounds.

To ensure that the documented information possesses minimal disturbance, it is recommended to employ software utilities such as Audacity for noise diminishment. Audacity is a costless and unbarred audio modification software that can aid in eradicating undesired interference from the documented audio. Figure 2 shows the user interface of Audacity software for audio modification.

To execute noise decrement in Audacity, commence by importing the audio file into the software. Then, opt for a segment of the audio that solely encompasses disturbance without any verbal communication.

This section is known as the noise profile, which is utilized to estimate the noise characteristics throughout the entire audio file.

After choosing the desired sound profile, proceed to the Effect tab and select the option for reducing noise. In the dialog box for noise reduction, modify the configurations to match your preferences, including the level of noise reduction and sensitivity. Ultimately, implement the noise reduction impact on the audio file to decrease ambient noise and improve the intelligibility of the recorded information.

Choose the window, pick the sound profile, and modify the noise reduction intensity to eliminate the undesirable sound from the audio. It is recommended to perform this process on each audio file before using it to train the wake word detection model. This ensures that the model is trained on clean audio data, and any background noise is minimized.

We have developed a wake word recognition system using TensorFlow, which exhibits exceptional precision of 98% on the test dataset. Furthermore, the deep learning model can be trained on any wake word of your choosing.

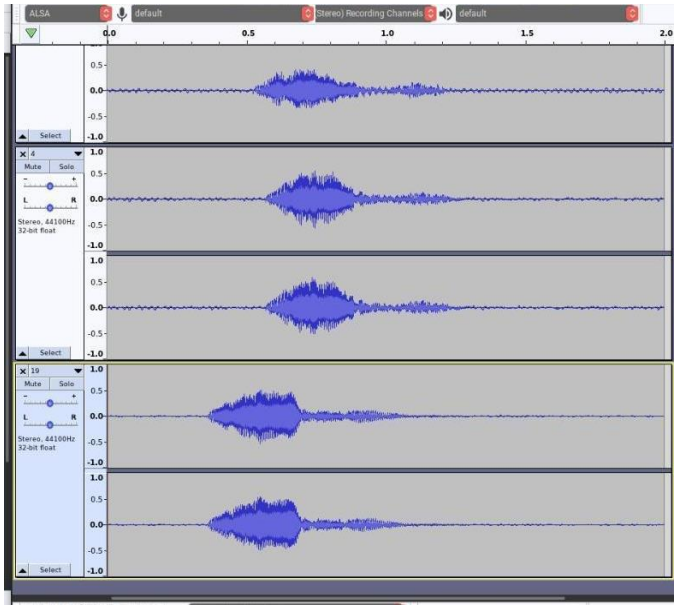

Figure 2 Audio modification in Audacity.

3.1 Step I

The Wake word detection model is created using TensorFlow. The first step in the process is to prepare the data by recording the Wake Word and background sounds. This is done using the file PreparingData.py, which contains two functions: record_audio_and_save() and record_background_sound. The recording process of wake word is shown in Figure 3.

record_audio_and_save() function records a 2-second audio of the user saying the Wake Word. The function requires two parameters: n_times and save_path. n_times determines the number of times the function will capture the Wake Word, while save_path indicates the location where the resulting .wav files will be saved.

The function record_background_sound() captures a 2-second audio sample of the surrounding noises, as shown in Figure 4. It takes n_times and save_path as its parameters. n_times specifies how many times the function should record the background sounds, and save_path denotes the directory for storing the resulting .wav files. It is crucial to note that the user must not utter the Wake Word during the background sound recording.

The user is prompted to initiate the recording of the Wake Word by pressing Enter. The function will execute n_times (default value is 50) and record the Wake Word each time it is spoken by the user. Following each recording, the individual is requested to hit the Enter key to advance to the subsequent one.

Likewise, the individual is encouraged to initiate the recording of ambient noises by pressing Enter. The operation will automatically seize the ambient sounds n_times (with a default value of 50) without necessitating any user intervention.

