Speech to Text Recognition for Videogame Controlling with Convolutional Neural Networks

Joaquin Aguirre-Peralta, Marek Rivas-Zavala and Willy Ugarte[a]
Universidad Peruana de Ciencias Aplicadas, Lima, Peru

Keywords: Speech to Text, Machine Learning, Deep Learning, Gamification.

Abstract: Disability in people is a reality that has always been present throughout humanity and all nations of the planet are immersed in this reality. Being communication and interaction through technology much more important than ever, people with disabilities are the most affected by having a physical gap. There are still few tools that these people can use to interact more easily with different types of hardware, therefore, we want to provide them a playful and medical tool that can adapt to their needs and allow them to interact a little more with the people around them. From this context, we have decided to focus on people with motor disabilities of the upper limbs and based on this, we propose the use of gamification in the NLP (Natural Language Processing) area, developing a videogame consisting of three voice-operated minigames. This work has 4 stages: analysis (benchmarking), design, development and validation. In the first stage, we elaborated a benchmarking of the models. In the second stage, we describe the implementation of CNNs, together with methods such as gamification and NLP for problem solving. In the third stage, the corresponding mini-games which compose the videogame and its characteristics are described. Finally, in the last stage, the application of the videogame was validated with experts in physiotherapy. Our results show that with the training performed, the prediction of words with noise was improved from 43.49% to 74.50% and of words without noise from 63.87% to 96.36%

1 INTRODUCTION

Although technology is constantly developing and more software, office automation, therapeutic tools, etc. are being created (Hersh and Mouroutsou, 2019), the tools that help people with motor disabilities of the upper limbs are not enough to cover the needs this people require because the target audience is too small, as well as not being sufficiently lucrative for companies to be able to create more on a large scale. It is important for disability people in many aspects, like, on the employment side, by using these tools, people with disabilities could be at the same level in terms of competitiveness and be able to have a comparable value as a worker. On the legal aspect, these people can improve and increase their interaction with other people if they are given the right tools to improve their sociability (Nevarez-Toledo and Cedeno-Panezo, 2019). For the medical aspect, this kind of tools can help doctors to do analysis, therapies, rehabilitations in a more efficient way when dealing with people with disabilities (Garnica et al., 2020; Sundaram et al., 2019). Many solutions have emerged over time, proposing new tools with different types of methods to help these people, studies such as (Kandemir and Kose, 2022), who created a video game for people with disabilities of different ages and disabilities, which promotes physical activity, improving attention, improving emotion and sensorimotor coordination. Studies such as (Hwang et al., 2021) in which a human computer interface was designed and implemented specifically for users with motor disabilities, which works using eye tracking and lip movement technology. The project aims to promote employment, education, social life and leisure for this type of people. The method that was used for the development of the neural network that supports the videogame, is speech-to-text. This is an interdisciplinary subfield of computer science and computational linguistics that allows the recognition and translation of spoken language to text by computers (O'Shaughnessy, 2008). To carry out this project we have used a Convolutional Neural Network (CNN), trained with voice data sets created by us, which we have used using the wav2vec2-large-xlsr-53-spanish model with the Spanish language. We have used

[a] https://orcid.org/0000-0002-7510-618X

Aguirre-Peralta, J., Rivas-Zavala, M. and Ugarte, W.
Speech to Text Recognition for Videogame Controlling with Convolutional Neural Networks.
DOI: 10.5220/0011782900003411
In *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2023)*, pages 948-955
ISBN: 978-989-758-626-2; ISSN: 2184-4313

the training results of the neural network [1] and we have converted this data as the main input of the videogame created, so it is able to understand through the "speech-to-text" method and successfully interact with the game.

Our main contributions are as follows:

- We developed a convolutional neural network architecture focused on speech-to-text to be able to recognize words, convert them to text and be able to use them in a videogame.
- We developed 3 turn-based local multiplayer mini-games within the videogame that can process the data provided by the convolutional neural network.
- We present an analysis of the most promising results showing the feasibility of our project.

