Unleashing The Power of GANs in Audio Generation: MVGAN Vocoder

Quah Johnnie

School of Computing

Singapore Polytechnic

Singapore

johnnie.20@ichat.sp.edu.sg

***Abstract*—This technical paper presents MVGAN, a GAN-based vocoder to convert low-resolution mel-spectrogram into realistic sounding audio. The proposed GAN-based vocoder is inspired by the architectures of MelGAN and VocGAN. It combines the characteristics of both networks to create a powerful GAN-based vocoder. The experiments in this paper have shown that MVGAN can produce realistic and natural-sounding audio samples, even with limited training data. Additionally, the proposed model has been evaluated and demonstrated to outperform traditional vocoder methods in terms of quality and efficiency. Samples of the synthesized mel-spectrograms vocoded by MVGAN can be viewed in the following link: https://youtu.be/g4IztLpcH40.**

Keywords—Voice generation, Generative Adversarial Networks, Audio synthesis, Text-to-speech, Deep learning

I. Introduction

Voice synthesis and manipulation is an active area of research, with applications ranging from speech-based interfaces to music production [1]. One key component of voice synthesis is the vocoder, which is used to generate synthetic audio from a set of parameters such as pitch and formants, e.g. mel-spectrogram, which is a low resolution representation of pitch and formants as seen below in Figure 1.

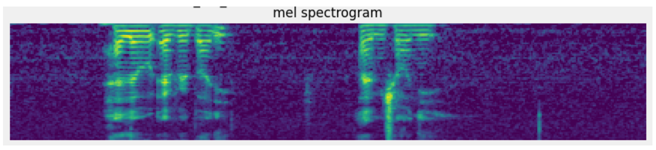

Fig. 1. Mel-spectrogram of me saying "Hello everybody, I'm Johnnie" -About 50 kilobytes of data

Traditional vocoder methods rely on parametric models, which can be computationally expensive and require large amounts of training data [2]. In essent, the job of a vocoder is to convert low resolution mel-spectrogram into realistic waveforms, or audio. This process is called vocoding or mel-spectrogram inversion.

In this paper, the focus is on the task of mel-spectrogram inversion, also known as vocoding. This is the second stage in the audio modeling process. The first stage involves synthesizing text and voice samples into a mel-spectrogram [3]. Mel-spectrograms typically consist of a few kilobytes of data; e.g. Figure 1 consists of approximately 50 kilobytes. However, when using a mel-spectrogram as input for the generator of a MVGAN, the final output is a realistic sounding waveform, which could be a few megabytes, depending on the length of the mel-spectrogram.

Producing a text-to-speech engine from scratch requires 3 main components, the encoder, synthesizer and vocoder. The encoder can be used to encode speakers' voices in the form of a waveform to create a mel-spectrogram [4]. More importantly, the synthesizer along with the encoder can be used to encode a generated grapheme or phoneme sequence, which are transformed text sequences in this context, to form a mel-spectrogram [5]. The mel-spectrogram is then fed to the generator of MVGAN for vocoding, which will produce a realistic sounding waveform. The whole process can be visualized below in Figure 2 or Figure 3.

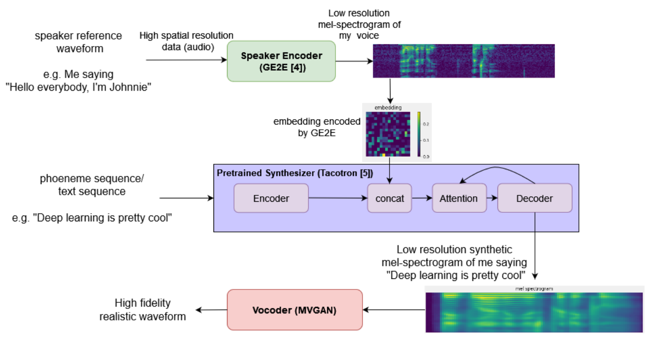
speaker reference waveform
e.g. Me saying "Hello everybody, I'm Johnnie"
High spatial resolution data (audio)
Speaker Encoder (GE2E [4])
Low resolution mel-spectrogram of my voice
embedding encoded by GE2E
embedding
phoeneme sequence/ text sequence
e.g. "Deep learning is pretty cool"
Pretrained Synthesizer (Tacotron [5])
Encoder
concat
Attention
Decoder
Low resolution synthetic mel-spectrogram of me saying "Deep learning is pretty cool"
mel spectrogram
Vocoder (MVGAN)
High fidelity realistic waveform

Fig. 2. Whole process to produce realistic human text-to-speech waveform with a sample voice and text.

