

Evaluation of Vector Font Rendering and Voice Recognition in Integrated Hearing Support Systems

K. Chun

Department of Semiconductor Electronic Engineering, Daegu Catholic University, Gyeongsan, Gyeongbuk, Korea

Article Info

Article history:

Received 25 Nov, 2025

Revised Dec 26, 2025

Accepted Jan 15, 2026

Keyword:

HSS(Hearing Support System)

Hangul Display

CNN

Deep Learning

ABSTRACT

This paper focuses on the implementation of core functionalities for a Hearing Support System (HSS) and the validation of its engineering feasibility. The system is designed to address the limitations of conventional hearing aids, specifically their restricted personalized calibration and environmental adaptation. The proposed HSS is a smartphone application-based system characterized by key functions: personalized settings derived from individual audiogram profiles, environment-specific presets, and real-time speech translation with textual display. Regarding the system's auxiliary output, the implementation of a Hangul (Korean) display is presented. A comparative analysis between a low-cost ESP32-based implementation (utilizing bitmap fonts) and a Raspberry Pi-based counterpart (employing vector fonts) empirically validates the necessity of vector fonts for enabling font scaling functions, which are crucial for users with low vision. For speech recognition, the study adopts an approach that transforms one-dimensional time-series audio waveforms into two-dimensional 'sound images,' specifically spectrograms, which serve as input for a Convolutional Neural Network (CNN). Conclusively, this research successfully prototyped the core functionalities of the HSS at a Proof of Concept (PoC) level, utilizing tools, thereby confirming its integration feasibility. Nevertheless, several key areas are identified as future tasks for practical deployment: the refinement of preset functionalities, the elimination of dependencies on external APIs, and fundamental enhancements to speech recognition performance through the adoption of deeper CNN architectures.

Copyright © 2026 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

K. Chun,

Department of Semiconductor Electronic Engineering,

Daegu Catholic University, Gyeongsan,

Gyeongbuk, Korea

Email: kchun@cu.ac.kr

1. INTRODUCTION

A substantial portion of the global population, estimated at roughly 20% by the World Health Organization's 2021 Global Hearing Report, experiences hearing impairment to varying extents [1]. While hearing aids are the standard intervention, their efficacy is often contingent upon an expert fitting process. This calibration procedure can be time-consuming and may fail to produce optimal outcomes, a challenge noted particularly in elderly populations. The landscape of hearing care was further altered by the 2017 OTC Hearing Aid Act in the United States. While this legislation increased accessibility for individuals with mild to moderate hearing loss, it concurrently introduced an unresolved research gap: how to reliably evaluate the speech signal processing quality of these direct-to-consumer devices without professional oversight.

The technical performance of a hearing aid is often characterized by its processed speech quality, a critical factor that directly influences user satisfaction and adoption [2]. Superior processing quality results in an auditory experience that users perceive as more natural [3]. Evaluating this quality is complex, but one

established method involves subjective assessment by the hearing-impaired users themselves, a technique that mirrors evaluation protocols used in the broader field of speech enhancement [4]–[7].

In parallel, the proliferation of data-driven methodologies, especially those leveraging deep learning (DL) [8]–[10] and reinforcement learning (RL), has significantly advanced related domains such as speech recognition, synthesis, NLP, and information retrieval (IR). This progress has enabled the development of sophisticated new architectures, including generation models, IR models, RL models, and various hybrid systems [11].

In this paper, we develop a hearing aid application to improve the listening ability of individuals with hearing impairments and additionally propose an auxiliary communication method via text output derived from speech recognition. Fundamentally, the application enhances auditory function by amplifying sound. Notably, it provides a location-based preset function to ensure optimal sound reception in various environments. Furthermore, for visual communication support, a text output feature is implemented; specifically, Hangul (Korean) output is implemented considering Korean users with hearing impairments. Regarding the speech recognition function for text output, we investigated how the dataset influences the CNN's results and, consequently, the overall speech recognition performance.

2. HEARING SUPPORT SYSTEM AND HANGUL DISPLAY

2.1. Hearing Support System (HSS)

Hearing loss constitutes a significant global health challenge affecting a substantial number of people worldwide. According to the World Health Organization (WHO), approximately 460 million individuals experience moderate or greater hearing loss, and this figure is steadily increasing. Hearing impairment can severely impact an individual's daily life, communication abilities, social interactions, and overall quality of life. Although various hearing aid devices and technologies have been developed to address this problem, many users still report dissatisfaction with their listening environments.

Conventional hearing aids often have limitations regarding user-specific customization and responsiveness to diverse listening environments. Consequently, many users experience suboptimal auditory outcomes in their daily lives. To mitigate these deficiencies, we have developed a hearing support application.

This paper introduces the functionalities of our hearing support application, developed for individuals with hearing impairments. We demonstrate how its primary features—personalized listening settings and environment-specific presets—enhance the user's auditory experience. The system provides optimal listening configurations based on the user's individual hearing profile and delivers a more convenient and effective listening environment through presets tailored to the user's surroundings. Furthermore, the application automatically translates audible sound in real-time and displays it as text, thereby helping users with hearing loss to overcome communication difficulties. These features are designed to empower individuals with hearing loss to achieve improved communication and a better quality of life.

Translation

A principal feature of the system presented in this paper is its automatic translation capability. This function captures audio from a Bluetooth-connected hearing aid, translates it in real-time into the user's desired language, and displays the resulting text on the application screen. This enables individuals with hearing impairments to understand speech in various languages and overcome communication barriers. The automatic translation function is particularly useful in multicultural environments or in situations conducted in a foreign language, supporting users in accessing a wider range of information. Through this, users with hearing impairments can transcend linguistic barriers, achieving a superior communication environment and an enhanced quality of life.

The implementation of the automatic translation function involves a process that captures and translates audio from the hearing aid in real-time. First, audio data is received from the Bluetooth-connected device. This data is then relayed to a speech recognition module and converted into text. The user can select a target language within the application, and the transcribed text is processed via a translation API to be translated into the selected language. The translated text is subsequently displayed on the app screen in real-time for the user to read. This feature dismantles linguistic barriers in diverse situations—such as daily conversations, meetings, and lectures—and provides an improved communication experience. Furthermore, continuous algorithmic improvements and user feedback are being incorporated to enhance accuracy and speed.