This paper is organized as follows. Therefore, in Section 2, similar solutions implemented in different fields for people with disabilities will be explained. In Section 3, we will explain the definitions of the technologies related to convolutional neural networks, the details of the architecture chosen for the solution and some terms such as NLP, CNN, speech-to-text and gamification. Finally, Section 4 will detail the experiments and their results. To conclude with Section 5

2 RELATED WORKS

Next, we will briefly discuss the different types of solutions used for serious games similar to the ones in this project using Gamification and NLP.

In (Kandemir and Kose, 2022), the authors propose the use of 3 different types of human-computer interaction games with the use of a Kinect sensor based on physical activity to improve attention, emotion and sensorimotor coordination. Unlike them, which seek to increase the value of attention of the participants and the willingness to continue with therapy sessions and tests, we also seek, through gamification, to improve the interaction of people with others who suffer from motor disability of the upper limbs of people with others.

In study (Hwang et al., 2021) the authors proposed, a nine-point fixation experiment to assess the efficacy of gaze data smoothing. To do this, they conducted an on-screen keyboard-based Chinese text input experiment with four different keyboard sizes to assess the influence of keyboard size on text input rate. This work aims to facilitate the interaction of disabled users with their computers for various communication applications using eye tracking. In the same way, we want to improve interaction for users with upper limb motor impairment but using natural language processing, speech recognition and speech-to-text conversion.

Within the article (Mavropoulos et al., 2021) an attempt is made to unify some advanced functionalities in IoT with a human-computer verbal interaction to improve the effectiveness of treatments/medications, operating expenses, optimize personnel management, among other benefits in order to limit the interaction between caregiver and patient to only what is necessary. In contrast to this work, although we want to facilitate human-computer interaction for people with motor disabilities of the upper back, they use this technology to provoke in hospital/clinic patients a sense of independence that will help them mentally to his speedy recovery.

In the study (Anjos et al., 2020) a comparison of the classification of sibilant consonants with neural networks is carried out and, by gamifying this technique, it seeks as a result that children do not lose interest and motivation to do their speech exercises. Regarding their work, they use CNN to convert matrixes into data samples that, through gamification methods, are transformed into serious games to help children who cannot pronounce the correct sibilant consonants. For our part, we use inputs in "wav" format with stereo channel that, we then process to obtain data for the training of the neural network and that later becomes input for the correct operation of our game that helps people with motor disabilities of the upper limbs to increase the interaction between them.

In (Song et al., 2006) the author creates a system which allows creating 3D cartoons from a text input, which is based on Unity, as well as the ability to automatically configure the environment without the need for the input to be a script. Although both works use NLP principles, in contrast to what we do, this studio is used to create 3D cartoons through the NLP and Graphic methods. Instead, our work uses NLP and Speech to Text to understand human language, interpret and process it to finally convert it into text and use it as input for the main functionality of the game.

[1] https://huggingface.co/jonatasgrosman/wav2vec2-large-xlsr-53-spanish

3 VIDEOGAME CONTROLLING FROM SPEECH TO TEXT RECOGNITION

3.1 Preliminary Concepts

Machine learning development have given us very significant advances that help improve aspects of our lives and make everyday tasks easier. Like pattern recognition, predictive behavior, facial recognition, etc. and for what this document is about, word recognition well known as speech-to-text. The model chosen for this work was wav2vec2-large-xlsr-53-spanish, which is based on Convolutional Neural Networks (CNN).

Convolutional Neural Networks. According to Keiron O'Shea and Ryan Nash, CNN is a type of Artificial Neural Network (ANN) specialized in recognizing patterns in images. Like the other ANNs, its neurons self-optimize and update their weights. In the last layer lie loss functions associated with the classes. CNNs are made up of three types of layers, these are convolutional layers, pooling layers, and fully connected layers. These layers have to be together to consider the architecture as CNN. Figure 1 shows a CNN architecture (O'Shea and Nash, 2015).