Once the data is recorded, the subsequent step involves training the Deep Learning model using TensorFlow with the recorded Wake Word and background sounds.

For recorded dataset:

1. To initiate recording the trigger phrase, press the "Enter" key. To continue recording or cease recording, press any alternative key or use "ctrl + C" respectively. This approach presents several advantages:
2. User autonomy: Users have the freedom to start and stop the recording at their convenience. They can begin recording by pressing "Enter" when prepared and continue recording as needed by pressing any key. Alternatively, they can halt the recording instantly by using "ctrl + C". This grants users greater flexibility and control over the recording process.
3. Batch recording: Enabling users to record multiple samples consecutively by pressing any key streamlines the batch recording procedure. It eliminates the need for manual initiation of each recording individually, ensuring efficiency.
4. Error handling: If users need to terminate the recording for any reason, they can easily do so by utilizing "ctrl + C". This action cleanly terminates the recording process without causing errors or leaving incomplete recordings.
5. Consistency: By requesting users to press a key to continue recording, the program ensures that each sample maintains a uniform duration and is recorded under similar conditions. This promotes improved consistency and quality of the recorded samples, which is crucial for training a dependable wake word detection system.

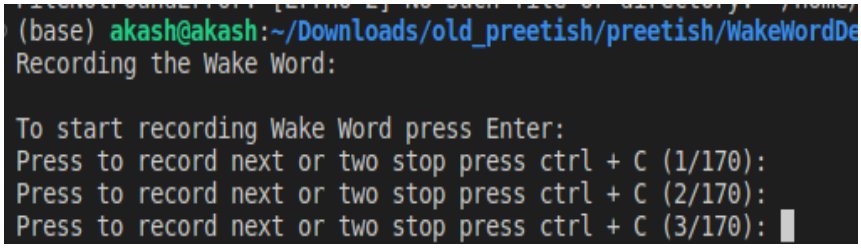

Figure 3 Recorded dataset.

For background dataset:

Script is designed to train a wake word detection system by capturing background sounds that might be present during use of the system, in order to improve the accuracy and reliability of the system.

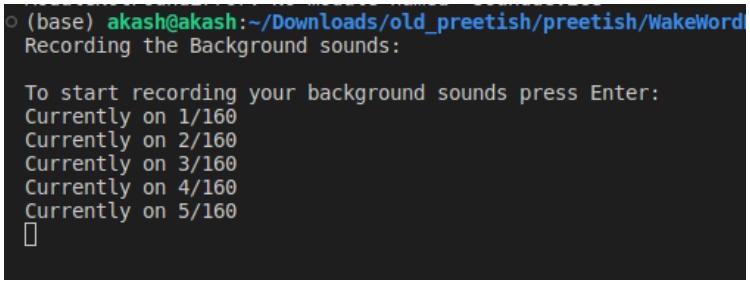

Figure 4 Background dataset.

3.2 Step II

Loading and pre-processing audio files for training a Wake word detection model. The code imports necessary libraries including os, librosa, matplotlib, numpy and pandas. The script then loads a sample audio file, visualizes its wave form and MFCC features using the librosa library. The same process is applied to all the audio files in two directories- "background_sound" which contains sounds that do not contain wake words, and "audio_data" which contains sounds that do contain wake words. For each file, the script loads the file, applies MFCC, processes it and stores it in a list along with the class label (0 or 1). The script then creates a pandas dataframe from the list and saves it as a CSV file using the to_pickle method. This CSV file can be used for training the Wake word detection model in the future. The accuracy of the model on test data is reported to be 98%.

Result of pre-processing in form of wave:

The form of an auditory indication is a graphical depiction of the alterations in atmospheric force over duration that generate auditory vibrations. In electronic audio, auditory vibrations are gauged at consistent intervals, and each measurement is denoted by a numeric quantity that corresponds to the magnitude of the auditory vibration at that particular moment.