Figure 1 illustrates the core process of this function.

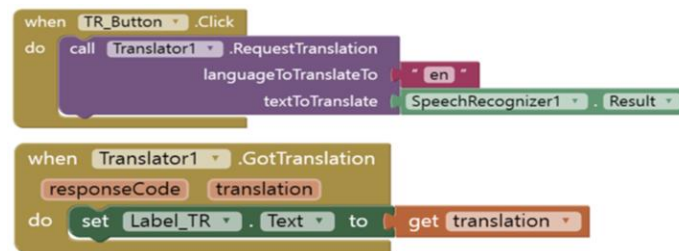


Figure 1. Translation

Preset

Another key feature of this paper is the environment-specific preset function, which adjusts sound volume and settings to match the user's location or environment. This feature is designed to ensure users experience optimal listening in various settings. The environment-specific preset function detects the user's location and ambient noise levels to apply appropriate sound settings. For example, a user's auditory requirements in a quiet indoor space differ from those while walking on a noisy street. The system utilizes the smartphone's Global Positioning System (GPS) to analyze the current environment and adjust the volume and frequency equalization based on pre-configured presets.

The environment-specific preset function identifies the user's surroundings and automatically applies the optimal sound configuration. By leveraging the smartphone's GPS, it monitors location in real-time. The collected data is then analyzed to select the appropriate preset. Users can also adjust these presets to fit their specific needs, and these customized presets are subsequently reapplied when the same environment is detected. This functionality obviates the need for users to manually adjust settings in different environments, ensuring they consistently receive an optimized listening experience. The primary advantages of the environment-preset feature are convenience and a tailored auditory experience. Consequently, users can listen more comfortably and effectively in diverse settings, leading to an improved quality of life. This facilitates better communication and interaction for users with hearing loss in their daily lives.

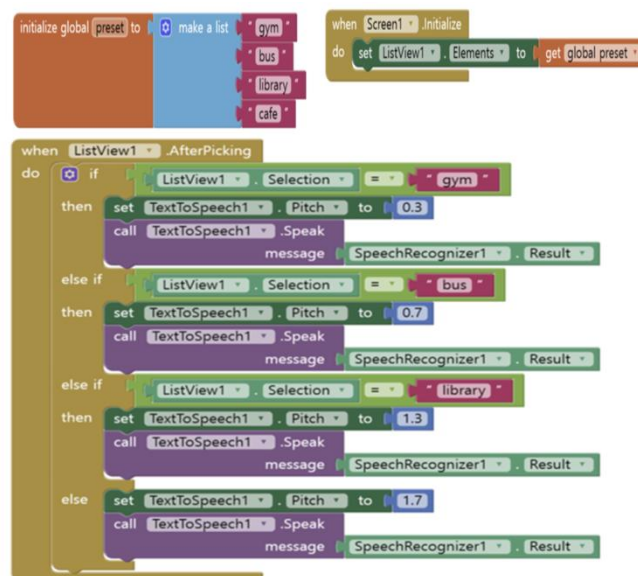


Figure 2. Preset

2.2. Hangul Display

Languages such as English or those in Europe are alphabet-based, making it possible to represent their character sets using 256 codes (e.g., Extended ASCII). However, the character sets used in China, Japan, and Korea (CJK)—which include Hangul, Hanja, Kana, Pinyin, and Bopomofo—are far too numerous to be processed by this standard. Consequently, a distinct methodology is required to handle CJK characters.

Hangul representation methods can be broadly classified into combination-based (Johab-hyung) and completion-based (Wanseong-hyung) systems, which can be further subdivided (e.g., N-byte combination, 3-

byte combination, 7-bit completion, 2-byte combination, 2-byte completion). The combination-based approach assigns individual codes to the initial (Choseong), medial (Jungseong), and final (Jongseong) consonants/vowels (collectively, Jamo), based on the generative principle of Hangul. Conversely, the completion-based method assigns a unique code to each pre-composed, complete syllable, such as '가' (ga), '갸' (gak), or '간' (gan). The completion-based method was adopted as the Korean standard, which subsequently influenced its initial adoption in the Unicode standard for Hangul.

The composition and respective advantages/disadvantages of each method are as follows:

Combination-based (Johab-hyung)

This model computerizes the foundational principle of Hangul, where a syllable is expressed by combining the codes for its constituent Jamo. The Jamo is the minimal unit; bits are allocated for the initial (e.g., 5-bit), medial (e.g., 5-bit), and final (e.g., 5-bit) components, which are logically combined to generate a character code. This results in a predictable and regular structure, allowing the Jamo to be programmatically deconstructed from the code value. Its theoretical range can represent all 11,172 possible syllables in modern Hangul, as well as archaic Hangul forms. It offers superior extensibility, as new syllables can be generated algorithmically without requiring new code points. But it necessitates a glyph composition process to render the final character shape, which can be computationally slower. Implementation is more complex than completion-based methods, as it requires a dedicated composition engine. This can also lead to inefficiencies; for example, lexicographical sorting is complicated, as it requires comparing the discrete initial, medial, and final components rather than performing a simple code-point comparison.

Completion-based (Wanseong-hyung)

This method involves pre-defining a set of frequently used Hangul syllables as complete characters and assigning a unique code point to each. It treats each syllable as an indivisible unit, similar to an English alphabet character. The syllable is the minimal unit. Since it maps a code directly to a pre-rendered glyph, it is exceptionally fast, requiring no on-the-fly composition. It ensures high backward compatibility, as it was the standard for early PC environments (MS-DOS, Windows) and web encodings (EUC-KR). Its most critical drawback is its limited character representation. Syllables not included in the pre-defined character set (e.g., the KSC 5601 standard defines only 2,350 syllables) cannot be displayed or input. This limitation is the root cause of mojibake (character corruption) and failures to represent specific proper nouns, geographical names, or neologisms.

