Integrating Shogi Game with AI Voice Assistant for Enhanced Interactivity and Gameplay Experience

Kai-Jun Zhong
Department of Electrical Engineering
National Dong Hwa University
Hualien, Taiwan
410823070@gms.ndhu.edu.tw

Chieh-Ming Yang
Department of Electrical Engineering
National Dong Hwa University
Hualien, Taiwan
j82887@gmail.com

Jen-Yeu Chen
Department of Electrical Engineering
National Dong Hwa University
Hualien, Taiwan
jenyeu@gms.ndhu.edu.tw

ABSTRACT

This study aims to combine the game of shogi with an artificial intelligence voice assistant to enhance the enjoyment and interactivity of playing shogi. We have designed a system capable of recognizing users' voices to awaken a voice assistant and perform voice-controlled actions for specific Chinese phrases. To achieve this goal, we are utilizing the technique of converting Mel-scale Frequency Cepstral Coefficients into spectrogram images. These images serve as inputs to the modified Inceptionv3 model, leading to the generation of a 128-dimensional feature vector as output. In our study, each individual's voice has a unique feature vector, and we are employing a triple loss function to differentiate between different voices. Our experimental results demonstrate that the accuracy of this research method reaches 93.00%, reliably identifying distinct users' voices. We are further applying this technology to our designed shogi game, enabling voice-controlled manipulation of pieces and moves to designated positions. The system autonomously determines whether the actions adhere to game rules, thereby ensuring fairness and legitimacy. The outcomes of this study offer a novel interactive approach to shogi, allowing players to engage in the game solely through voice commands. The application of voice-controlled actions has the potential to enhance gaming experiences and user engagement.

Keywords: Shogi, Voice Assistant, Speech Verification, Mel-scale Frequency Cepstral Coefficient, Triple Loss, Inceptionv3.

I. INTRODUCTION

Prominent virtual voice assistants like Siri and Google Assistant not only encompassed identity verification capabilities but also executed commands, responded to simple queries, and even conducted online searches, establishing themselves as essential applications within the realm of artificial intelligence. With the advent of technologies such as ChatGPT, a broader array of applications emerged, offering enhanced functionalities and superior user experiences, adaptable to users' needs and intents during conversations.

The objective of speaker verification was akin to facial recognition in images, aiming to confirm individual identities. However, this task was susceptible to environmental noise and device discrepancies, which in turn impacted recognition accuracy. Extensive research in the field of speaker verification has been conducted previously. Heigold et al. [1] proposed an end-to-end system trained on the phrase "OK Google," which learned embeddings and a similarity metric for comparing pairs of embeddings. Li Wan et al. [2] proposed the Generalized End-to-End (GE2E) loss function, designed to enhance the efficiency of training speaker verification models when contrasted with the Tuple-Based End-to-End (TE2E) loss. David Snyder et al. [3] had augmented data to improve x-vector-based deep neural network embeddings, resulting in superior speaker recognition performance. David Snyder et al. [4] proposed deep neural network embeddings with temporal pooling for speaker verification, surpassing i-vectors' performance on short segments and achieving competitive results on longer segments, demonstrating their effectiveness in establishing speaker-discriminative representations. Chao Li et al. [5] proposed the introduction of the Deep Speaker system, which involved the conversion of speech into spectrograms using Mel-scale Frequency Cepstral Coefficients (MFCCs) for tasks including speech classification, speaker identification, and verification.

We could observe unique application prospects by extending these concepts to the realm of shogi games. By integrating voice verification technology into shogi games, players could operate the game through voice commands, such as moving pieces or executing specific actions, without relying on traditional keyboard or mouse inputs. This intuitive interaction approach enhanced the smoothness and naturalness of the gaming experience, while also augmenting the entertainment value of the game. Additionally, the inclusion of an AI voice assistant enriched game interactions, enabling players to converse with the virtual assistant and seek hints or advice, thereby elevating the strategic and challenging aspects of the game.

In the second chapter of this paper, we introduced the shogi game, including fundamental rules such as the inability to stack pieces, correct movement positions, and restrictions on jumping over pieces. In the third chapter, we provided a detailed overview of the speaker verification method, capable of recognizing specific users' voices and determining whether they were the current players for the respective game round, while also ensuring compliant shogi piece movements.