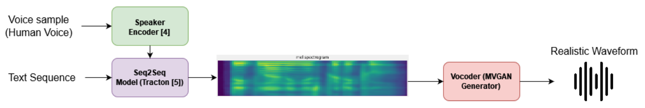
Voice sample (Human Voice)
Speaker Encoder [4]
Text Sequence
Seq2Seq Model (Tracton [5])
mel spectrogram
Vocoder (MVGAN Generator)
Realistic Waveform

Fig. 3. Simplified version of the whole realistic human text-to-speech process.

II. Related Works

There have been many popular traditional, GAN-based and non-GAN based neural vocoders. In this section we explore different forms of vocoder used in the past to convert low-resolution mel-spectrogram into high fidelity waveform or audio.

A. Griffin-Lim - Pure signal processing approach

A variety of techniques in signal processing have been examined to identify low-resolution audio representations that can be both accurately modeled and efficiently transformed back into temporal audio. One such technique is the Griffin-Lim algorithm [6], which follows a pure signal processing approach.

The Griffin-Lim algorithm approximates the magnitude spectrogram of a signal, given the phase spectrum of that signal. It utilizes an iterative process, beginning with an initial estimate of the magnitude spectrogram. The algorithm repeatedly applies the inverse short-time Fourier transform (STFT) and the STFT to the signal, adjusting the magnitude spectrogram at each iteration to minimize the difference between the estimated and actual phase spectrums [6]. The algorithm was first published by its inventors, Griffin and Lim. However, it is worth noting that vocoded audio produced using the Griffin-Lim algorithm has strong robotic artifacts [1].

B. WaveNet & WaveRNN - Autoregressive RNN-based models

WaveNet is a deep neural network architecture that is used for generating audio signals. It was developed by the Google DeepMind team [3][4]. The architecture is based on a stack of dilated causal convolutional layers, which allow the network to have a large receptive field while maintaining a small number of parameters. This means that WaveNet can learn to model long-term dependencies in a given mel-spectrogram.

One of the key innovations of WaveNet is that it models the probability distribution of each sample in the mel-spectrogram, rather than just generating a fixed sequence of samples. This allows the network to produce high-quality, realistic-sounding audio, and also enables the use of techniques such as sampling and quantization to control the trade-off between audio quality and computational cost.

WaveRNN is a more efficient auto-regressive model that utilizes a single-layer recurrent neural network architecture [7]. It incorporates various techniques like sparsification and sub-scale generation to enhance synthesis speed.

Despite the enhanced efficiency of autoregressive RNN-based models with advanced techniques, they remain inherently slow and inefficient due to the sequential nature of audio sample generation. This makes them unsuitable for real-time applications, as generating a few minutes of audio on a CPU can take several hours [8].

C. WaveGlow - Non autoregressive models

WaveGlow is a non-autoregressive model that is much faster and computationally less expensive than autoregressive models [9]. This makes it more suitable for real-time applications, such as speech synthesis and music generation; it generates all samples of the audio signal at once, unlike autoregressive models, which generate audio samples one at a time. WaveGlow has been shown to produce high-quality audio synthesis and it can be used for a variety of audio-related tasks such as speech synthesis, music generation, and audio denoising.

WaveGlow is a flow-based model that is trained to learn the mapping between a simple noise distribution and the target audio signal. A flow based model is a type of generative model that uses a flow of data or information to generate a desired output. This is typically done using a series of transformations, which are known as "flows". These flows are used to transform a given input into a desired output. In the case of WaveGlow, the input is an audio signal and the desired output is a synthesized audio signal. WaveGlow uses a series of flows to transform the audio signal into a synthesized audio signal that is similar to a mel-spectrogram.

However, what makes non-autoregressive models like WaveGlow impractical to implement is the amount of computation power needed to train these models. The researchers trained WaveGlow on 8 V100s for over a week and took at least 1 V100 which has 32GB VRAM to synthesize speech [9]. WaveGlow contains 88 million parameters.

