Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)
7-8 October 2021, Mauritius

2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) | 978-1-6654-1262-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICECCME52200.2021.9590892

Audio CAPTCHA for Visually Impaired

Angelo Mathai
Computer Engineering
Vishwakarma Institute of Technology
Pune, India
angelo.mathai18@vit.edu

Atharv Nirmal
Computer Engineering
Vishwakarma Institute of Technology
Pune, India
atharv.nirmal18@vit.edu

Purva Chaudhari
Computer Engineering
Vishwakarma Institute of Technology
Pune, India
purva.chaudhari18@vit.edu

Vedant Deshmukh
Computer Engineering
Vishwakarma Institute of Technology
Pune, India
vedant.deshmukh18@vit.edu

Shantanu Dhamdhere
Computer Engineering
Vishwakarma Institute of Technology
Pune, India
shantanu.dhamdhere18@vit.edu

Pushkar Joglekar
Computer Engineering
Vishwakarma Institute of Technology
Pune, India
pushkar.joglekar@vit.edu

***Abstract*—Completely Automated Public Turing Tests (CAPTCHA) have been used to differentiate between computers and humans for quite some time now. There are many different varieties of CAPTCHAs - text-based, image-based, audio, video, arithmetic, etc. However, not all varieties are suitable for the visually impaired. As time goes by and Spambots and APIs grow more accurate, the CAPTCHA tests have been constantly updated to stay relevant, but that has not happened with the audio CAPTCHA. There exists an audio CAPTCHA intended for the blind/visually impaired but many blind/visually impaired find it difficult to solve. We propose an alternative to the existing system, which would make use of unique sound samples layered with music generated through GANs (Generative Adversarial Networks) along with noise and other layers of sounds to make it difficult to dissect. The user has to count the number of times the unique sound was heard in the sample and then input that number. Since there are no letters or numbers involved in the samples, speech-to-text bots/APIs cannot be used directly to decipher this system. Also, any user regardless of their native language can comfortably use this system.**

Keywords—Audio CAPTCHA, CAPTCHA, Generative Adversarial Network (GAN), Short-time Fourier Transform (STFT), Visually impaired

I. Introduction

The World Health Organization (WHO) claims that globally the number of people of all ages visually impaired is estimated to be 285 million, out of which 39 million are blind [1]. WHO also states that with the growing population and aging, the number is expected to increase in the coming years? Thus the number of blind people is large. With the increased interaction of the visually impaired with technology, there has also been a significant increase in cyber threats and impersonation. It is not feasible for each entity to develop a high-security layer to avoid spambots as adding more features for security can also create accessibility issues for users. Hence CAPTCHA remains relevant for this purpose.

CAPTCHAs are widely used by web applications worldwide for security and privacy. They are used to differentiate between paper real users and automated/spambots. They are intended to provide challenges that are difficult for computers to perform but relatively easy for humans to solve. Some commonly used CAPTCHAs are text CAPTCHA which requires the user to identify the text usually written in a distorted manner, image CAPTCHAs which require the user to identify specific objects in multiple images, arithmetic CAPTCHAs which ask the user to solve some simple arithmetic equations, audio CAPTCHAs where the user must identify the spoken alphabets or numbers, etc. Except for the audio CAPTCHA, most of these techniques cannot be used by the visually impaired.

To overcome these obstacles our proposed solution doesn't require the user to identify letters or numbers thus defeating the language barrier. It asks the user to count the number of unique sounds which play with GAN-developed music, which makes it easier for the visually impaired as well as difficult for spam bots to break the CAPTCHA.

II. Literature Review

First, before proceeding with our approach to develop an audio-based CAPTCHA we went through the previous works that have been done to analyze their drawbacks and propose a better solution.

The following table depicts various CAPTCHAs commonly used and their limitations.