II. GAME ENVIRONMENT SETUP

The game is played on a 9x9 board with a designated area for captured pieces. Shogi has fixed initial configurations and various rules including capturing pieces, promotion (upgrading pieces under specific conditions), dropping pieces onto the board, and rules related to illegal moves. In this study, we developed a

shogi board user interface using the Tkinter [6] library in Python. This interface enables players to make moves on the board while adhering to game rules. The designed game interface is visually represented in Fig. 1.

Figure 1. Game interface design.

In the process of rule implementation, we start by designing the board layout and placing the pieces in their initial positions. We establish movement vectors for each type of piece and design trigger events to enable piece movement. These triggers consist of two parts: the first involves left-clicking on a piece and then right-clicking on a target square; the second involves left-clicking on a piece to select one from the captured pieces area and then middle-clicking to place the piece onto the board. The process diagrams for left-click and right-click actions in the system are depicted in Fig. 2 and Fig. 3, respectively.

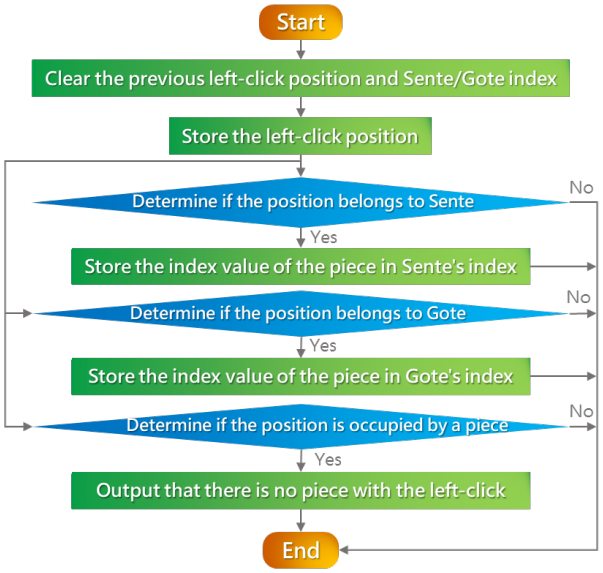


Figure 2. Left-click interaction process diagram.

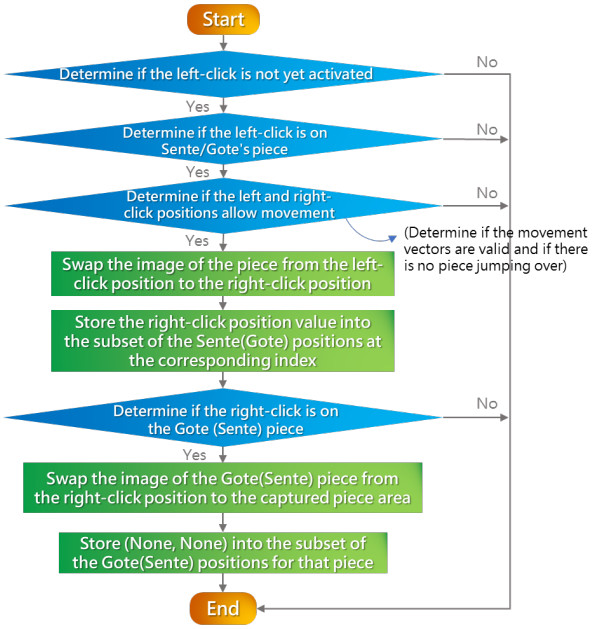


Figure 3. Right-click interaction process diagram.

The movement vectors are centered around (0,0) and their permissible ranges are stored in an array. Taking Promoted Pawns as an example, there are a total of six vectors. By utilizing left-click and right-click positions, these vectors are obtained, and it is checked whether they exist within the piece's movement vector array. Apart from this, it is also necessary to determine stacking and jumping over pieces. These methods ensure the correctness of the movement, as illustrated in Fig. 4.