Furthermore, text output is contingent on font technology. In digital environments, fonts are broadly categorized by their data storage and rendering methods: bitmap fonts and vector fonts.

Bitmap (Raster) Fonts

These fonts store each glyph as a grid array of pixels (a pixel map). The character's shape is defined by the specific ON/OFF state or color information of these pixels. This method presupposes a pre-generated map optimized for a specific size (e.g., 12pt). Rendering is extremely fast, as the rasterized data is copied directly from memory to the screen without complex computation. This is highly advantageous for low-specification or embedded systems with limited computing resources. But scalability is the primary limitation. When magnified, the pixel grid is enlarged, resulting in aliasing stair-stepping artifacts as shown in Figure 3. Conversely, minification causes pixels to merge, severely degrading legibility.

Vector (Outline) Fonts

These fonts define glyphs by storing their outlines as a set of mathematical equations (i.e., coordinates for points, connected by lines and curves). Because the outline is recalculated based on these mathematical definitions during scaling, the font can be enlarged or reduced to any size without quality degradation, always producing a smooth and sharp result as shown in Figure 3. To be displayed on a screen, the mathematical data must undergo a rasterization process (conversion to pixels). This requires comparatively more computational resources, although this disadvantage has been largely obviated by modern CPU/GPU performance.

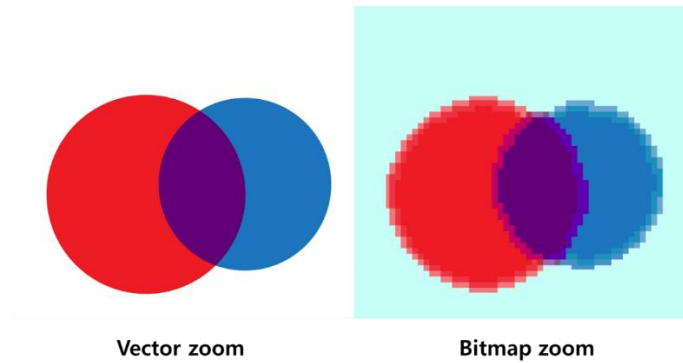


Figure 3. Font zoom

In summary, bitmap fonts were historically preferred in environments where rendering speed was critical and display resolutions were low. However, in the modern era of high-resolution displays and significant hardware advancement, the superior scalability of vector fonts has made them the standard.

We first implemented a simple Hangul display based on the ESP32 to communicate with the HSS in Figure 4. This implementation, utilizing the completion-based (Wanseong-hyung) encoding and bitmap fonts, successfully displays Hangul but suffers from limitations in character count and a three-line output restriction. Furthermore, its reliance on bitmap fonts precludes the ability to scale text, creating a significant constraint for users with low vision.



Figure 4. Hangul display with ESP32

Subsequently, we implemented a Hangul display using a Raspberry Pi, which communicates with the HSS via Bluetooth and is implemented in Python. Critically, this implementation utilizes vector fonts, specifically to accommodate text scaling. This approach allows for text to be rendered at arbitrary positions and sizes as shown in Figure 5, functionality that is essential to provide support for users with low vision.

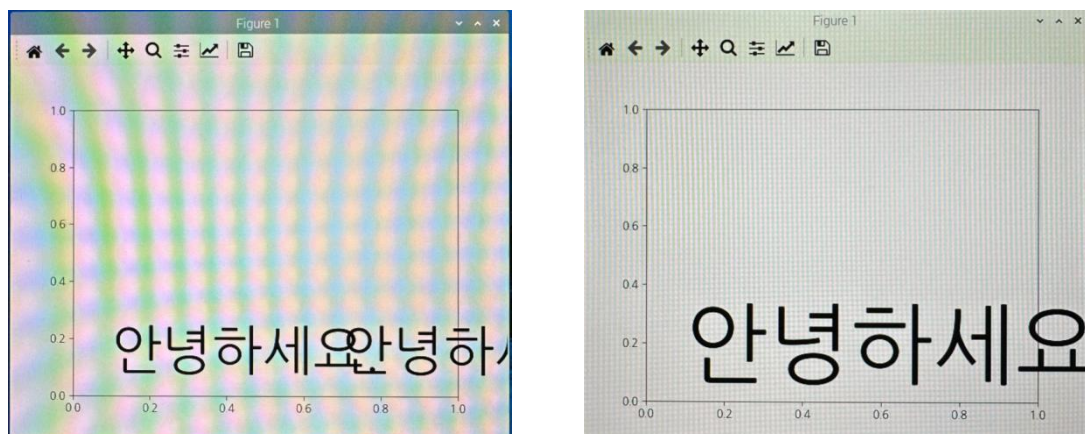


Figure 5. Hangul zoom with Raspberry pi

To compare the two systems, the latency was measured by transmitting identical Korean text data from a smartphone to each device. The duration was recorded from the initiation of data reception to the completion of the display output.

Table 1. Communication and display output latency (Unit: s)

Trial	ESP32 Latency	Raspberry Pi 4 Latency	Difference
1	0.42	0.65	0.23
2	0.45	0.62	0.17
3	0.41	0.68	0.27
4	0.39	0.61	0.22
5	0.48	0.70	0.22
6	0.44	0.64	0.20
7	0.40	0.66	0.26
8	0.43	0.63	0.20
9	0.46	0.69	0.23
10	0.42	0.62	0.20
Mean	0.430	0.650	0.220

Experimental results indicate that the ESP32 processed and displayed Korean text approximately 0.22 seconds faster than the Raspberry Pi 4 on average. This performance gap is attributed to differences in system architecture and software stack processing rather than raw hardware specifications.

The ESP32 and Raspberry Pi represent distinct categories: the Microcontroller Unit (MCU) and the Single Board Computer (SBC), respectively. They exhibit significant disparities in power efficiency and initialization speed. To evaluate these characteristics, power consumption and boot time were analyzed.