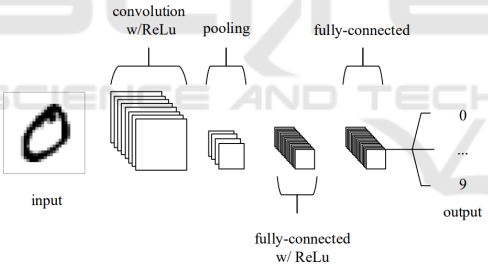


Figure 1: A simple CNN architecture, composed of only five layers (O'Shea and Nash, 2015).

Natural Language Processing (NLP). According to Ann Neethu Mathew et al. Neethu Mathew et al., NLP is focused on how to train a machine to learn and process human language using computational and linguistic technology. (Jung et al., 2019).

Gamification. According to Oriol Borrás Gené, gamification is the use of mechanics, techniques or some element related to game design in a context in which it is not, with the goal of increasing user engagement and with the goal of solving or helping to solve a problem. Figure 3 shows some of the characteristics that Gamification incorporates.

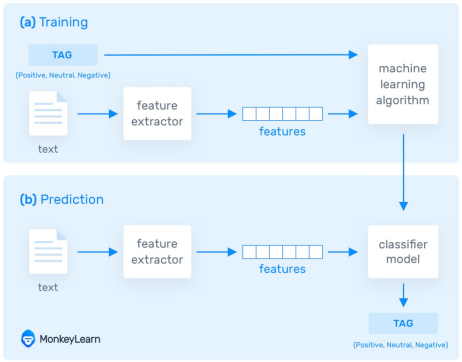


Figure 2: Image of a simplified NLP process[2].

Figure 3: Characteristics that make up gamification[3].

3.1.1 Wav2vec2.0 Structure

Below in Figure 4 it is shown that wav2vec2.0 is made up of several convolutional and self-attending layers. This is a common type of structure recently used in the end-to-end ASR family of models. The job of the convolution layers is to reduce the speech sample X and generate a more compressed latent representation Z. In detail, Z are the raw audio signals and X are sampled at 16kHz. With a lead time of 20 ms and a receptive field of 25 ms. Self-care layers are responsible for creating C-contextualized representations and capturing high-level content Z. These self-attention layers have a high level of context dependency modeling, which allows the model to make the correct decisions during the following contrastive training given the masked Z (Yi et al., 2020).

3.2 Method

In this section we will describe our contributions. Section 3.2.1 presents the workflow of our project,

[2]https://monkeylearn.com/blog/what-is-natural-language-processing/

[3]https://blog.efmdglobal.org/2021/09/08/redefining-failure-teaching-digital-skills-with-gamification/

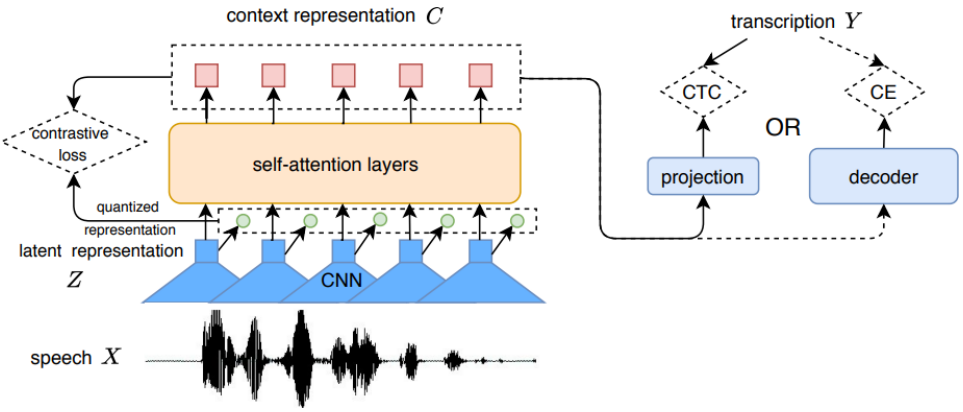


Figure 4: Left: the structure of wav2vec2.0 and the corresponding self-training criteria. It contains a stack of convolution layers and self-care layers; Right: Two decoding branches applying wav2vec2.0 to ASR tasks with extra projection or decoder, which is trained with CTC or cross-entropy loss respectively (Yi et al., 2020).