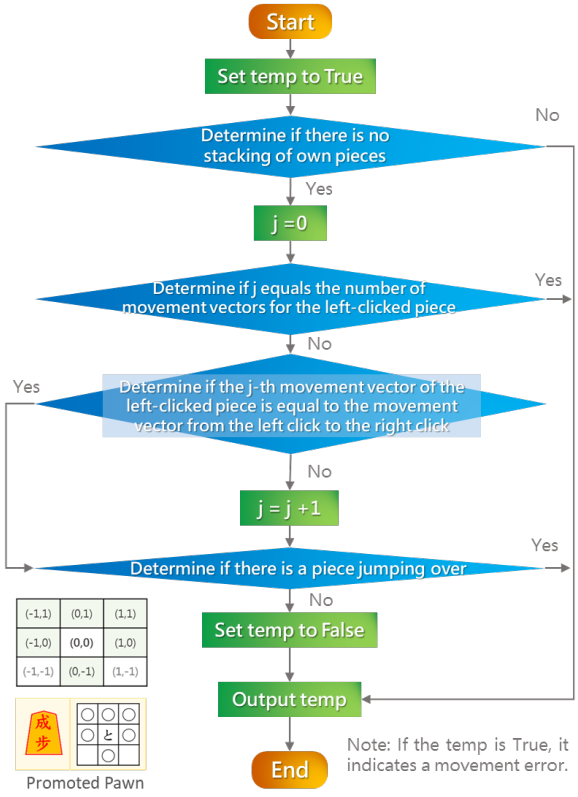


Figure 4. The process diagram for movement rules.

From the aforementioned movement rules, we have created movement vector cues. By clicking on a piece on the screen, one can ascertain the allowable movement directions, as highlighted by the red box in Fig. 5. Additionally, an information panel has been set up to provide details like the current round number and rule violations.

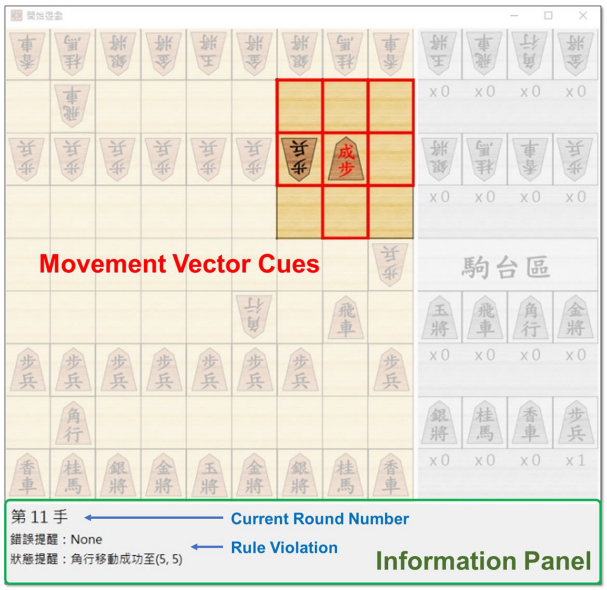


Figure 5. The illustrative diagram of movement cues for the Promoted pawn.

III. SPEAKER VERIFICATION

We designed a system capable of recognizing users' voices, responding to specific vocal commands, waking up a voice assistant, and performing voice-controlled actions. The process flow of this system is depicted in Fig. 6 and is divided into three stages: data collection, data preprocessing, and model training.

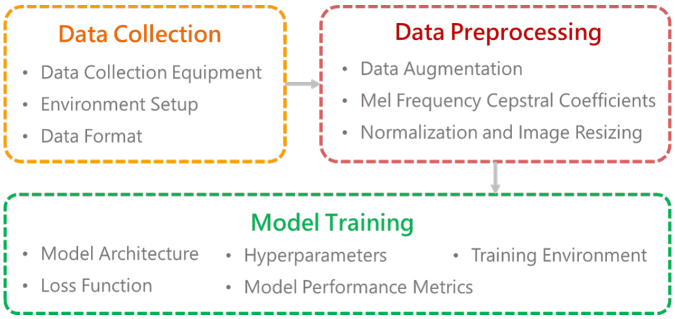


Figure 6. Method flowchart for speaker verification.

A. Data Collection

When collecting voice data for our study, we utilized the same recording device, an iPhone 10, for audio capture. All recordings were conducted within a consistent recording environment. We recorded 10 audio files for each of the 5 participants, with each file containing the designated Chinese phrase “píngguǒ.” We employed the Pydub audio processing library in Python to convert these recorded audio files into WAV format to meet specific formatting requirements.