TABLE I. Limitations of different types of CAPTCHAs used today

Sr.No	Types Of CAPTCHAs	Limitations
1.	Text-based CAPTCHA	1. Simple CAPTCHA can be identified by OCR techniques. 2. Complex CAPTCHA can be too difficult for the user to understand due to i. Multiple fonts ii. Font size iii. Wave motion

2.	Image-based CAPTCHA	It is difficult for users to identify CAPTCHA who have low vision
3.	Audio-based CAPTCHA	1. It is accessible in English Hence the end client must have a far-reaching English vocabulary. 2. Character that has a comparative sound.
4.	Video-based CAPTCHA	For videos of large size, users may face problems downloading such videos.
5.	Puzzle-based CAPTCHA	Users in puzzle-based CAPTCHAs take more time to solve puzzles.

The existing audio CAPTCHA which is present consists of an alphanumeric sequence that is played along with some noise. In the popularly used reCAPTCHA [2], the user then has to type the correct input alphabets/numbers, and then the CAPTCHA is successfully solved. These inputs have some background noise of crowd or babble. The problem with current audio CAPTCHAs is that it requires the user to understand specific linguistic syllables. It is also relatively easier for spam bots to crack this type of CAPTCHAs as they can easily distinguish syllables. On the contrary, some surveys also point out that Audio CAPTCHAs are more difficult than visual CAPTCHAs, especially for blind users who use screen-reader programs. These programs tend to talk over the audio CAPTCHA as the user navigates between the audio playback controls and answer box, frustrating the user's ability to hear the CAPTCHA.

A real-time audio-based CAPTCHA in HearAct [3] enables easy access for blind or visually impaired users. The user is supposed to listen to the sound of the "sound maker" and then identify what it is. The HearAct mechanism will then identify a word and the user will be needed to analyze a word and determine if it has that stated letter or not. If the letter is found, the user will tap or else swipe if not found. The results from this study show a success rate of 82.5% for solving the CAPTCHA. Some of the limitations that they came across with this approach were the lack of variations when it came to the audio samples and the need to cover a wide range of different users.

In another related work, a CAPTCHA strategy, a basic numerical problem is presented by predefined examples [4] and changed over to speech by using a Text-To-Speech (TTS) framework. Then at that point, a sound is played for the user and they should enter the appropriate response of the sound. This response could be a simple answer to an audio file asking the sum of two numbers, for example, if the audio file says, "Sum of 4 and 5" your response should be "9". Here we can recognize that the limitation this method has is that it requires the user to perceive the sound, understand the matter and then have the ability to solve it. This could work only if addressing the inquiry was straightforward which would require it to be in a form of simple sentences and language that could be understood by the visually impaired and not a mathematical problem.

SoundsRight CAPTCHA [5] devises a new CAPTCHA prototype that is completely audio-based. The user needs to identify audio clips of environmental sounds or sounds related to a specific concept such as train, animal, rain, etc. within a limited time frame of 10 seconds. The results obtained from the usability testing were positive when it came to ease of use. Also due to the implementation of a specified time limit, the CAPTCHA provides improved security protection from bots. One of the major limitations they came across with this approach was the limited array of audio profiles that can be easily identified by humans. Also, though we haven't performed explicit experiments, the level of abstraction in the underlying recognition problem suggests that it might be solvable efficiently using *deep* neural networks. We note that there is no mixing or layering of different audios in this approach.

There has been another CAPTCHA technique that is intended for the visually impaired [6]. In this strategy, a sound (in the form of a general question) for example, "what is the color of an apple?" is played for the client and they are approached to say a word, which would be considered as the response to the sound clip heard. At that point, the client's reaction - which is an audio document then is checked if it is the word required for clearance and not produced by a computer or a bot. This technique doesn't need any screen and can be utilized effectively by non-visually impaired users. The limitation of this is that it is not always easy to recognize if the user is a bot or not, also the audio file sent by the user could be corrupted by background noise and other environmental aspects, this approach also takes into consideration that a blind person can speak fluently.

The authors of the paper Breaking Audio CAPTCHAs [7], to test the security of audio CAPTCHA's made use of several machine learning techniques to break them. From their findings, they were able to come to several significant conclusions when it comes to the security of audio CAPTCHA. Their study claims that audio samples containing longer phrases as their solutions and multiple different speakers tend to be difficult to break. Also, the use of background noise which consisted of human voices was more effective when compared to running water which is static.