The parallel axis of a signal representation symbolizes duration, whereas the perpendicular axis symbolizes the magnitude of the auditory vibration. When you execute an audio document and scrutinize its signal representation, you observe a chart that illustrates how the magnitude of the auditory vibration fluctuates over duration. Generally, the signal representation will alternate between affirmative and opposing quantities, as the auditory vibration transitions between compressions (where the atmospheric force is higher than normal) and rarefactions (where the atmospheric force is lower than normal).

The waveform can be valuable for scrutinizing audio signals, as it enables you to visualize alterations in volume and identify specific characteristics of the sound. For instance, you can utilize the waveform to detect periods of silence or discern when a specific instrument or vocal line enters or exits the mix.

The waveform is a visual depiction of the sound wave. It illustrates the sound's magnitude over time. The magnitude refers to the sound's volume, while time signifies the sound's duration. Regarding wake word detection, the waveform plays a vital role in identifying the wake word. The wake word is a specific word or phrase that the device is programmed to recognize. When the device detects the wake word, it will trigger a specific action, such as turning on the device or starting a voice assistant.

The importance of the waveform in wake word detection lies in its ability to distinguish the wake word based on its distinct frequency and amplitude properties. For instance, the wake word "Alexa" exhibits specific frequency and amplitude characteristics that set it apart from other words and phrases (Holy 2018).

Figure 5 demonstrates an example of a waveform, represented on a graph with the y-axis representing the amplitude and the x-axis depicting time. The wake word "Alexa" is identifiable as the sharp peak in the waveform, which signifies its unique acoustic signature. This distinctive waveform pattern aids in accurately detecting and recognizing the wake word "Alexa" within audio data.

The waveform is an important part of the wake word detection process. By analyzing the waveform, the device can identify the wake word and trigger the appropriate action.

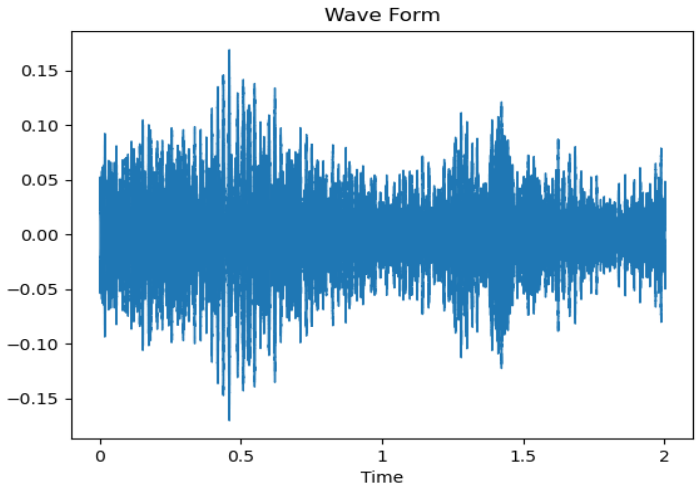

Figure 5 Graphical depiction of the alterations in atmospheric force over duration.

- The magnitude of the waveform is crucial because it dictates the volume of the sound. The wake word must possess a sufficient magnitude to be detected by the device.
- The rate of the waveform is also significant because it dictates the tone of the sound. The wake word must have a particular rate to be discerned from other words and phrases.
- The length of the waveform is also critical because it dictates how long the wake word must be spoken to be detected. The wake word must be spoken for a specific duration for the device to recognize it.

3.3 Step III: Train model

The model is built using a Sequential model architecture with dense layers, activation functions, and dropout layers. The model is trained on audio data that has been preprocessed and prepared for training. The data is split into training and testing datasets using the train_test_split function. The accuracy of the model is evaluated by calculating the loss and metrics such as accuracy using the evaluate function. The model is then saved to a file for future use. Finally, the performance of the model is evaluated using confusion matrix and classification report metrics.