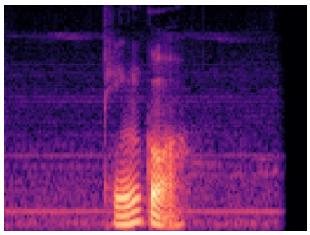
B. Data Preprocessing

1)Data Augmentation

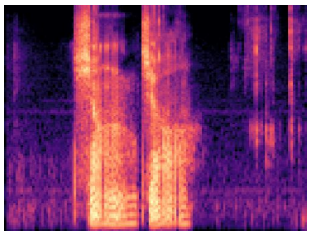
This study utilized the open-source software FFmpeg (Fast Forward MPEG) for data augmentation. FFmpeg is a versatile multimedia framework capable of handling various audio and video, including encoding conversion, recording, streaming, and more. We used FFmpeg to adjust audio speed and volume. Following data augmentation, we obtained a total of 180 audio files. We further normalized the audio durations, trimming audio files longer than 3 seconds and adding silence to those shorter than 3 seconds.

2)Mel-frequency Cepstral Coefficients

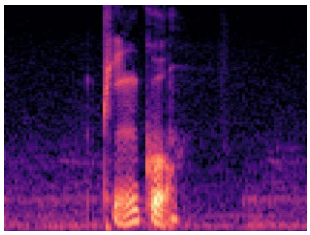
MFCC [7] designed based on human auditory sensitivity to different frequencies, is suitable for representing audio data. We employed the Librosa library to conduct this transformation, converting the original speech data into image files. Fig. 7 showcases MFCC spectrograms after conversion, representing designated and non-designated speech recordings from Participant A, as well as designated speech recordings from Participant B. The differences between these spectrograms are observable.



(A) Participant A: Designated



(B) Participant A: Non-Designated



(C) Participant B: Designated

Figure 7. Comparison of MFCC spectrograms for different speech recordings.

3)Normalization and Image Resizing

We processed the collection of MFCC spectrogram image data. Firstly, we normalized the pixel values of each image by dividing them by 255, bringing the pixel values within the range of 0 to 1. This normalization ensures consistent value ranges across all images, facilitating model training and convergence. Additionally, we uniformly resized the images to dimensions of 224x224 for further analysis.

C. Model Training

This section introduces the training of the speaker verification model, including model architecture, loss function, hyperparameters, performance metrics definition, and the training environment.

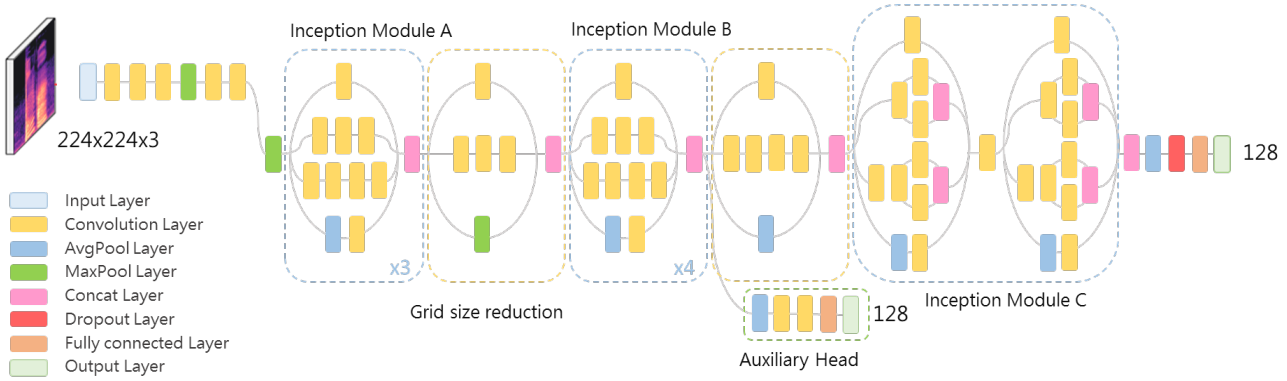


Figure 8. Neural network architecture for speaker verification in this study.

1)Model Architecture

We adopted the InceptionV3 [8] architecture for our speaker verification model, a widely used model in image classification tasks. InceptionV3 employs a multi-layer structure that focuses on extracting and transforming image features at different levels. This multi-layered design enables the model to capture features ranging from low-level edges and textures to high-level object shapes and structures. Moreover, InceptionV3 integrates three types of modules: the Inception module, which simultaneously processes input feature maps with various-sized convolution kernels and pooling operations to capture multi-scale information; the auxiliary head module, introducing an auxiliary classifier to enhance gradient propagation during training; and the grid size reduction module, which efficiently reduces feature map dimensions while maintaining the model's ability to capture features across different scales and hierarchies. In our model, the output layer was adjusted to yield a 128-dimensional feature vector. To expedite convergence, we employed a pre-trained model from ImageNet for transfer learning. The neural network architecture designed for this speaker verification is depicted in Fig. 8.