After reviewing all the above research, we have devised an approach that can overcome most of the current limitations

III. Methodology

In the solution that we propose, the user must count the number of beats/bangs of a unique sound made by percussion instruments (bell, drum, hammering, etc). Any other unique sound can also be chosen. There will be an additional layer of white noise along with this we also add a music sample that has been generated through a GAN algorithm making it hard for a bot or an API to detect and pass the authentication process.

A. GANs

Generative Adversarial Networks or GANs are considered to be highly versatile and thus able to generate unorthodox content. They are used in various fields like generating examples of image datasets, 3D object generation, photo blending, etc. Nowadays GANs are used in the creation of state-of-the-art music pieces. MidiNet and MuseGAN are examples of research on using GAN models to generate music.

There are many APIs' capable of detecting songs/music. The reason to add a GAN-produced music layer is to randomly generate music patches making it hard for the APIs to identify and distinguish the original unique sound (percussion instrument

beats or similar) and count its beats for authentication. On the other hand, it is easier for a human to distinguish them. The wave plots in the next section describe their nature and help in understanding the reason.

A simple C-RNN–GAN model [13] had been trained over a small dataset to create the samples. The composition of the generative adversarial neural network C-RNN–GAN is based on an LSTM neural network for generating music. In GAN, the generator and discriminator (called adversaries) are two different recurrent neural networks. The generator and discriminator are alternatively trained. The discriminator takes in real data as well as fake data generated from random noise. The discriminator in GANs acts as classifiers. It classifies real data from the fake data which is generated by the generator. On the other hand, the generator's task is to generate fake data from random noise by incorporating the feedback from the discriminator.

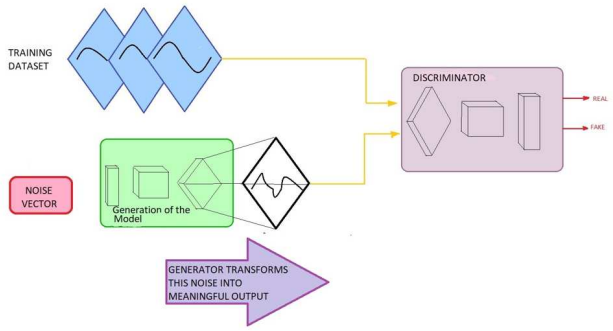

Fig. 1. GAN block diagram

When a generator gets trained properly, the discriminator finds it difficult to classify between real and fake data, and its accuracy decreases. We trained the GAN model over a dataset of 30 MIDI files of piano chords. The binomial cross-entropy loss function was used. The generator and discriminator loss converge near 300 epochs and diverge a little above 3500 epochs. The model can be effectively improved by increasing the number of samples in the dataset and pre-training the discriminator with random noise.

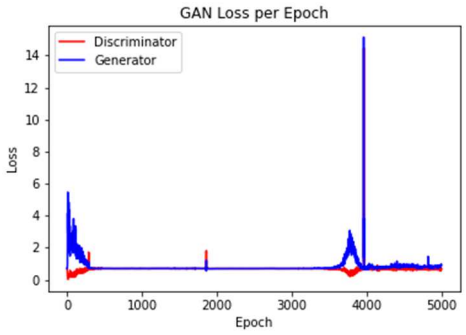

Fig. 2. GAN Loss per epoch

B. *Waves*

We have used the Short-Time Fourier Transform (STFT) to visualize the spectrum of different audio samples we studied. The Fourier transform assumes that the signal is analyzed over a duration of time, which means that there can be no concept of time in the frequency domain and similarly no concept of frequency changing over time. These two domains, frequency, and time cannot be mixed as they are orthogonal to each other. There exist some audio signals whose frequency component changes with time. The Fourier transform fails to show us this change of frequency with time. Using the Short-Time Fourier Transform we can evaluate the frequency change of audio signal over time.