D. MelGAN and VocGAN - GAN-based Vocoders

GAN, or Generative Adversarial Network, is composed of two main components: a generator and a discriminator [10]. The generator is trained to generate realistic audio waveforms given a mel-spectrogram, while the discriminator is trained to distinguish between real and fake audio waveforms [11]. The two networks are trained together in an adversarial manner, where the generator tries to fool the discriminator into thinking that its output is real, while the discriminator tries to correctly identify whether the audio waveform is real or fake.

MelGAN is trained using a slightly modified version of hinge loss, equations (1) and (2) show the hinge loss used to train both MelGAN's generator and hinge loss. Note that it is a simplified version of the hinge loss function used in MelGAN.

Hinge loss for MelGAN discriminator:

$$L_D = -\frac{1}{m}\sum_{i=1}^{m}[\min(0, 1 - D_k(x_i)) + \min(0, 1 + D_k(G(s,z)))] \text{ for all } k = 1,2,3 \quad (1)$$

Hinge loss for MelGAN generator:

$$L_G = -\frac{1}{m}\sum_{i=1}^{m}[D_k(G(s,z))] \text{ for all } k = 1,2,3 \quad (2)$$

Where `m` is the number of samples in the dataset. `G(s, z)` is the generated waveform data, where `s` is the mel-spectrogram input data and `z` is the random latent input.

VocGAN is trained on different loss functions [12], mainly hierarchically-nested adversarial loss, which is a combination of the Joint Conditional and Unconditional loss and is short for JCU loss. The equation for JCU loss can be seen in (3) and (4). JCU loss is a loss function used in the VocGAN model to improve speech quality. It combines the idea of conditional and unconditional adversarial losses to guide the generator to map the acoustic feature of the input mel spectrogram to the waveform more accurately. This helps to reduce the discrepancy between the acoustic characteristics of the input mel spectrogram and the output waveform. VocGAN uses 5 sub-discriminators with different weightage to JCU loss, the final JCU loss is the concat of all loss accumulated by the discriminators.

JCU loss used for VocGAN discriminator:

$$\mathrm{L}_{JCU}^{D}(G,D) = \tfrac{1}{2}\textstyle\sum_{k=1}^{K}[E_s[D_k(\hat{x}k)^2 + Dk(\hat{x}k,s)^2] + Es,x_k[(D_k(x_k)-1)^2 + (D_k(x_k,s)-1)^2]] \quad (3)$$

JCU loss used for VocGAN generator:

$$\mathrm{L}^{G}_{JCU}(G,D) = \frac{1}{2}\sum_{k=1}^{K} E_s[(D_k(\hat{x}k) - 1)^2 + (Dk(\hat{x}_k, s) - 1)^2] \quad (4)$$

Where `k` is the number of discriminators in the hierarchical architecture, `G` is the generator, `s` is the mel-spectrogram data, `xk` is the real audio data

III. EXPERIMENT

A. *Datasets & Preparation*

In this experiment, we used the LJSpeech [13] and openSLR train-clean-100 [14] corpus of read English speech datasets. The LJSpeech dataset is composed of audio recordings of a single female speaker reading a selection of texts, with the goal of training text-to-speech models. It includes around 13,100 audio files, each paired with a corresponding text transcript. The openSLR train-clean-100 dataset is similar, but it features a variety of speakers, both male and female, reading from a selection of audiobook texts. Together, the two datasets consist of approximately 125 hours of audio, with 100 hours from openSLR train-clean-100 and 25 hours from LJSpeech. Both dataset adds up to about 70,000 samples.

To create the mel-spectrograms for all of the audio, the raw data from the datasets were passed through an encoder [4]. The output mel-spectrogram along with a random latent input will serve as the input for our generator, with the output being a fake audio for our discriminator to distinguish. The real audio will also be used to train our discriminator. The ultimate goal of this experiment is to develop a generator that can generate realistic-sounding audio from a given mel-spectrogram, as seen in Figure 4.

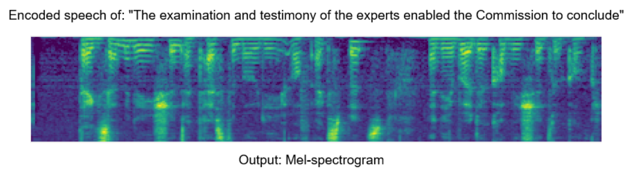

Fig. 4. Example of a sample mel-spectrogram output, which will serve as the input for generator

B. *Metrics For Evaluation*

Mel Cepstral Distance (MCD) and Perceptual Evaluation of Speech Quality (PESQ) will be used as the two main metrics to measure the realism of the generated audio.