2)Loss Function

Our model training utilized the triplet loss [9] as the loss function. The triplet loss aims to minimize feature distances between samples of the same class while simultaneously maximizing the distances between samples of different classes. The loss function is defined as follows:

$$L(A,P,N) = max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0). \quad (1)$$

Here, $\|f(A) - f(P)\|^2$ represents the distance between the anchor sample (A) and the positive sample (P), while $\|f(A) - f(N)\|^2$ represents the distance between the anchor sample and the negative sample (N). α is a predefined margin value used to ensure differentiation between positive and negative samples; in this study, α is set to 1. This model with triplet loss embeds samples from the same category into close regions and separates samples from different categories into distinct regions, as illustrated in Fig. 9. Minimizing the triplet loss can enhance the neural network's performance in similarity measure or contrastive learning tasks.

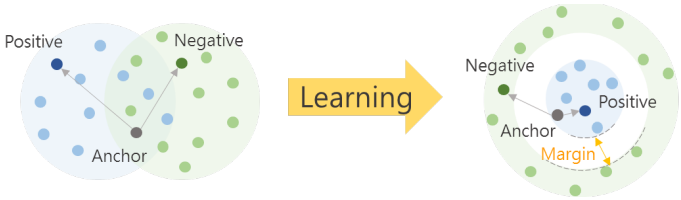


Figure 9. Illustration of triplet loss in embedding space.

3)Hyperparameters

During training, we utilized a variant of the gradient descent method called Adam [10] to minimize the Triplet Loss. In each batch, we selected multiple triplets (anchor, positive, and negative samples) from the dataset, computed their loss, and updated the model's parameters to reduce the loss. The learning rate for this study was set to 0.0001, the number of epochs was set to 50, and the batch size was configured to 30 data samples per batch.

4)Model Performance Metrics

We calculated the distances between pairs of samples using their respective 128-dimensional feature vectors and compared these distances against a threshold value (Th_d) to determine whether the samples belonged to the same person or different people. In this study, the Th_d was set within the range of 0.5 to 2. If the distance between the anchor and positive sample was less than the Th_d, it was considered a correct match; otherwise, it was incorrect. Similarly, if the distance between the anchor and negative sample was less than the Th_d, it was considered incorrect; otherwise, it was correct. D_{True} represents the count of correct matches, and D_{False} represents the count of incorrect matches. Consequently, accuracy is defined as follows:

$$Accuracy = \frac{D_{True}}{D_{True} + D_{False}}. \quad (2)$$

5)Training Environment

We utilized Python 3.9 and TensorFlow 2.12.0 for development and training in our research environment. Google Colaboratory was chosen for implementation because it provided free GPU resources that expedited the training process and facilitated faster acquisition of training results.