In Short-Time Fourier Transform we do not perform the Fourier transform across the whole duration of the signal but rather, we consider small segments or chunks of the signal called frames and then apply a Discrete Fourier transform for each frame. To enhance the results, frames are overlapped, and each frame is multiplied by a window that is tapered at its endpoints. Window functions help to truncate the audio frame and make the signal smooth. Choosing the right window functions can reduce the number of filter bands in meeting the requirements of the situations.

We have used the Hann window in our project. The formula used is:

$$S(i) = w(i) * x(i) \quad (1)$$

$$W(n) = (0.5 *(1 - \cos((2*PI*n)/(N-1))),$$

$$0 \leq n \leq N-1 \quad (2)$$

Where S(i) is the output signal, x(i) is the input signal, w(n) is the Hann windowing function, N is the length of the window and n is the location of the sample in the window.

The samples which we created were analyzed individually. A spectrogram is a visual representation of frequencies of a given signal with time. In a spectrogram representation plot — the X-axis represents the time; the Y-axis represents frequencies, and the colors represent the magnitude (amplitude) of the observed frequency at a specific time. Bright colors represent strong frequencies.

We have depicted here four graphs showing how each audio file would look when viewed with frequency on the Y-axis and time on the X-axis.

- In Fig. 3 we have an audio file with bell beats as it is and no noise is added. (Counting the beats of similar random percussion instruments is the aim of the authentication). Every time audio has a discrete nature and has a level of periodicity with no background disturbance it becomes easier for bots to recognize the beats in the original time-domain waveform itself.
- Next is Fig. 4 with bell beats and a layer of noise, by the graph, it is observable that the beats somewhat blend with the background noise. However such noise layers are easily identified and eliminated by bots and APIs, thus making it ineffective. The reason bots can get rid of the noise is that such systems can select the optimal audio stream for the speech recognizer by using top-down knowledge from the trigger phrase detectors. A smart bot mostly focuses on audio de-reverberation and audio suppression

- In Fig. 5 the sample has phone rings layered with bell beats. Here we see that the phone rings at a constant frequency of around 1024 Hz Since they do not blend, in addition to this the periodic background phone ring appears as almost discrete in the frequency domain it makes it easy for bots and APIs to eliminate the layer and separate the beats. This problem could be dealt with to an extent by using an effective band pass filter

The last picture, Fig. 6, we have used in our solution is bell beats with a layer of GAN-generated music patch and white noise. We see the frequency range is easily observable but the number of times the bell rings (periodicity) merge with the GAN music layer