Table 2. Power consumption and boot time

Trial	ESP32 Power(mW)	RPi 4 Power(MW)	ESP32 Boot(s)	RPi 4 Boot(s)
1	245	3,150	0.82	28.5
2	238	3,210	0.79	30.2
3	251	3,080	0.85	27.8
4	242	3,300	0.81	29.1
5	247	3,120	0.83	31.4
6	235	3,250	0.80	28.9
7	255	3,180	0.84	29.5
8	240	3,220	0.82	30.7
9	249	3,100	0.86	28.2
10	243	3,240	0.81	29.9
Mean	244.5	3,185.0	0.823	29.42

The empirical data highlights the fundamental hardware distinctions between a low-power embedded system (ESP32) and a general-purpose OS-based system (Raspberry Pi).

In terms of power consumption, the ESP32 consumed over 13 times less energy than the Raspberry Pi 4. This efficiency stems from the ESP32's specialized hardware accelerators and low-power modes optimized for specific tasks like Bluetooth communication. Conversely, the Raspberry Pi requires higher baseline power to maintain its multi-core CPU, GPU, and extensive memory subsystems. This suggests that the ESP32 holds a decisive advantage for battery-powered mobile display applications.

Regarding boot time, the ESP32 achieved near-instantaneous operation with a mean of 0.82 seconds, whereas the Raspberry Pi required 29.42 seconds. This delay in the latter is due to essential OS-level procedures, including Linux kernel loading, file system mounting, and Bluetooth service initialization. Consequently, for Korean text terminals where constant power is unavailable or frequent power cycling is required, the ESP32 demonstrates superior logical suitability due to its real-time responsiveness and energy efficiency, although there is zoom difficulty for users with low vision.

Remark) Reading captions while simultaneously attempting to decode speech—a process known as audiovisual integration—imposes a significant "cognitive load" on the listener. According to the Ease of

Language Understanding (ELU) model, when the auditory signal is degraded, the brain must compensate by allocating more working memory resources to phonological matching, which often leads to mental fatigue [12](Rönnberg et al., 2013). Clinical studies, such as those by Hick and Tharpe (2002)[13] and Pichora-Fuller (2006)[14], demonstrate that individuals with hearing loss experience higher subjective and objective cognitive effort compared to normal-hearing peers during speech perception tasks.

In this context, our comparative analysis of system latency and font rendering (ESP32 vs. Raspberry Pi) becomes even more significant. High latency in speech-to-text translation can disrupt the temporal synchronization between the speaker's lip movements and the displayed text, thereby increasing the "split-attention effect" and further exacerbating cognitive load (Mayer & Moreno, 1998)[15]. Our findings show that the ESP32-based system minimizes processing delay, while the Raspberry Pi's vector font rendering provides superior legibility (scaling). By optimizing these engineering parameters, our system aims to reduce the extraneous cognitive load, allowing users to allocate more cognitive capacity to the actual comprehension of the message.

2.3. Voice Recognition Method

Convolutional Neural Networks (CNNs) are models that inherently excel at learning the spatial hierarchies present in image data. However, the potent feature extraction capabilities of CNNs can be effectively applied to the domain of speech recognition by transforming the one-dimensional (1D) time-series data of an audio signal into a two-dimensional (2D) representation, typically in the time-frequency domain. This approach effectively treats the audio signal as a sound image, allowing the network to detect salient acoustic features using the same principles applied to the recognition of local patterns in visual data.

In this paper, to demonstrate the speech recognition process employed, a Convolutional Neural Network (CNN) model is constructed using the TensorFlow deep learning library. The model is specifically designed to recognize speech commands. This procedure encompasses the fundamental stages of AI modeling, including data preprocessing, model architecture design, training, and inference.

For training the speech recognition model, Google's mini speech commands dataset, comprising 8,000 audio files, was utilized. The dataset is structured into eight distinct command classes (e.g., yes, no, up, down), with each class containing 1,000 corresponding WAV audio files.

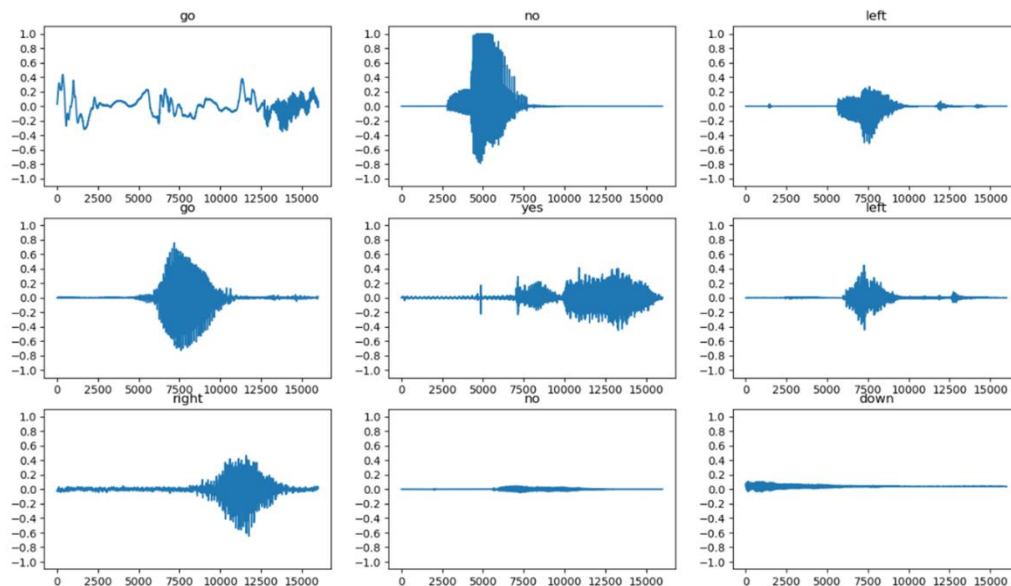


Figure 6. Waveform

The entire dataset was partitioned into a training set (80%) and a validation set (20%) for model performance verification. Furthermore, as the recognition process utilizes single-channel (mono) audio, a step was included to remove superfluous dimensions.