which allows us to interact with the videogame from the trained results of the implemented CNN. Section 3.2.2 shows our data preprocessing method to obtain audio that can be useful when training the convolutional neural network. Section 3.2.3 corresponds to explain the configuration and the training procedure of the CNN with the data obtained. Section 3.2.4 explains the link of the neural network outputs with the videogame speech controller to create the human-computer voice interaction.

3.2.1 Workflow

The objective of our proposed model is to convert the audio into text so that it works as the data input for the correct operation of the videogame. We created our own dataset from the voices of 62 volunteers. Which contains more than 11718 audios with durations of up to 3 seconds. The words found in this dataset represent all the inputs and combinations that we needed for the game to work in its entirety.

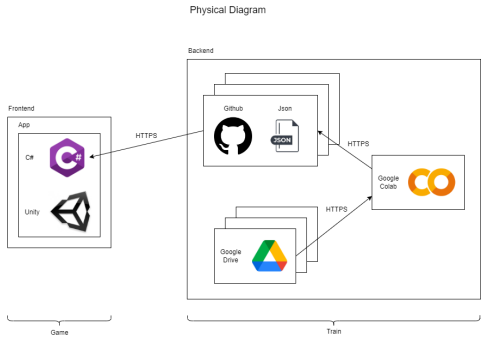


Figure 5: Physical diagram of our project.

Figure 5 shows the workflow of our project, which starts in the “training” section. Our neural network is stored in the shared file platform ”Google Drive” which is coded in python in the ”Google Collab” platform. The network training results were saved in JSON format to a GitHub repository. Finally, the video game was developed with the Unity game engine, which connects to the repository and uses the JSON as input to function.

3.2.2 Data Preprocessing

The dataset consisted of 27 words in spanish, which are divided into:

- Words that designate games or actions such as “michi”, “memoria”, “damas”, “salir”, “pausa”, “continuar”, “reiniciar” and “cerrar”.
- Letter names such as “a”, “b”, “c”, “f”, “g”, “h”, “i”, “j”, “k”, “l” and “m”.
- In addition to the names of the first eight numbers “1”, “2”, “3”, “4”, “5”, “6”, “7” and “8”.

Very few audios of the people who helped us had pronunciation problems, too much noise or bad recordings; therefore, those audios were not used in the train or test phases.

3.2.3 Neural Network Model and Training

As seen in section 3.1.1 wav2vec is based on multiple convolutional layers. And then, the training process consisted of providing the model with noise audio of the keywords, this with the intention that the model learns to identify and separate the words from the noises.

We took the 27 pre-recorded words and combined them with 7 different noises, obtaining 7 variants of the same audio, thus allowing the model to identify only the words. This process is known as Augmentation (Laptev et al., 2020). We listened and validated

the correct pronunciation of every audio, letting just 8.06% of the data obtained as not valid to proceed with the training and testing. The training was carried out in the Google Colab platform, we only used audios of 57 people in the training phase, training and saving the models each time. In the end, the model was retrained with 10,773 audios.

3.2.4 Video Game Input and Human-Computer Interaction

After the neural network has trained using the previously described dataset and has predicted the words with a high percentage (74.50% in the case of noise and 96.36% in the case of no noise), these were saved in a JSON and exported to the GitHub virtual repository. These are then used by the game's Speech Controller, which is responsible for parsing the JSON extracted from the web so that the Unity engine can interpret the data. With this said, the user will be able to use his microphone to say a word within the catalog of trained words and interact with his voice with the videogame.