The model works by taking preprocessed audio data as input and processing it through a series of dense layers with activation functions such as relu and dropout layers. The output of the model is a softmax layer with two output nodes representing whether the audio contains a Wake Word or not. The model is trained using the categorical_crossentropy loss function and the adam optimizer. During the evaluation phase, the model predicts the output class of the input audio data by using the argmax function on the model's output probabilities. The performance of the model is evaluated using confusion matrix and classification report metrics to measure the accuracy of the model. The model can be saved to a file for future use.

Description for creating, training, and evaluating a Deep Learning model for Wake Word Detection (WWD) using TensorFlow.

1. It starts with importing required libraries such as NumPy, Pandas, train_test_split from sklearn.model_selection, to_categorical, Sequential, Dense, Activation, Dropout from tensorflow.keras, confusion_matrix, classification_report from sklearn.metrics and plot_confusion_matrix.
2. Then it loads the saved CSV file containing audio data using Pandas read_pickle() function and stores the 'feature' values into a NumPy array X and 'class_label' values into a NumPy array y.
3. It then concatenates all values of 'feature' along the axis=0 and reshapes it into a 2D NumPy array of shape (len(X), 40), to prepare the data for training.
4. It then one-hot encodes the 'class_label' values using to_categorical() function.
5. After that, it splits the data into training and testing sets using train_test_split() function from sklearn.model_selection.

6. It then creates a Sequential model using Keras API, with 3 layers, having 256 units each for the first two layers and 2 units in the output layer, with softmax activation.
7. It compiles the model with 'categorical_crossentropy' loss function, 'adam' optimizer and 'accuracy' metrics.
8. It trains the model using fit() method, with X_train and y_train as input, for 1000 epochs, and saves the trained model in the directory "saved_model/WWD.h5".
9. It evaluates the model on the test set using evaluate() method and prints the model's accuracy score on the test set.
10. Finally, it evaluates the model's performance using classification_report() and confusion_matrix() functions and plots the confusion matrix using plot_confusion_matrix() function, to visualize the performance of the model.

3.4 *Step IV: Steps of Prediction*

1. Import necessary libraries: Import the required libraries such as sounddevice, scipy.io.wavfile, librosa, numpy, and tensorflow.keras.models.
2. Define constants: Define the required constants such as sample rate, duration of recording, filename, and class names for the predicted results.
3. Load the saved model: Load the saved model that was previously trained using machine learning algorithms.
4. Begin prediction: Start the prediction process by continuously recording the live sound and giving it to the loaded model for prediction.
5. Record sound: Record sound for a specified duration using sounddevice library and write the recorded sound to a file using scipy.io.wavfile.
6. Process audio data: Load the recorded sound file using librosa library and extract features from the sound file using MFCC (Mel-Frequency Cepstral Coefficients).
7. Preprocess MFCC data: Preprocess the extracted MFCC features by calculating the mean of each feature and transpose the matrix.
8. Make predictions: Use the loaded model to predict whether the wake word has been detected or not. If the predicted probability is greater than the threshold, then the wake word is detected, and the corresponding message is displayed.
9. Print results: Print the results of the prediction along with the confidence score of the predicted result.
10. Repeat: Repeat steps 5 to 9 for continuous prediction of the live sound recording.
11. End prediction: End the prediction process after completing the required number of predictions or on user interruption.

4. Experimental Result

The output in Figure 6 shows the result of running a wake word detection model, likely using a machine learning algorithm. The model is detecting whether a wake word has been spoken, and reports its confidence level in the prediction.

The first line shows that the model detected a wake word with high confidence, with a confidence score of 0.99966896. The next three lines show that the wake word was not detected, with decreasing confidence scores. The confidence scores are 1.0, 0.6970928, and 1.0 respectively. The following line shows that the wake word was detected with high confidence again, with a confidence score of 1.0. The final line reports that the wake word was not detected, but the confidence score is still high at 1.0.

The "Say Now" lines are likely indicating that the model is waiting for a spoken input to detect whether the wake word has been spoken or not. The numbers after "1/1" may indicate the batch size used for the wake word detection.