Direct processing of raw audio data by AI models, particularly CNNs, is computationally inefficient. Therefore, a preprocessing stage to transform the data into an image-like 2D format is essential. In this study, the 1D time-series audio waveform is converted into a spectrogram as shown in Figure 7. A spectrogram visualizes the changes in frequency components over time, enabling signal analysis within the time-frequency domain. This conversion is executed via a Fourier transform function. The magnitude of the

frequency components is extracted by taking the absolute value of the resulting complex numbers. Subsequently, a channel dimension is added to conform to the standard CNN input format (batch size, height, width, channels), completing the 2D image representation.

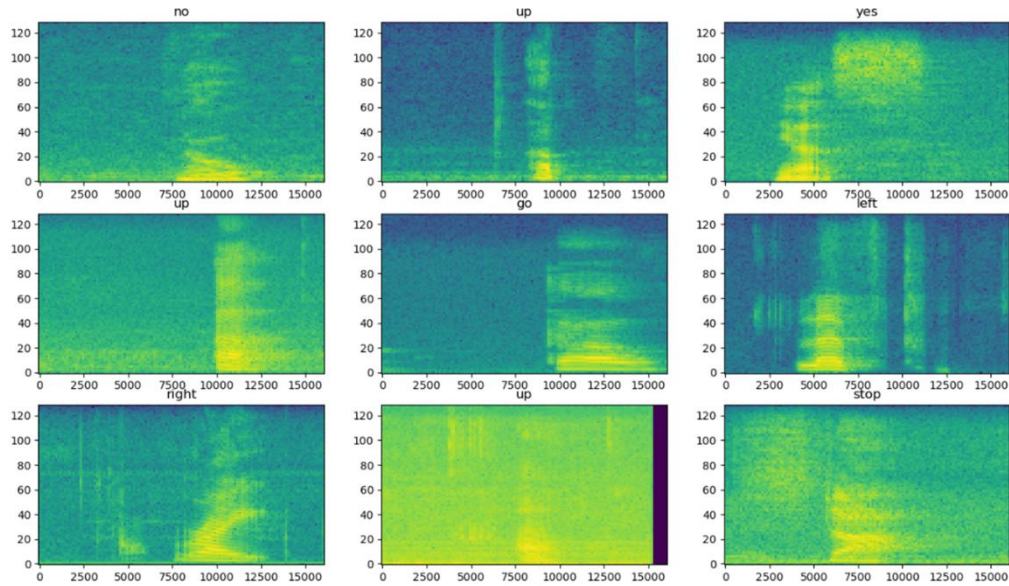


Figure 7. Spectrogram

For data processing, a CNN model was constructed based on the Keras Sequential model. CNNs are typically built using either the Sequential or the Functional API. The Sequential model is the simplest approach, involving a linear, sequential stacking of layers, analogous to stacking Lego blocks. This method is intuitive and concise, making it ideal for rapid prototyping or simple models. However, it is restricted to a single input and a single output, and its linear-only pathway is insufficient for implementing complex, non-linear architectures such as residual connections or multi-input/output branches. In contrast, the Functional API treats layers as functions that are called on tensors, allowing for direct I/O connections. This permits the design of highly flexible and complex architectures, conceptualized as Directed Acyclic Graphs (DAGs), which can accommodate multiple inputs, multiple outputs, and shared layers. This flexibility, however, increases implementation complexity, as the code is more verbose and the model's structure must be explicitly defined.

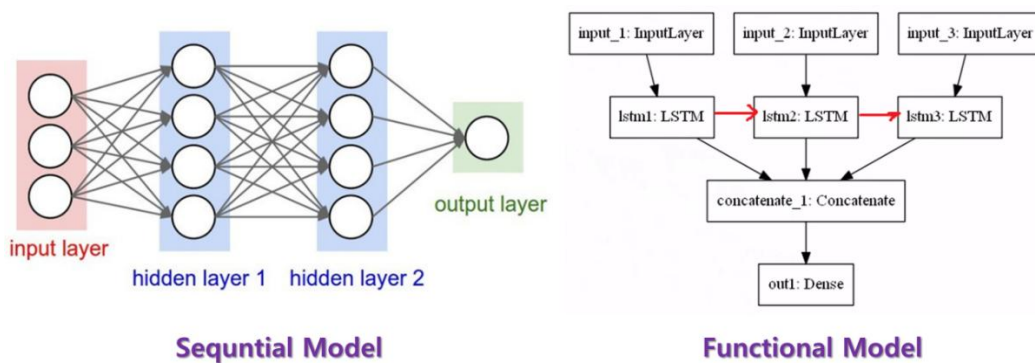


Figure 8. CNN

In the Sequential model applied in this paper, the input spectrogram is first standardized to a uniform size via a Resizing layer and its data scale is normalized via a Normalization layer. For feature extraction, two Conv2D layers are interleaved with MaxPooling2D layers. These are used to extract local features from the spectrogram and to downsample the feature maps. The extracted feature maps are then "flattened" into a 1D vector by a Flatten layer. This vector is then passed to a Dense (fully connected) layer, which performs the final classification into one of the eight command classes.

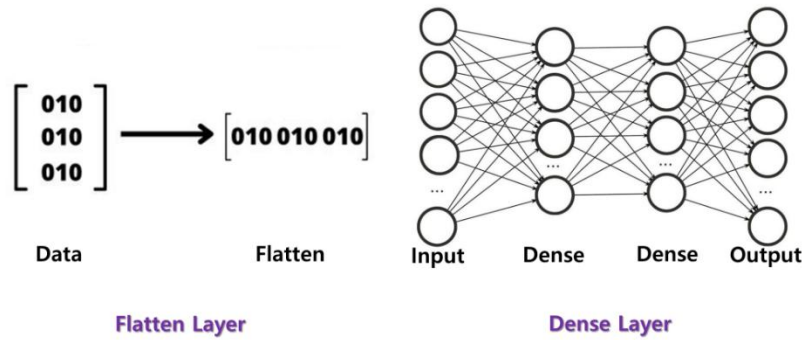


Figure 9. Flatten and dense layer

To improve the model's generalization performance and mitigate overfitting (the excessive optimization to training data), the Dropout technique is applied. During the compilation stage, the Adam optimizer, one of the most efficient and widely used gradient descent algorithms, was selected. For the loss function, SparseCategoricalCrossentropy was chosen, as it can directly process integer-form labels. This is a standard and appropriate choice for multi-class classification problems.