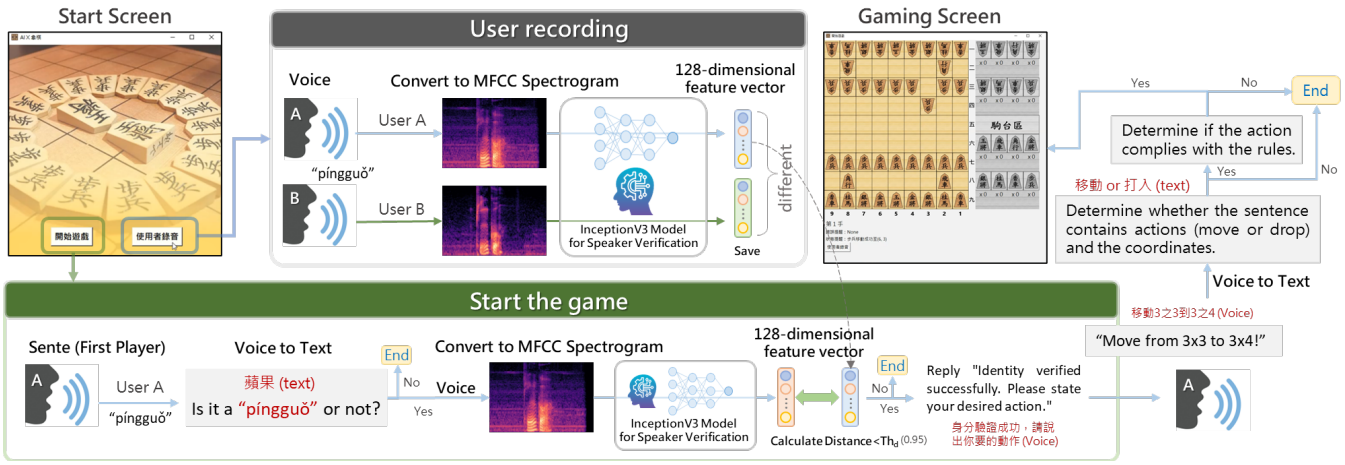


Figure 10. Process flowchart for integration of features.

D. *Integration of Features*

We applied the model to the shogi game, introducing an AI voice assistant feature to enhance interactivity and user experience. To verify the identities of two users, sente and gote respectively, we have a button on the initial screen before entering the game. Its function is to convert the user's voice into MFCC spectrograms, then pass them through a speaker verification model to obtain 128-dimensional feature vectors, and store them. Once the game starts, a user can call out "píngguǒ." First, the system checks if the word is "píngguǒ," and then it proceeds with the speaker verification process. This involves determining if the 128-dimensional feature vector is smaller than the Th_d and whether it belongs to sente (or gote). If all these checks are successful, the system responds with "Identity verified successfully. Please state your desired action." Only then can the user voice specific actions, moving the shogi pieces, that comply with the rules. The process flowchart for the integration of features is shown in Fig. 10.

Through speaker verification technology, we can verify user identities, rendering the game more personalized and engaging. Users can convey game instructions to the AI assistant via voice commands, such as moving pieces or making specific moves, eliminating the need for conventional keyboard or mouse operations. This intuitive interaction method makes the game smoother and amplifies its entertainment value. Incorporating the AI voice assistant feature gives users a unique gaming experience, allowing them to enjoy the pleasures of shogi in a novel way.

The provided link, https://youtu.be/TEKexuwlq8Y, directs you to a demonstration video. The segment within the first 3 minutes and 42 seconds features an introduction to the game and environment setup, while the portion following 3 minutes and 42 seconds showcases the integration of a voice assistant with the game of shogi.

IV. RESULTS

In speaker verification of the research methodology and steps, we employed InceptionV3 as the speaker verification model and utilized triplet loss as our loss function. The dataset consisted of training sets from 5 participants and test sets from 3 participants (A, B, and C). Fig. 11 illustrates the test loss during the training process, which converged at Epoch 10. Fig. 12 displays the accuracy curve for different $Th_d s$, revealing that the highest accuracy of 93.00% was achieved when the Th_d was set to 0.95.

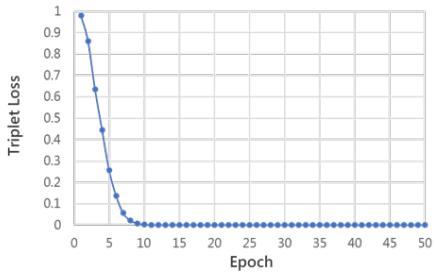


Figure 11. Test loss convergence during training.

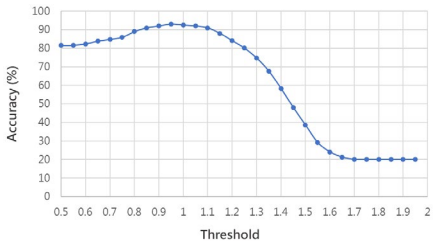


Figure 12. Accuracy curve for different $Th_d s$.

Table 1 displays the test set accuracy for participants A, B, and C at the Th_d of 0.95. Notably, Participant A exhibited an accuracy of only 52.50%. This observation can be attributed to two potential factors. Firstly, significant variations in the vocal characteristics of Participant A, such as notable pitch disparities, could pose challenges for the model in adapting its weights accordingly. Alternatively, the limited scope of our voice data collection might have contributed to premature overfitting of the model.

We also attempted six different models, namely NASNetMobile [11], EfficientNetV2B0 [12], Xception [13], ResNet50V2 [14], ResNet101V2 [14], and InceptionV3 [8]. Table 2 presents the comparison results for the state-of-the-art (SoTA). It was observed that ResNet101V2 outperformed the other models in terms of accuracy. Considering both detection time and the total params, this study still chooses InceptionV3 as our backbone model, which is the second-best option.