4 EXPERIMENTS

In this section, we are going to detail the hardware and software specifications that we have been using in the project with their respective configurations in order to replicate the experiments. In addition to showing the results obtained and discussing their interpretation in the evaluation of our metrics.

4.1 Experimental Protocol

Development environment: The neural network was programmed and the models were trained in Google Collaboratory, which provides us with an Intel Xeon CPU at 2.20 GHz with 13 GB of RAM, a Tesla K80 accelerator and 12GB of GDDR5 VRAM. On the other hand for the videogame, the version of Unity 2021.3.2f1 with the C# programming language was used to program the three minigames with voice recognition.

As we mencioned in section 3.2.2, 10773 audios were used as a dataset for the different phases of training our neural network model.

We separate the training by phases to keep track of the progress of each phase. Each phase has in an average between 945 and 1512 audios trained, in a limited way because of restrictions of the free version of Google Collab you can only train for a limited time and amount. Also, it was trained according to the number of audios that were available at that time.

In the first scenario, we have 4 test audio groups, 1 for training and 3 for testing:

- **Group A.** It is the original data collected, mixed with 7 different noises, being a total of 10773 audios.
- **Group B.** It is the original data collected, it is not with noise, it was not used in the training and it consists of 1539 audios.
- **Group C.** It is data collected from 5 other people, it is mixed with the 7 noises and in total it consists of 945 audios.
- **Group D.** It is the data from which Group C starts, but without the 7 noises and consists of 135 audios.

On the other hand, the video game was created in Unity 3D with full functionality by mouse, which later had to be transformed into voice commands, so that the use of people with motor disabilities can be validated. This environment will load the results of the GitHub in which the results of the trained models were uploaded, which will serve as input for the operation of the videogame.

The neural network code found in Google Drive/Google Collaboratory can be found at the following links: https://goo.su/nia1a2 and https://goo.su/5MONA. Meanwhile, the video game repository with all the validated and complete features is on Github: https://github.com/Xerathox/GamiHelper.

4.2 Results

4.2.1 Experiment Data Sets

As previously mentioned in section 4.1 the dataset in total consisted of 11,718 audios, which went through a preprocessing process mentioned in section 3.2.2. Of these, 8.06% were replaced by synthetic audios. In order to reproduce the training process, access to the Python code file in Google Drive was provided in section 4.1. First, the necessary files for the training must be uploaded to your own Google Drive, which are the audios and a "csv" format file with a column called "file" in which the path of each training audio in your Google Drive is specified and the other column called "text" which specifies the word it represents. Second, you must change the paths of the training file to match the paths of the training "csv" file, the path where the model will be saved after training and also load a previously saved model.

4.2.2 Experiment Configuration

In the comparison with our work we have the following models:

DeepSpeech. The first hyper-parameter setup we had was 10 epochs and 10 layers.

- With our Group B test we had a WER of 100% (0% accuracy).
- With our Group C test we had a WER of 100% (0% accuracy).
- With our Group D test we had a WER of 100% (0% accuracy).

In the second hyper-parameter configuration we had 20 epochs and 20 layers.

- With our Group B test we had a WER of 96.30% (accuracy of 3.70%).
- With our Group C test we had a WER of 96.30% (accuracy of 3.70%).
- With our Group D test we had a WER of 100% (accuracy of 0%).

Tensorflow Model from 0. The hyper-parameter configuration in this case was in the layers:

- InputLayer: Input layer.
- Resizing: Resizes the image to 32x32 pixels.
- Normalization: A normalization layer with default parameters.
- Conv2D: A 2D Convolution layer with filter of 32, kernel size of 3 and activation function “relu”.
- Conv2D: A 2D Convolution layer with filter of 96, kernel size of 3 and activation function “relu”.
- MaxPooling2D: A MaxPool layer with default parameters.
- Dropout: A Dropout layer with rate of 0.25.
- Flatten: A Flatten layer with default parameters.
- Dense: A dense layer with units of 256 and activation function “relu”.
- Dropout: A Dropout layer with rate of 0.5.
- Dense: A dense layer with units of the same number of labels.