The CNN-based speech recognition model represents an innovative approach, applying image processing techniques to 2D spectrograms derived from 1D audio signals. This process is highly effective for automatically learning and extracting the local features and hierarchical structures inherent in the time-frequency domain of speech data. The convolutional filters learn to become optimal, data-driven acoustic feature detectors, while pooling ensures invariance for these features. This methodology effectively replaces the complex manual feature engineering required in traditional approaches and serves as a powerful paradigm for achieving high performance in various audio-related tasks, such as speech recognition, speaker identification, and emotion analysis.

3. TEST AND VALIDATION

In this study, the proposed Hearing Support System (HSS) was implemented using a rapid prototyping tool as shown in Figure 10.

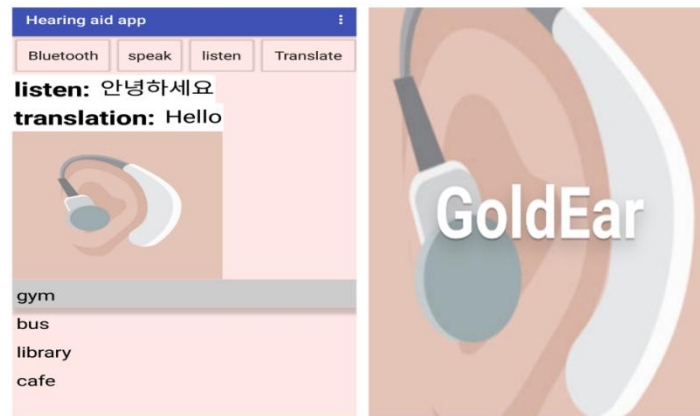


Figure 10. HSS

From an engineering standpoint, the real-time translation feature successfully validates the HSS, as it demonstrates the integration of heterogeneous systems: Bluetooth connectivity, a Speech-to-Text (STT) component, and a translation API. For practical deployment, however, the critical performance metrics will be the end-to-end pipeline latency (from audio input, through API call, to text return) and the recognition accuracy in noisy ambient environments.

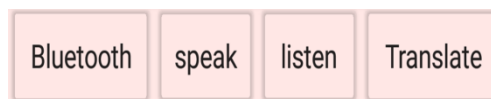


Figure 11. Functions

The environment-specific preset function was implemented using GPS-based location awareness, which constitutes a foundational level of automation via geofencing. However, dynamic acoustic environment analysis—specifically, ambient noise level detection using the microphone sensor—has not yet been implemented. This is a necessary future step, as the noise profile can differ completely even at the same location (e.g., a bus at a stop versus the same bus in motion). Therefore, this paper has implemented the HSS concept at a Proof of Concept (PoC) level. Future work will focus on advancing the system by reducing API dependency to improve responsiveness and by implementing a true environment-adaptive preset. This will be achieved by introducing a sensor fusion algorithm that integrates both GPS and microphone inputs, thereby better aligning with the system's original objective.

The CNN applied for speech recognition utilized the Keras Sequential model, with its architecture detailed in Table 3. The architecture comprises two Conv2D layers, one MaxPooling2D layer, and Dense layers. Dropout layers were included to mitigate overfitting and enhance generalization.

Table 3. CNN-Seuential API

Layer	Output Shape	Number of Parameters
Resizing	(None, 32, 32, 1)	0
Normalization	(None, 32, 32, 1)	3
Conv2d	(None, 30, 30, 32)	320
Conv2d_1	(None, 28, 28, 64)	18,496
Max_pooling2d	(None, 14, 14, 64)	0
Dropout	(None, 14, 14, 64)	0
Flatten	(None, 12544)	0
Dense	(None, 128)	1,605,760
Dropout_1	(None, 128)	0
Dense_1	(None, 8)	1,032

The model was trained for 5, 10, 15, and 20 epochs. An EarlyStopping callback was applied. As shown in Figure 12, the 20-epoch training session was halted prematurely at the 19th epoch. Consequently, the final results were compared primarily across the 5, 10, and 15-epoch models.