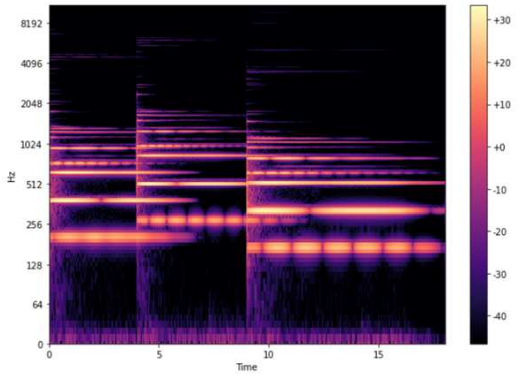

Fig. 3. Sample with bell beats and no background

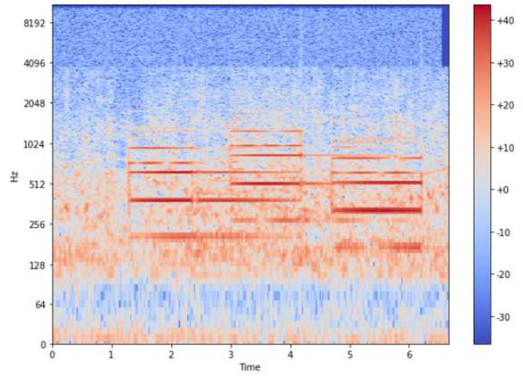

Fig. 4. Sample with bell beats and noise

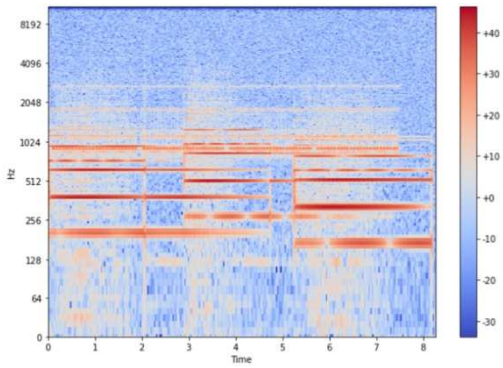

Fig. 5. Sample with bell beats and phone tune (monotonic)

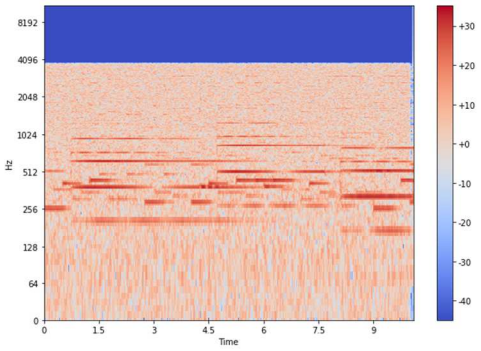

Fig. 6. Sample with bell beats, GAN generated music patch, and noise.

IV. Results

We generated some samples of alphabets and tested them over Google speech recognition API with different backgrounds.

TABLE II. Google API test on various samples

Sr. No	Input Alphabets	Prediction
1	'iox'	{'alternative': [{'confidence': 0.98762906, 'transcript': 'i o x'}, {'transcript': 'o x'}, {'transcript': 'i o'}, {'transcript': 'i x'}], 'final': True}
2	'iox' with babble	{'alternative': [{'confidence': 0.8343113, 'transcript': 'i o x'}, {'transcript': 'i o s'}, {'transcript': 'o s'}], 'final': True}
3	'iox' with phone tune	{'alternative': [{'confidence': 0.72167331, 'transcript': 'i o x'}, {'transcript': 'I owe app'}, {'transcript': 'i o o x'}], 'final': True}
4	'iox' with music	{'alternative': [{'confidence': 0.59365875, 'transcript': 'oh yes'}, {'transcript': 'oh EX'}, {'transcript': 'oh X'}, {'transcript': 'OS'}, {'transcript': 'oh'}], 'final': True}

It was observed that the API can detect the alphabets when merged with random noise like babble/crowd as most of these APIs have noise cancellation features. In samples (like sample 3) where the alphabets are merged over some noise of constant periodic frequency (like phone tune), it can detect the alphabets. In sample 4 the alphabets are merged with a music patch The API finds it a little difficult to detect them. However, some stronger systems can isolate speech from music and thus making it possible to detect alphabets with music patches as background.

To overcome these shortcomings, we propose a CAPTCHA technique in which the user aims to count the

number of beats of a unique sound (percussion instrument). Currently, the common APIs are not able to perform the task of counting beats. Adding to it, merging the beats with a piece of music, makes it difficult for the APIs to distinguish beats from music and count them. On the contrary, it is easy for a human to distinguish them. Thus satisfying the purpose of CAPTCHA.

We have made some samples of beats and merged them with different backgrounds as explained in the methodology. It can be observed that adding just noise or periodic music won't suffice the purpose. In figure 3.4, it can be seen that the bell sounds merge with the GAN-generated music layer. This makes it difficult for spambots or APIs to separate the two layers. Moreover, it is even difficult for them to distinguish between two music layers. Since the GAN-generated layer would be a random piece of music, it wouldn't be possible for APIs to identify the music patch and differentiate the layer which contains the beats of the percussion instrument. Additionally adding a layer of white noise would add to the difficulty of identification.

The samples have been tested on a batch of around 50 people. The people were given different GAN-generated samples and asked to count the number of beats. The accuracy with which people were able to count the number of beats in various samples was nearly 98.73%.