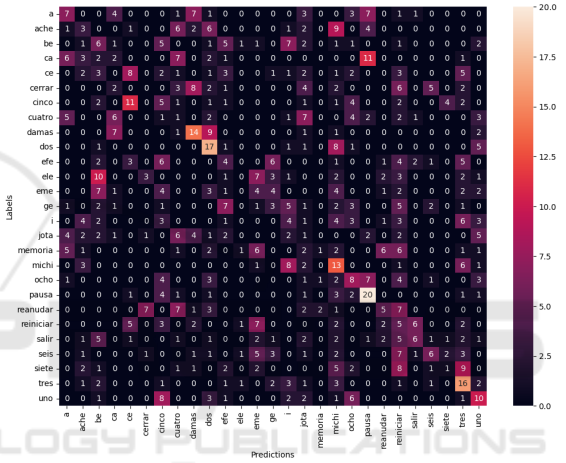
This is a special case because this model uses validation data. Two tests were carried out: Each time the model is trained a different model can be obtained which can give different accuracies to the same test, this means that if the same steps are followed to recreate the experiment it may not give the same accuracy as what we present below. For Case 1 we obtained in 5 training and testing attempts 19.26%, 13.33%, 15.56%, 11.11% and 14.81%. We will consider an average of 14.81% accuracy. For Case 2, 16.51%, 17.99%, 13.76%, 15.56% and 18.10% were obtained in 5 training and test attempts. We will consider an average of 16.38% accuracy.

This would be better visualized in Table 1.

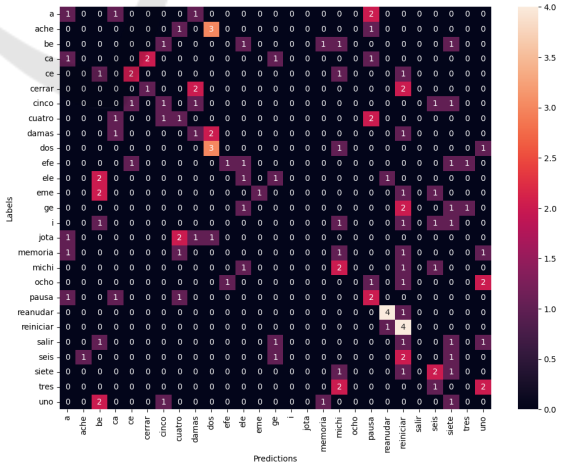
Table 1: Tensorflow Model Test on average.

	Train #Audios	Validation #Audios	Test #Audios	Accuracy on average	Try 1	Try 2	Try 3	Try 4	Try 5
Case 1 Noisy Data	9,828	945	945	14.81%	19.26%	13.33%	15.56%	11.11%	14.81%
Case 2 Clean Data	1,404	135	135	16.38%	16.51%	17.99%	13.76%	15.56%	18.10%

Figure 6a shows a heat map pertaining to the accuracy of this model when trained and tested with noisy data. And Figure 6b shows another heat map, but this time when trained and tested with data without noise.



(a) For noisy audio and 17.99% of accuracy.



(b) For clean audio and 19.26% of accuracy.

Figure 6: Heatmaps of Tensorflow models trained and tested with audios.

Table 2: Azure Cognitive Services Model.

	Group B		Group C		Group D	
Weight	Error rate	Accuracy	Error rate	Accuracy	Error rate	Accuracy
70%	4.22%	95.78%	13.33%	86.67%	2.96%	97.04%
30%	4.35%	95.65%	15.87%	84.13%	3.70%	96.30%
10%	5.13%	94.87%	18.52%	81.48%	4.44%	95.56%

Azure Cognitive Services Model. The configuration of hyper-parameters in this case, the weight of the new training data. It was tested for the cases where the training weights were 70%, 30% and 10%. This is shown in Table 2.