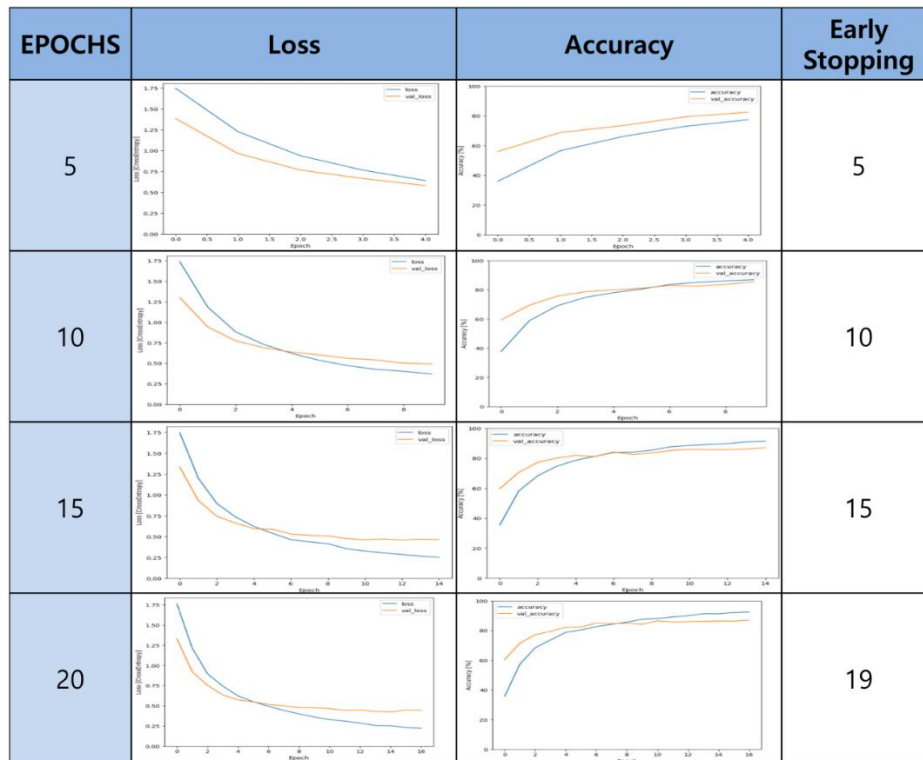


Figure 12. Early Stopping

Figure 13 presents the accuracy results corresponding to the training durations. The comparison indicates a trend of improving accuracy as the number of training epochs increases. However, a notable decline in validation accuracy is observed when training exceeds 15 epochs (e.g., the 20-epoch run), dropping from 86.3% to 84.7%. This serves as definitive evidence of overfitting, where the model has begun to learn granular noise specific to the training data, thereby degrading its generalization performance on unseen (validation) data. Therefore, the 15-epoch model is considered the most reliable.

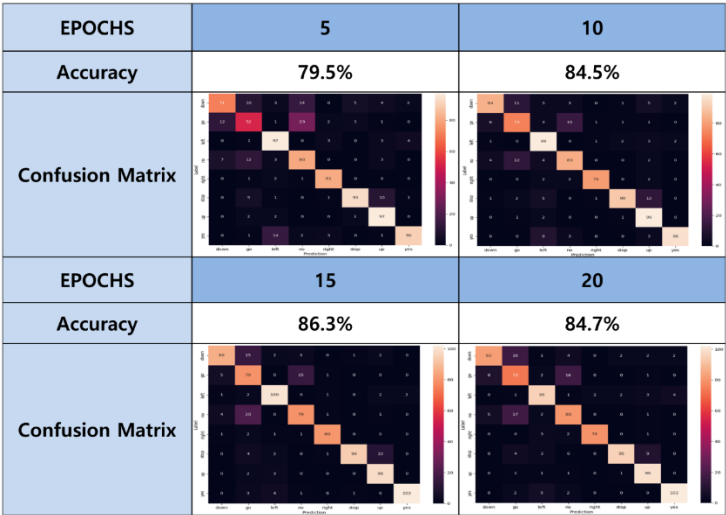


Figure 13. Confusion martix and accuracy

Recognition tests were also performed on individual data samples. The results for Test #1 in Figure 14 show that the model's confidence in the correct label generally increases with training. For the command 'No', the confidence score increased from 0.5 to 0.7, and finally to 0.8 (15-epoch model). For 'Up', the score improved from 0.8 (5 epochs) and saturated at 1.0 (10 and 15 epochs). 'Go' showed a steady increase from 0.5 to 0.9. Furthermore, the magnitude of misclassification (i.e., the confidence score assigned to incorrect labels) tends to decrease as training progresses.

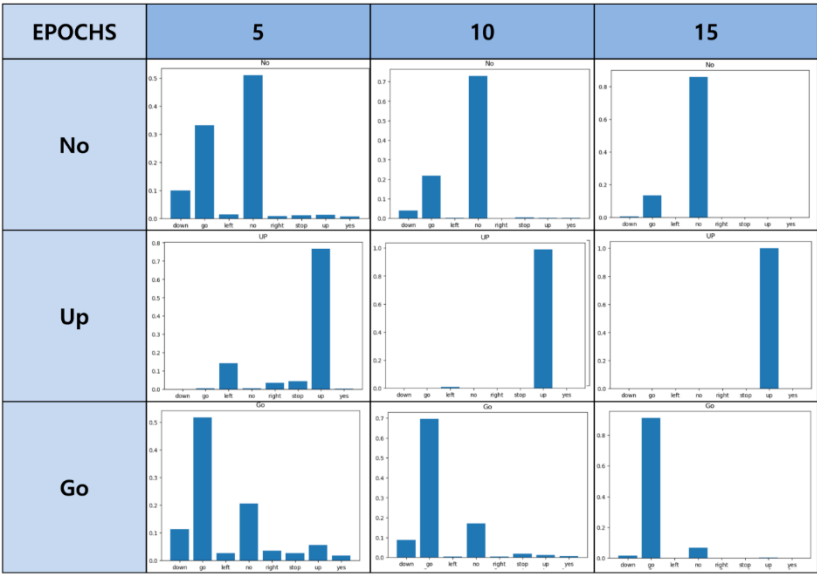


Figure 14. Test #1(No, Up, and Go)

The results for Test #2 in Figure 15 are largely consistent, showing changes in accuracy and misclassification corresponding to training duration. For 'stop', an anomaly was observed: the 5-epoch model (0.7) outperformed the 10-epoch model (0.5), which correlated with a high misclassification magnitude (0.4) for the 'up' command in the 10-epoch model. The 'Right' command was recognized with near-perfect accuracy (1.0) across all models. The 'Down' command exhibited a clear proportional increase in accuracy

with training (0.35, 0.7, 1.0). Notably, in the 5-epoch model, the misclassification score for 'go' (0.4) was higher than the correct score for 'down' (0.35), suggesting a fundamental recognition error rather than just low confidence.