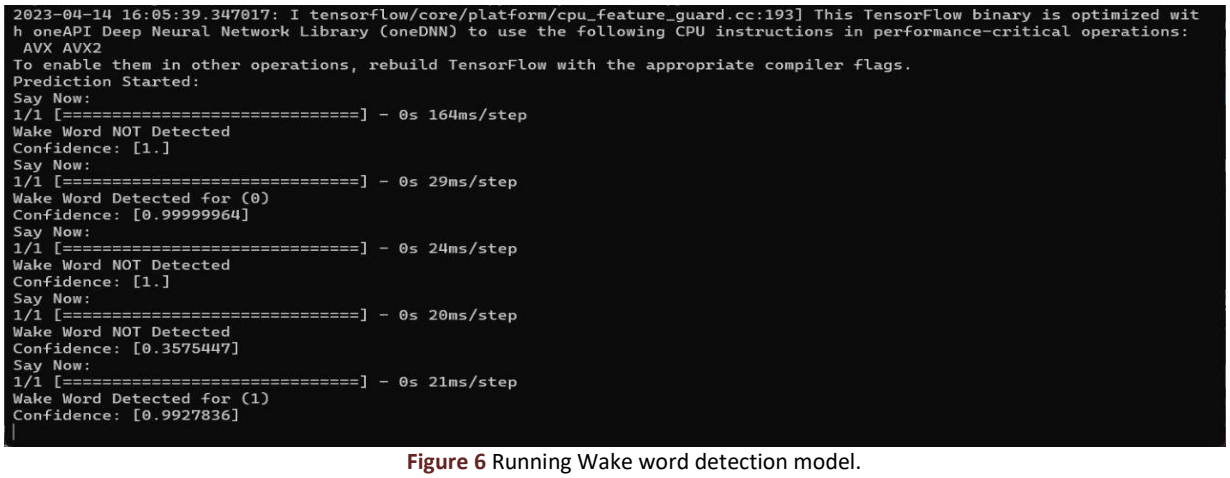

Figure 6 Running Wake word detection model.

In this plot shown in Figure 7, the horizontal axis signifies the count of wake word predictions generated by a machine learning model, while the vertical axis represents the certainty rating of each prediction. The certainty rating is assessed on a scale, where a rating of indicates high confidence in the accuracy of the predicted word. A graph of the certainty ratings would exhibit a horizontal line at y=1.0.

All predictions made by the model would possess a certainty rating of 1.0, indicating complete assurance in the accuracy of the predictions. There would be no variability in the certainty ratings as every prediction would be 100% accurate.

In this situation, there would be no need to create a histogram of the confidence ratings as every prediction would possess an indistinguishable value of 1.0. However, if a histogram were to be generated, it would exhibit a solitary peak at the highest value of 1.0, devoid of any other data points at distinct values, as shown in Figure 8.

An ideal confidence rating graph with flawless precision would be preferable, but it's worth noting that attaining 100% precision in real-world scenarios is frequently difficult or unattainable. It is also crucial to assess the model's performance on diverse datasets and in various contexts to ensure its capacity to generalize and not only excel on a specific dataset.

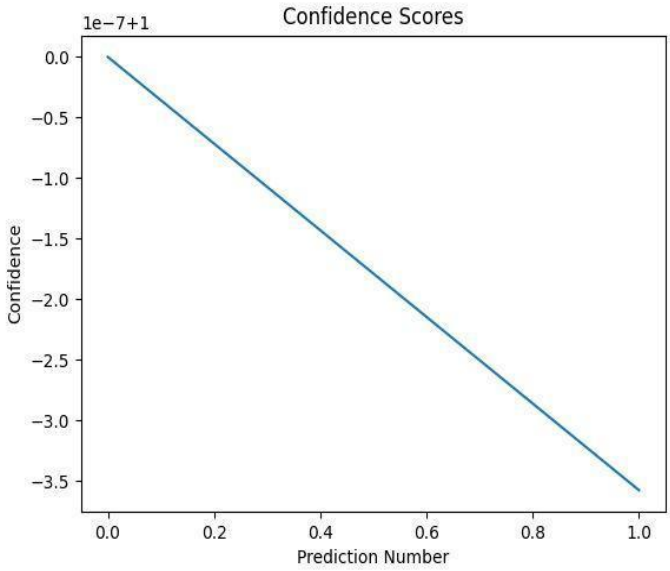

Figure 7 Count of Wake word predictions generated by machine learning model.