Mel Cepstral Distance is a metric used to measure the similarity between two speech signals in the frequency domain. It is calculated by converting the speech signals to the Mel-Cepstral domain, and then computing the Euclidean distance between the two resulting Mel-Cepstral coefficients. This distance represents the dissimilarity between the two signals, with a lower MCD indicating a higher similarity [15].

The Perceptual Evaluation of Speech Quality (PESQ) is a standard metric that measures the perceived quality of speech signals. PESQ compares the original and the degraded speech signal and it is calculated by combining the results of a number of different measurements, including the MCD, the Root Mean Square Error (RMSE) of the frequency response, and the RMSE of the excitation. PESQ score ranges from -0.5 to 4.5, where a high score indicates a high speech quality and low score indicates poor speech quality [16].

C. *MVGAN - The Architecture*

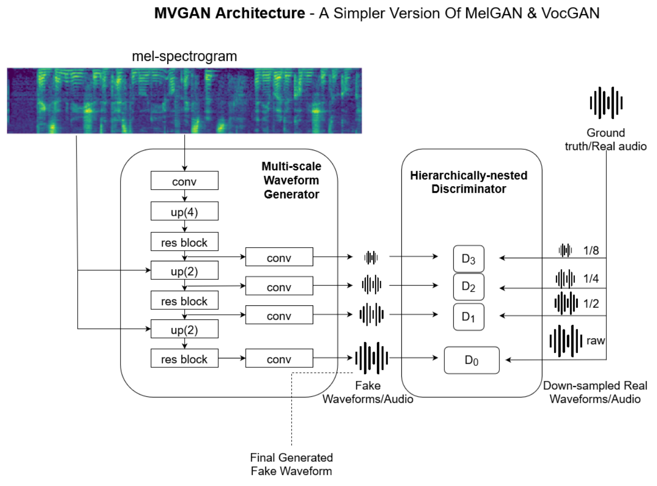
MVGAN Architecture - A Simpler Version Of MelGAN & VocGAN
mel-spectrogram
Multi-scale Waveform Generator
conv
up(4)
res block
up(2)
res block
up(2)
res block
conv
conv
conv
conv
Hierarchically-nested Discriminator
D3
D2
D1
D0
Ground truth/Real audio
1/8
1/4
1/2
raw
Fake Waveforms/Audio
Down-sampled Real Waveforms/Audio
Final Generated Fake Waveform

Fig. 5. Simplified version of MVGAN

The architecture in Figure 5 is designed by me and easily implemented with the code toolkit provided by coqui-ai [17]. The final generator has only 3.1 million parameters, while the discriminator has 2.7 million parameters. 'up(n)' depicts the upsampling of data by n times.The generated waveform is discriminated a total of 4 times where the final discriminator (D_0) has the most weightage to the JCU loss.

D. *Training*

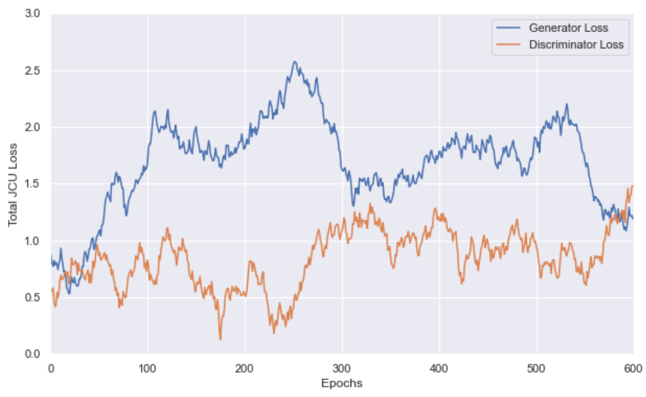

Fig. 6. Loss graph of MVGAN during training

The whole model will be trained for 600 epochs with the Adam optimizer [18] with a β1 = 0.5, β2 = 0.9 and learning rate of 0.0001, these values were chosen due to their notable stability during training for GANs [10]. The batch-size used was 64 and it was trained using 1 RTX3080 for a total of about 13 hours excluding the time taken to calculate MCD and PESQ. The losses were recorded every epoch. However, the two metrics MCD and PESQ were only recorded every 150 epochs and compared against only the LJSpeech dataset, this decision was chosen due to the computationally intensive nature of calculating both metrics of MCD and PESQ between fake and real waveform of LJSpeech dataset.