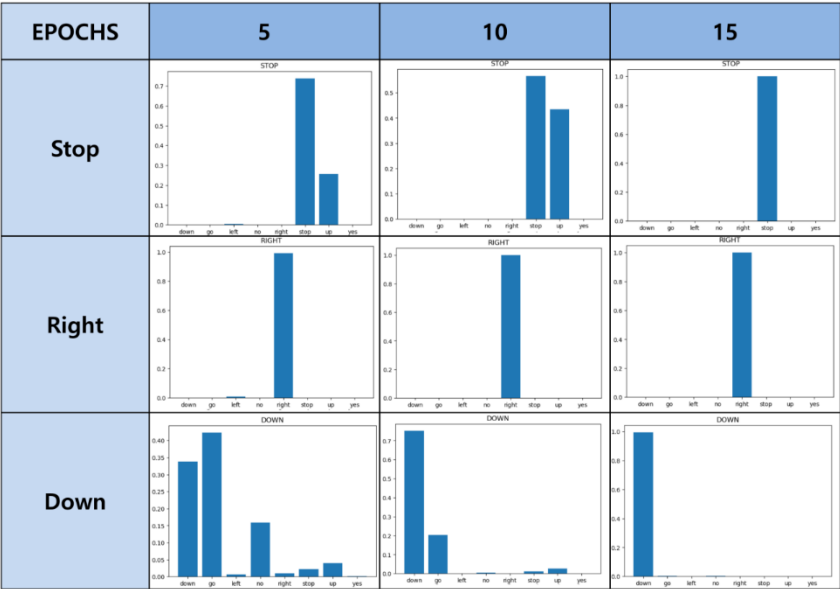


Figure 15. Test #2(Stop, Right, and Down)

Remark: The decision to halt training at 15 epochs, guided by the EarlyStopping callback, represents a sound engineering judgment that successfully prevented overfitting and secured the optimal generalization performance. To further enhance model performance, future work should consider strategies that improve both representational power and generalization ability, such as: (1) increasing the Dropout rate, (2) introducing audio augmentation, or (3) adopting deeper CNN architectures, such as VGG or ResNet.

4. CONCLUSION

In this paper, the core functionalities of a Hearing Support System (HSS) for individuals with hearing impairments were implemented, and its engineering feasibility was validated. The system comprises context-aware presets utilizing smartphone sensors, an AI-based real-time speech translation feature, and the implementation of a Hangul (Korean) output system for an auxiliary display. For speech recognition, a core component of the system, a signal processing technique was applied to convert 1D audio waveforms into 2D spectrograms. A CNN model based on two convolutional layers was constructed using these spectrograms as input. Through experimentation, it was confirmed that the model achieves optimal generalization performance at 15 epochs. This result represents a rational engineering trade-off derived to prevent overfitting. Conclusively, this research successfully prototyped the core functionalities of the HSS, thereby demonstrating its engineering viability. Future research should be directed toward enhancing the system's responsiveness, accuracy, and robustness by eliminating API dependencies, refining sensor fusion algorithms, and applying advanced overfitting control techniques.

ACKNOWLEDGEMENTS

This work was supported by research grants from the Daegu Catholic University in 2024.

REFERENCES

[1] Chadha S, Kamenov K, Cieza A. World report on hearing. World Health Organization, Geneva, Switzerland, 2021.

[2] Picou E. M. MarkeTrak 10 (MT10) survey results demonstrate high satisfaction with and benefits from hearing aids. *Seminars in Hearing*, New York, NY, USA: Thieme Med. Publishers, 2020; 41(01): 21–36.

[3] Kates J. M., Arehart K. H. The hearing-aid audio quality index (HAAQI). *IEEE/ACM Transactions on Audio Speech Language Processing*. 2016; 24(2): 354–365.

[4] Richard J, Zimpfer V., Roth S. Comparison of objective and subjective methods for evaluating speech quality and intelligibility recorded through bone conduction and in-ear microphones. *Applid Acoustics*. 2023; 211. Art. No. 109576. [Online]. Available: <https://www. sciencedirect.com/science/article/pii/S0003682X23003742>

- [5] Schröter H., Rosenkranz T., Escalante-B A. N., Maier A. Low latency speech enhancement for hearing aids using deep filtering. *IEEE/ACM Transaction on Audio Speech Language Processing*. 2022; 30: 2716–2728.
- [6] Kavalekalam M. S., Nielsen J. K., Boldt J. B., Christensen M. G. Model-based speech enhancement for intelligibility improvement in binaural hearing aids. *IEEE/ACM Transactions on Audio Speech Language Processing*. 2019; 27(1): 99–113.
- [7] Eskimez S. E., Yoshioka T., Wang H., Wang X., Chen Z., Huang X. Personalized speech enhancement: New models and comprehensive evaluation. *Proceeding of IEEE International Conference on Acoustic Speech Signal Processing*. 2022; 356–360.
- [8] Deepa P., Khilar R. A Report on voice recognition system: techniques, methodologies and challenges using deep neural network. 2021 *Innovations in Power and Advanced Computing Technologies (i-PACT)*. 2021; 1–5.
- [9] Cayir A. N., Navruz T. S. Effect of Dataset Size on Deep Learning in Voice Recognition. 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (IEEE HORA). 2021; 1–5.
- [10] Mykhailichenko I., Ivashchenko H., Barkovska O., Liashenko O. Application of Deep Neural Network for real-time voice command recognition. 2022 IEEE 3rd KhPI Week on Advanced Technology (IEEE KhPIWeek). 2022; 1–4.
- [11] Cui F., Cui Q., Song Y. A Survey on learning-based approaches for modeling classification of human-machine dialog systems. *IEEE Transactions on Neural Network and Learning Systems*. 2021; 32(4): 1418–1432.
- [12] Rönnberg J., Lunner T., Zekveld A., Sörqvist P., Danielsson H., Lyxell B., Dahlström O., Signoret C., Stenfelt S., Pichora-Fuller MK., Rudner M. The Ease of Language Understanding (ELU) model: theoretical, empirical, and clinical advances. *Frontiers in Systems Neuroscience*. 2013.
- [13] Hick C. B., Tharpe A. M. Listening effort and fatigue in school-age children with and without hearing loss. *Journal of Speech Language and Hearing Research*. 2002; 45(3): 573–84.
- [14] Pichora-Fuller M. K., Singh G. Effects of age on auditory and cognitive processing: implications for hearing aid fitting and audiologic rehabilitation. *Trends in Amplification*. 2006; 10(1): 29–59.
- [15] Mayer, R. E., Moreno, R. A split-attention effect in multimedia learning: Evidence for dual processing systems in working memory. *Journal of Educational Psychology*, 1998; 90(2), 312–320.