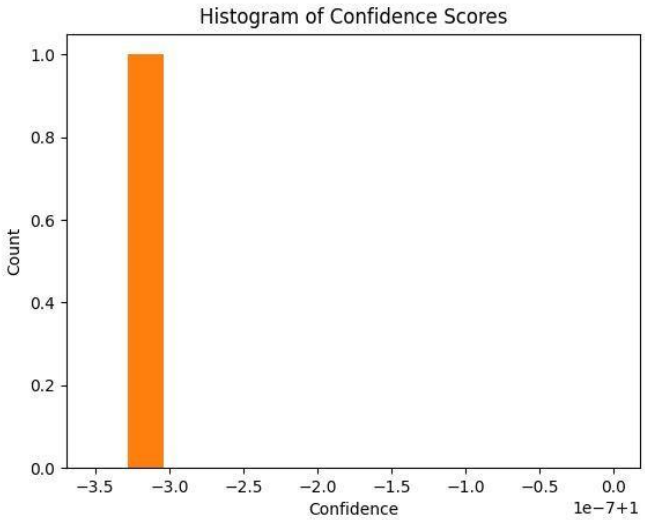

Figure 8 Certainty scores produced by artificial intelligence network.

The distribution of the assurance ratings is illustrated by the bar chart of certainty scores produced by the artificial intelligence network during its forecasts. The certainty score indicates the likelihood of detecting the specified keyword in the captured audio excerpt.

The bar chart portrays the count of forecasts (vertical axis) versus the certainty scores (horizontal axis). The histogram_bins parameter in the code defines the quantity of divisions in the bar chart, thereby influencing the level of detail in the distribution.

If the neural network were 100% accurate in detecting the wake word, then the histogram would show a peak at a confidence score of 1.0, indicating that the network was highly confident in detecting the wake word for every prediction. However, in practice, no neural network can achieve perfect accuracy, so the histogram would likely show some spread in the confidence scores, as shown in Figure 9.

The bar chart is beneficial for examining the effectiveness of the neural network. If the bar chart indicates a slender pinnacle at elevated certainty ratings, it implies that the network is functioning proficiently and possesses a substantial level of assurance in its forecasts. Conversely, if the bar chart reveals a wide array of certainty ratings, it suggests that the network is encountering difficulties in distinguishing between wake word and non-wake word audio samples.

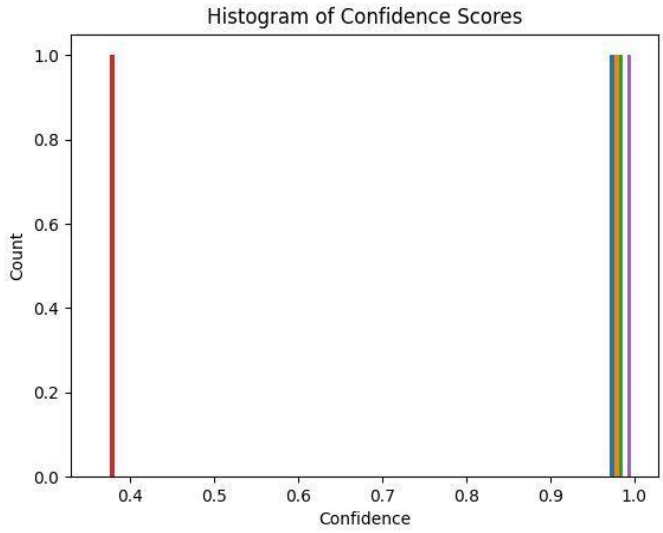

Figure 9 Frequency graph indicating frequent uncertainty in network's forecast.