Looking at Figure 6, for the most part of my training, the discriminator has a lower loss than the generator. However, towards the end the generator has about the same loss as the discriminator. This could indicate that the generator has effectively learned the distribution of the real data and is able to generate new samples that are more indistinguishable from the real data. This is desirable, as the goal of a GAN is to have the generator produce outputs that are indistinguishable from the real data.

E. Results & Comparison

TABLE I. Results Metrics Compared Against LJSpeech Dataset

Models | Epoch Trained	**MCD** *(Lower Better)*	**PESQ** *(Higher Better)*
MVGAN | 150	11.48	0.673
MVGAN | 300	5.16	1.468
MVGAN | 450	4.83	2.273
MVGAN | 600 (Mine)	4.21	2.896
MelGAN [11] | 500	4.61	2.74
VocGAN [12] | 3000	**3.20**	**3.44**
Ground Truth	0	4.5

Note that the epochs trained in Table 1 are not directly comparable with one another as they are trained with differing training sets. My training set used to train MVGAN has 70,000 samples, MelGAN has about 48,000 samples and VocGAN only has 26,000 samples.

IV. Discussion

A. Results Interpretation & Comparison

My GAN model, MVGAN, outperforms MelGAN in both MCD and PESQ metrics, demonstrating its ability to convert mel-spectrograms more realistically. It is noteworthy that MVGAN was trained longer and with a larger dataset compared to MelGAN. VocGAN, with its training of 3000 epochs using 26,000 samples, exhibits state-of-the-art performance and outperforms both MVGAN and MelGAN.

Most vocoders [1][3][6][7][8][11][12] use the metric Mean Opinion Score or MOS [19], which is a subjective evaluation metric used to measure the quality of audio, video, and speech signals. It is a rating scale in which listeners are asked to rate the quality of audio samples on a scale of 1 to 5, where 1 being the lowest quality and 5 being the highest quality. The MOS score is then calculated as the average of the individual ratings.

B. Limitations

I was unable to use the subjective but effective metric MOS. During my research on neural vocoders and text-to-speech generators, many of the research papers, with the exception of VocGAN [12], do not actually use objective metrics such as MCD and PESQ to measure the quality of their generated audio samples. Instead they use Mean Opinion Score or MOS, which is a very effective way to determine many audio generators or vocoders [19]. Although I was able to prepare generated audio samples from MVGAN, I was unable to obtain enough opinions and surveys to generate a MOS to compare different generative audio models.

Using a larger architecture and training my model, MVGAN, for a longer period of time would likely achieve better results, MVGAN is based on MelGAN and VocGAN, I was able to prepare and train MVGAN with a larger dataset. However, due to time limitations I have to cut down on the size of MVGAN as well as its training epoch. Nonetheless, it achieves better results than MelGAN.

C. Rooms For Improvement

Most exploration was put into creating a working GAN-based vocoder to generate audio. However, not as much exploration was placed into improving MVGAN, there were many hyperparameters to tweak and improve, especially the architecture. An area to put more effort into would be the use of attention-based modules in the architecture as there has already been research thoroughly and done on GAN-based vocoders [20], it also has proven quality of audio generated.

V. Conclusion

In this paper, we introduced a novel GAN-based vocoder for audio generation. Our results showed that our proposed model achieved a higher level of audio quality and diversity compared to existing models. The implementation of the self-attention mechanism in the generator was a key factor in improving the generated audio.

In conclusion, the proposed GAN-based vocoder for audio generation is a promising approach for generating high-quality audio. The results demonstrate the effectiveness of the proposed architecture, although the metric MOS was not used. This GAN-based vocoder has the potential to be applied in a variety of real-world scenarios, such as speech synthesis, music generation, and audio augmentation.

Acknowledgement

I would like to express our gratitude to the developers of the "TTS" repository by coqui-ai [17] and the "Real-Time-Voice-Cloning" repository by CorentinJ [21]. The "TTS" repository provided the code toolkit that enabled us to build our GAN-based vocoder, while the "Real-Time-Voice-Cloning" repository provided the pretrained synthesizer, Tacotron, [5] and encoder [4] that allowed me to generate embedded high quality mel-spectrograms to test my GAN-based vocoder. These resources were instrumental in making this technical paper possible and I am grateful for the efforts of the developers in making them publicly available.