V. Conclusion

Thus, through this form of audio CAPTCHA, we are making sure that the blind/visually impaired users can solve the CAPTCHA test effortlessly while at the same time keeping bots/APIs at bay thereby satisfying the requirement of what a CAPTCHA test should fulfill. Since there are no numbers or letters to identify, speech-to-text APIs would not be able to extract anything. Various unique sound samples can be used to create different challenges and provide novelty. We believe that our work is a first of its kind which uses music in the background to create a CAPTCHA and we have demonstrated the effectiveness of this approach to some extent via our experiments. This is just a starting point and there are several interesting directions for future explorations.

VI. Future Work

The beats in our model are created using bell sound. It is a good idea to explore the possibility of replacing the bell sounds with tapping sounds of some other percussion instruments (most preferably tabla or drum). As the spectrum in case of Tabla (a percussion instrument typically used as an accompanying instrument in Hindustani classical music) or drum doesn't have discrete harmonic structure and it may blend well with the spectrum of GAN music (bots cannot conveniently exploit and separate it). We are performing experiments on this line. Also, the possibility of using a burst of more complex sounds in place of beats can be tried.

We also aim to build an Adaptive counter Neural Network and check for the limitations of our current approach to further enhance its strength against the Adaptive Neural net-based attacks. Some experiments can be performed to observe the effect of the loudness of sounds used in various layers and plot their spectrograms to deduce the conclusions. We are planning to build an easy interactive App using which the extent of the experiment for evaluating our model can be increased.

References

[1] WHO (2018). Blindness and vision impairment. Retrieved from http://www.who.int/news-room/factsheets/detail/blindness-and-visual-impairment. (11 October 2018)

[2] https://support.google.com/reCAPTCHA/answer/6175971?hl=en

[3] Alnfiai, M. (2020). A Novel Design of Audio CAPTCHA for Visually Impaired Users. International Journal of Communication Networks and Information Security (IJCNIS), 12(2)

[4] Shirali-Shahreza, M., & Shirali-Shahreza, S. (2007). CAPTCHA for Blind People. 2007 IEEE International Symposium on Signal Processing and Information Technology. doi:10.1109/isspit.2007.4458048

[5] J. Lazar, J. Feng, T. Brooks, G. Melamed, B. Wentz, J. Holman, A. Olalere, and N. Ekedebe, "The SoundsRight CAPTCHA: An Improved Approach to Audio Human Interaction Proofs for Blind Users," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 2012, pp. 2267–2276

[6] Shirali-Shahreza, S., Abolhassani, H., Sameti, H., & Hassan, M. (2009). Spoken CAPTCHA: A CAPTCHA system for blind users. 2009 ISECS International Colloquium on Computing, Communication, Control, and Management.

[7] Tam, J., Simsa, J., Hyde, S. and Ahn, L.V., 2008. Breaking audio CAPTCHAs. In Advances in Neural Information Processing Systems (pp. 1625-1632).

[8] Elie Bursztein, Steven Bethard, Stanford University Decaptcha: Breaking 75% of eBay Audio CAPTCHAs

[9] Li, S.; Jang, S.; Sung, Y. Automatic Melody Composition Using Enhanced GAN. Mathematics 2019, 7, 883. https://doi.org/10.3390/math7100883

[10] J.P. Bigham, and A.C. Cavender, Evaluating existing audio CAPTCHAs and an interface optimized for non-visual use. In Proceedings of ACM CHI 2009 Conference on Human Factors in Computing Systems, 2009, pp. 1829-1838

[11] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In Eurocrypt, 2003.

[12] pp. 403-417. [17] K.K. Paliwal and L. Alsteris, Usefulness of Phase Spectrum in Human Speech Perception, Eurospeech 2003 - Geneva, 2003, pp.2117-2120.

[13] C-RNN-GAN: Continuous recurrent neural networks with adversarial training, Olof Mogren Chalmers University of Technology, Sweden

[14] Generative Adversarial Networks, Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, arXiv:1406.2661 [stat.ML]

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.

[16] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on, pages 747–756. IEEE, 2002.

[17] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. arXiv preprint arXiv:1609.05473, 2016