The frequency graph would contain a greater number of data points at lower certainty scores, indicating frequent uncertainty in the network's forecasts. The frequency graph would also exhibit some data points at higher certainty scores, revealing occasional confidence in the network's predictions.

The frequency graph would lack a distinct peak or display a singular mode, as the neural network's accuracy in predictions is inconsistent. Instead, it would exhibit a relatively even distribution with multiple peaks or bumps, depending on the arrangement of certainty scores.

A histogram of this nature would imply that the neural network is encountering difficulties in distinguishing between wake word and non-wake word audio samples, and further efforts are required to enhance its performance. The network could undergo additional training with expanded data, fine-tuning with alternative hyperparameters, or modification of its architecture to enhance accuracy.

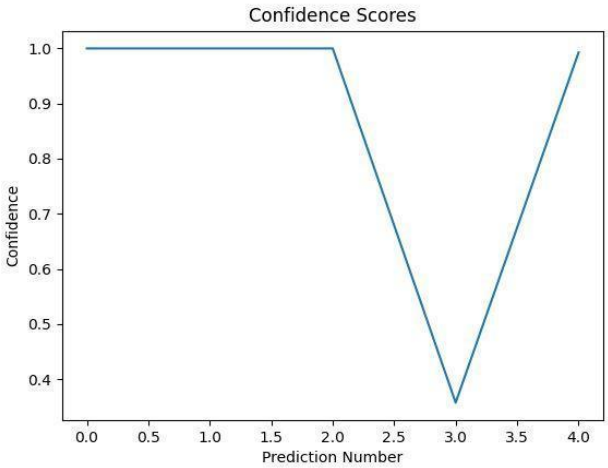

Figure 10 Wake word forecasts generated by machine learning algorithm.

In this graph shown in Figure 10, the horizontal axis symbolizes the quantity of wake word forecasts generated by a machine learning algorithm, while the vertical axis represents the certainty rating for every forecast. The confident score is measured on a scale, where a score indicates the model is very confident that the predicted word is the actual wake word. This

above score is whenever user speak different word. See it is not waked and confident score also low. If the accuracy of the neural network is 33% when detecting negative words (i.e., 67% of the time it fails to correctly identify the wake word), then the confidence score graph would likely show a lot of variation, with some predictions having high confidence scores and others having low confidence scores.

When the network detects a negative word (i.e., fails to detect the wake word), The certainty rating will be nearer to 0.0. On the contrary, when the system accurately detects the activation phrase, the certainty rating will be nearer to 1.0.

Hence, if the neural network has a precision of 33%, then the certainty rating graph would display numerous points below 0.5 (the midpoint between 0.0 and 1.0), suggesting that the network frequently doubts its predictions. Nevertheless, there would also be some points above 0.5, indicating occasional confidence in the network's predictions.

It is crucial to highlight that a low precision rate such as 33% is suboptimal and signifies that the model requires further enhancement. The model should be trained again with more data and/or adjusted with different hyperparameters to enhance its performance.

5. Conclusions

In this study, we have explored the crucial aspects of wake word activation and its implications for voice-activated systems. To ensure the effectiveness of our audio-related undertaking, it is paramount to have high-quality documented information with minimal interference from disturbances. To achieve this, we recommend introducing ambient noise such as discussions and other sounds that may be present in the surroundings during documentation to ensure that the system can effectively differentiate the desired sound from other sounds. Additionally, it is proposed to incorporate unfavourable words or phrases in the backdrop as they might impact the system's performance and precision. By integrating these elements into the documentation, we can ensure that the system is resilient and capable of handling real-life scenarios.