Ending Note

This technical paper is free of copyright, open-source and reusable. All diagrams are original by the author, Quah Johnnie, and made with draw.io [22]. However, this paper is NOT fact checked nor submitted by any institution or any legitimate academic personnel, this is purely for an assignment. Do NOT quote me on any figures or analysis done in this technical paper as it may appear unintentionally false. To listen to the generated audio from MVGAN, visit the following link: https://youtu.be/g4IztLpcH40. Thank you!

References

[1] A. Balyan, S. S. Agrawa, and Amita, *Speech Synthesis: A Review*, vol. 2, no. 6, pp. 57–59, 2013.

[2] H. U. Mullah, *A Comparative Study of Different Text-to- Speech Synthesis Techniques*, pp. 1–4, Jul. 2015.

[3] Oord A ̈aron van den, S. Dieleman, K. Simonyan, O. Vinyals , A. Graves, N. Kalchbrenner , A. Senior, and K. Kavukcuoglu, *WaveNet: A Generative Model For Raw Audio*, vol. 2, pp. 1–3, Sep. 2016.

[4] L. Wan , Q. Wang, A. Papir, and I. L. Moreno, *Generalized End-To-End Loss For Speaker Verification*, vol. 5, pp. 1–4, Nov. 2020.

[5] Y. Wang, R. J. S. Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, *Tacotron: Towards End-To-End Speech Synthesis*, vol. 2, pp. 4–5, Apr. 2017.

[6] D. W. Griffin and J. Lim, *Signal estimation from modified short-time Fourier transform*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[7] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, Oord A ̈aron van den, S. Dieleman, and K. Kavukcuoglu, *Efficient Neural Audio Synthesis*, vol. 2, pp. 1–4, Jun. 2018.

[8] W. Ping, K. Peng, A. Gibiansky, Arık Sercan ̈O., A. Kannan, and S. Narang, *Deep Voice 3: Scaling Text-To-Speech With Convolutional Sequence Learning*, vol. 3, pp. 1–3, Feb. 2018.

[9] R. Prenger, R. Valle, and B. Catanzaro, *WaveGlow: A Flow-Based Generative Network For Speech Synthesis*, vol. 1, pp. 1–4, Oct. 2018.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Nets*, vol. 1, pp. 1–4, Jun. 2014.

[11] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, *MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis*, vol. 3, pp. 1–6, Dec. 2019.

[12] J. Yang, J. Lee, Y. Kim, H. Cho, and I. Kim, *VocGAN: A High-Fidelity Real-time Vocoder with a Hierarchically-nested Adversarial Network*, vol. 1, pp. 1–4, Jul. 2020.

[13] K. Ito and L. Johnson, “The LJ speech dataset,” *The LJ Speech Dataset*, 2017. [Online]. Available: https://keithito.com/LJ-Speech-Dataset/. [Accessed: 20-Jan-2023].

[14] V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, QLD, Australia, 2015, pp. 5206-5210, doi: 10.1109/ICASSP.2015.7178964.

[15] R. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, Victoria, BC, Canada, 1993, pp. 125-128 vol.1, doi: 10.1109/PACRIM.1993.407206.

[16] A. W. Rix, J. G. Beerends, M. P. Hollier and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, Salt Lake City, UT, USA, 2001, pp. 749-752 vol.2, doi: 10.1109/ICASSP.2001.941023.

[17] Eren, G., & The Coqui TTS Team. (2021). Coqui TTS (Version 1.4) [Computer software]. https://doi.org/10.5281/zenodo.6334862

[18] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, vol. 9, pp. 1–5, Dec. 2014.

[19] Streijl, R.C., Winkler, S. & Hands, D.S. *Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives.* Multimedia Systems, vol 22, pp. 213–227. https://doi.org/10.1007/s00530-014-0446-1, Mar. 2016

[20] C. Miao, T. Chen, inchuan Chen, J. Ma, S. Wang, and J. Xiao, *A Compact Transformer-Based GAN Vocoder*, vol. 1, pp. 1636–1639, Sep. 2022.

[21] C. Jemine, *Real-Time Voice Cloning*, 26-Jun-2019. [Online]. Available: https://github.com/CorentinJ/Real-Time-Voice-Cloning. [Accessed: 27-Jan-2023].

[22] Draw.io, “Draw.io Diagramming WebApp,” Draw.io. [Online]. Available: https://app.diagrams.net/. [Accessed: 27-Jan-2023].