4.2.3 Baselines

We re-trained the wav2vec2 model mentioned above with our data from "Group A" in 14 different versions, at 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25 and 30 epochs to have evidence of how many epochs learn best. Table 3, shows the results of each test. Due to

Table 3: Groups B, C and D tested with each epochs.

Model	Group B	Group C	Group D
Standard	63.87%	43.49%	65.93%
5 epochs	95.06%	70.26%	98.52%
6 epochs	94.87%	71.22%	97.78%
7 epochs	95.32%	72.49%	98.52%
8 epochs	85.06%	60.74%	87.41%
9 epochs	96.30%	72.80%	98.52%
10 epochs	96.30%	73.44%	98.52%
11 epochs	95.71%	73.44%	97.78%
12 epochs	81.48%	57.14%	80.74%
13 epochs	96.36%	74.50%	97.78%
14 epochs	82.20%	57.04%	83.70%
15 epochs	80.12%	55.24%	79.26%
20 epochs	88.30%	64.55%	89.63%
25 epochs	77.58%	57.99%	75.56%
30 epochs	77.45%	59.05%	76.30%

what is presented in the table 3 we can conclude that the hyper-parameter "number of epochs" with which the wav2vec 2.0 model, presented a better learning was with 13 epochs. It has better precision in Grups B and C. Group D has less weight as it has fewer audios.

4.3 Discussion

Observations for the DeepSpeech model:

- Very uneven results were achieved; the results of the implemented neural network returned us a percentage of 0% and 100%.
- Training with a lot of data with little training and few convolution layers, the results tend to return 0% accuracy. However, with little data and a lot of training, the model tends to return 100% accuracy making it an unstable model.
- We could modify the hyper-parameters or increase the epochs and layers in order to receive more accurate results, however, we didn't have much time at the date the models were being made and trained for comparisons.

Observations for the Tensorflow model from 0:

- A lot of data is needed when you want to start training a speech-to-text model.
- By using the free version, this model was limited and only a specific number of outputs could be given (labels).
- The model couldn't sometimes differentiate speech from noise, the model compares the waveform of the spectrogram and that interference makes it much more difficult to find the word to which each sound belongs.
- The size of the duration of every audio long between 1 second to 4 seconds made learning very difficult, as it seems to compare the audio waveform of the spectrogram very simply. If it detects that an audio has the sounds of 2 audios in different positions it can consider them as different words even though they are not, reducing the accuracy percentage.

Observations for the model in Azure Cognitive Services:

- Allows you to set the language of the audio input.
- It is a very powerful model, that is evidenced by the high accuracy rate.
- This model is a paid model compared to wav2vec2 which is free.

Main Conclusions.

- Pertaining to the recommendations of the interviewees:
  - The game has an easy learning curve, quite intuitive to use, intuitive at first use.
  - Simple but plain but attractive and colorful design.
  - It could be focused on another target audience such as children and elderly.
  - More games could be added.
  - The letters indicating how to play within the mini-games could be more detailed to help guide the player a little more when starting the game.
  - Some mentioned that the lighting looked unclear.
  - The large minority criticized the choice of colors, as better color combinations could be chosen for the game.

  - 20% did not understand how to get started as it was only mentioned to them that the game worked by voice commands.
- Belonging to the wav2vec2 model:
  - It uses the torch library which is not compatible with Windows, it can only be trained by virtual machines, other operating systems or by Google Colab.
  - This model does a good job to isolate the speech from the noise, it also does not lose accuracy due to the size of the audio, compared to the Tensorflow model that needed the duration of the audios to be very close to each other for example.
  - The language of the input cannot be specified, that is done by the same model internally, which can cause a precision problem for a syllable, word or phrase that sounds the same in two different languages. For example, "i" (spanish) and "e" (english) or "ji" (spanish) and "he" (english).

5 CONCLUSIONS

By implementing the W2V2 convolutional neural network model that transforms speech to text and increasing the accuracy percentage of the trained words, the computer can effectively understand the keywords provided to it so that they can be used in this case as input for the video game.