TABLE I. THE ACCURACY OF PARTICIPANTS A, B, AND C IN THE TEST SET.

		AA		***AB***		***AC***	
D_{True}	D_{False}	210	190	390	10	394	6
Accuracy		52.50%		98.00%		98.50%	
		BA		***BB***		***BC***	
D_{True}	D_{False}	390	10	370	30	400	0
Accuracy		98.00%		92.50%		100%	
		CA		***CB***		***CC***	
D_{True}	D_{False}	394	6	400	0	400	0
Accuracy		98.50%		100%		100%	
		Total					
D_{True}	D_{False}	3348		252			
Average accuracy		93.00%					

TABLE II. SOTA COMPARISON RESULTS.

Model	Accuracy	Th_d	Total params
NASNetMobile [11]	83.77%	0.75	4,405,012
EfficientNetV2B0 [12]	72.44%	0.50	6,083,280
Xception [13]	89.11%	0.90	21,123,752
ResNet50V2 [14]	92.67%	0.95	23,827,072
ResNet101V2 [14]	**95.56%**	0.80	42,888,832
InceptionV3 [8]	93.00%	0.95	22,065,056

V. DISCUSSION

This study develops a shogi game integrated with a voice assistant. An improved InceptionV3 trained with a triplet loss is used to differentiate user voices in the game through MFCC spectrogram images of input audio. The accuracy for distinguishing user voices reaches 93.00% when the Th_d is set at 0.95. This enables the successful pairing of two users and facilitates a voice-controlled game while adhering to its rules.

Regarding the issue of improving the performance of the voice verification model, we have future solutions to address it. Firstly, we plan to expand the dataset by collecting more voice data to address the current limitations of our study. Additionally, we will continue optimizing the model's training process to mitigate the overfitting issue. We believe these improvements will enhance model performance, making it more suitable for practical applications.

There is still room for improvement in this research. For instance, in terms of operations, we aim to enhance the way the AI voice assistant is invoked. Our goal is to transform it into a persistent background wake-up mechanism, rather than relying on the current button-click wake-up mechanism. Furthermore, the current system relies on specific trigger phrases for further interactions. In the future, we aim to implement user identity recognition based on the sound of their actions, eliminating the need for predefined trigger phrases. This innovation will provide users with a more convenient gaming experience.

In the next phase of our research, we will explore enhancing the real-time capabilities of the model. This will be achieved by employing model compression techniques, including quantization, pruning, and knowledge distillation, to accelerate the model's inference speed. Additionally, we plan to refine the model, broaden the range of data collection, and explore more diverse data augmentation techniques. Moreover, reinforcement learning is an aspect we haven't yet implemented in our research. This will empower the AI voice assistant to provide a wider array of strategies and suggestions, offering users a more diverse range of choices and experiences. Ultimately, this will introduce more challenging and personalized characteristics to the game system.

ACKNOWLEDGEMENT

This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 111-2622-8-259-001-TE2.

REFERENCES

[1] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 2016, pp. 5115–5119.

[2] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 4879–4883

[3] D. Snyder, D. G. Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 5329–5333.

[4] D. Snyder, D. G. Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in 2016 IEEE Spoken Language Technology Workshop (SLT), San Diego, CA, USA, 2017, pp. 999–1003.

[5] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," arXiv preprint arXiv:1705.02304, 2017.

[6] F. Lundh, "An introduction to tkinter," 1999. Available online: http://www.tcltk.co.kr/files/TclTk_Introduction_To_Tkiner.pdf (accessed on 27 August 2023).

[7] B. Logan et al., "Mel frequency cepstral coefficients for music modeling", International Society for Music Information Retrieval (ISMIR), vol. 270, pp. 1–11, 2000.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 2818–2826, 2016.

[9] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in Neural Information Processing Systems, pp. 1473–1480, 2006.

[10] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in 3rd International Conference for Learning Representations (ICLR), San Diego, 2014.

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018.

[12] Mingxing Tan and Quoc V. Le, "EfficientNetV2: smaller models and faster training," in 38th International Conference on Machine Learning (ICML), 2021.

[13] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in European Conference on Computer Vision (ECCV), pp. 630–645, 2016.