Moreover, we propose the integration of unfavourable words or phrases in the background data during training. This can simulate challenging scenarios and enable the system to handle real-life situations with resilience and robustness. By training the system on a substantial quantity of background data (evidenced by the 100:160 proportion), we create a more realistic environment for the system to operate. This elevated ratio of background data to documented data significantly contributes to the system's performance, resulting in a more reliable wake word activation system.

Our findings emphasize the importance of considering real-world conditions during wake word activation development. By incorporating ambient noise and background data into the documentation and training process, we can design a system capable of meeting the demands of diverse and dynamic environments. These improvements in wake word detection hold practical implications for the broader adoption of voice-activated systems, enhancing user experience and system reliability.

Ethical considerations

Not applicable.

Conflict of Interest

The authors declare no conflicts of interest.

Funding

This research did not receive any financial support.

References

Hoy MB (2018) Alexa, Siri, Cortana, and more: An introduction to voice assistants. Medical Reference Services Quarterly 37:81–88. DOI:10.1080/02763869.2018.1404391

Kadam SU, Shinde SB, Gurav YB, Dambhare SB, Shewale CR (2022) A novel prediction model for compiler optimization with hybrid meta-heuristic optimization algorithm. Int J Adv Comput Sci Appl 13. DOI:10.14569/ijacsa.2022.0131068

Kathole A, Shinde S, Wadhwa L (2023) Integrating MLOps and EEG Techniques for Enhanced Crime Detection and Prevention. Multidisciplinary Science Journal 6:2024009. DOI:10.31893/multiscience.2024009

Khoje S, Shinde S (2023) Evaluation of ripplet transform as a texture characterization for Iris recognition. J Inst Eng (India) Ser B. 104:369–380. DOI:10.1007/s40031-023-00863-6.

Mozilla common voice. Mozilla. Available in: https://commonvoice.mozilla.org/en/datasets Accessed on: April 28, 2023.

Mu W, Yin B, Huang X, Xu J, Du Z (2021) Environmental sound classification using temporal-frequency attention based convolutional neural network. Scientific reports. DOI: 10.1038/s41598-021-01045-4

Quantum (2020) DIY audio data processing: Wake word detection and sound classification. Medium. Available in: https://quantum-inc.medium.com/diy-audio-data-processing-wake-word-detection-and-sound-classification-96968fe40a93 Accessed on: May 12, 2023.

Sardeshmukh M, Chakkaravarthy M, Shinde S, Chakkaravarthy D (2023) Crop image classification using convolutional neural network. Multidisciplinary Science Journal 5:2023039. DOI:10.31893/multiscience.2023039

Shewale C, Shinde SB, Gurav YB, Partil RJ, Kadam SU (2023) Compiler optimization prediction with new self-improved optimization model. Int J Adv Comput Sci Appl. 14(2). DOI:10.14569/ijacsa.2023.0140267.

Shinde S, Khoje S, Raj A, Wadhwa L, Shaikha AS (2023) Artificial intelligence approach for terror attacks prediction through machine learning. Multidisciplinary Science Journal 6:2024011. DOI:10.31893/multiscience.2024011

Shinde S, Wadhwa L, Bhalke D (2021) Feedforward back propagation neural network (FFBPNN) based approach for the identification of handwritten math equations. In: Advances in Intelligent Systems and Computing. Cham: Springer International Publishing: 757–775. DOI: 10.1007/978-3-030-51859-2_69.

Shinde SB, Alagirisamy M, Wagh K, Dhore P (2022) Math accessibility for blind people in society using machine learning. ECS Trans. 107:18071–18090. DOI: 10.1149/10701.18071ecst

Speech_commands. TensorFlow. Available in: https://www.tensorflow.org/datasets/catalog/speech_commands Accessed on: May 2, 2023.

Terzopoulos G, Satratzemi M (2020) Voice assistants and smart speakers in everyday life and in education. Informatics in Education 473–490. DOI:10.15388/infedu.2020.21

VoxCeleb. Robots. Available in: https://www.robots.ox.ac.uk/~vgg/data/voxceleb/ Accessed on: May 15, 2023.