In the future, in terms of the video game, we would like to increase the catalog of games, add difficulty levels, some visual aspects such as color palette or more detailed instructions on the screens. On the other hand, as the speech-to-text method is such a versatile topic, similar techniques could be applied in other areas, such as voice-controlled office tools, cell phone applications to understand the functionalities of the cell phone for people with other disabilities, etc. As long as you have a good speech-to-text model or implement one and train it long enough and accurately, any model is valid. Evenmore, the user experience can be improven greatly with animation models (Silva et al., 2022) or story creation (Fernández-Samillán et al., 2021).

REFERENCES

Anjos, I., Marques, N. C., Grilo, M., Guimarães, I., Magalhães, J., and Cavaco, S. (2020). Sibilant consonants classification comparison with multi- and single-class neural networks. *Expert Syst. J. Knowl. Eng.*, 37(6).

Fernández-Samillán, D., Guizado-Díaz, C., and Ugarte, W. (2021). Story creation algorithm using Q- learning in a 2d action RPG video game. In *IEEE FRUCT*.

Garnica, C. C., Archundia-Sierra, E., and Beltrán, B. (2020). Prototype of a recommendation system of educative resources for students with visual and hearing disability. *Res. Comput. Sci.*, 149(4):81–91.

Hersh, M. A. and Mouroutsou, S. (2019). Learning technology and disability - overcoming barriers to inclusion: Evidence from a multicountry study. *Br. J. Educ. Technol.*, 50(6):3329–3344.

Hwang, I., Tsai, Y., Zeng, B., Lin, C., Shiue, H., and Chang, G. (2021). Integration of eye tracking and lip motion for hands-free computer access. *Univers. Access Inf. Soc.*, 20(2):405–416.

Jung, S., Son, M., Kim, C., Rew, J., and Hwang, E. (2019). Video-based learning assistant scheme for sustainable education. *New Rev. Hypermedia Multim.*, 25(3):161–181.

Kandemir, H. and Kose, H. (2022). Development of adaptive human-computer interaction games to evaluate attention. *Robotica*, 40(1):56–76.

Laptev, A., Korostik, R., Svischev, A., Andrusenko, A., Medennikov, I., and Rybin, S. (2020). You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation. In *CISP-BMEI*, pages 439–444. IEEE.

Mavropoulos, T., Symeonidis, S., Tsanousa, A., Giannakeris, P., Rousi, M., Kamateri, E., Meditskos, G., Ioannidis, K., Vrochidis, S., and Kompatsiaris, I. (2021). Smart integration of sensors, computer vision and knowledge representation for intelligent monitoring and verbal human-computer interaction. *J. Intell. Inf. Syst.*, 57(2):321–345.

Nevarez-Toledo, M. and Cedeno-Panezo, M. (2019). Application of neurosignals in the control of robotic prosthesis for the inclusion of people with physical disabilities. In *INCISCOS*, pages 83–89.

O'Shaughnessy, D. D. (2008). Invited paper: Automatic speech recognition: History, methods and challenges. *Pattern Recognit.*, 41(10):2965–2979.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *CoRR*, abs/1511.08458.

Silva, S., Sugahara, S., and Ugarte, W. (2022). Neuranimation: Reactive character animations with deep neural networks. In *VISIGRAPP (1: GRAPP)*.

Song, I., Jung, M., and Cho, S. (2006). Automatic generation of funny cartoons diary for everyday mobile life. In *Australian Conference on Artificial Intelligence*, volume 4304 of *Lecture Notes in Computer Science*, pages 443–452. Springer.

Sundaram, D., Sarode, A., and George, K. (2019). Vision-based trainable robotic arm for individuals with motor disability. In *UEMCON*, pages 312–315. IEEE.

Yi, C., Wang, J., Cheng, N., Zhou, S., and Xu, B. (2020). Applying wav2vec2.0 to speech recognition in various low-resource languages. *CoRR*, abs/2012.12121.