Stutter-TTS: Synthetic Generation of Diverse Stuttered Voice Profiles

Anonymous Author(s)
Affiliation
Address
email

Abstract

Stuttering is a speech disorder where the natural flow of speech is interrupted by blocks, repetitions or prolongations of syllables, words and phrases. The majority of existing automatic speech recognition (ASR) interfaces perform poorly on utterances with stutter, mainly due to lack of matched training data. Synthesis of stuttering voice profiles thus presents an opportunity to improve ASR for these speakers with stutter. We describe Stutter-TTS, an end-to-end neural text-to-speech model capable of synthesizing diverse types of stuttering utterances. We develop a simple, yet effective prosody-control strategy whereby additional tokens are introduced into source text during training to represent unique stuttering characteristics. By choosing the position of the stutter tokens, Stutter-TTS allows word-level control of where stuttering occurs in the synthesized utterance.

1 Introduction

According to the National Institute on Deafness and Other Communication Disorders, there are nearly three million Americans suffering from lifelong stuttering. Advances in deep learning facilitate the development of ASR systems and encourage the integration of voice assistant in various commercial electronics (Kepuska and Bohouta [1]). However, people who stutter by and large have not benefited from this convenience, as existing ASR systems have difficulties understanding atypical speech, resulting in poor performance when it comes to stuttering (Barrett et al. [2]).

Driven by deep neural networks, recent research efforts have been dedicated to improved detection and recognition of disfluent speech (Bayerl et al. [3], Jouaiti and Dautenhahn [4]). Despite advances in modeling technology, one of the persisting bottlenecks is the lack of data representative of diverse stuttering patterns. The performance of stutter detection or recognition systems greatly depends on sufficient stuttered speech for model training (Barrett et al. [2]). For example, the most recently introduced SEP-28 dataset contains utterances with stutter comprising less than 24 hours (Lea et al. [5]).

One possible solution to this scarcity of matched speech data would be synthetic speech generated by text-to-speech (TTS) systems, as has already been used for other ASR training scenarios (Zheng et al. [6]).As a necessary preliminary step towards this goal, we focus here on the design of a TTS model capable of generating realistic and natural speech with diverse forms of stutter.

TTS technology has been widely utilized to produce artificial voices that closely emulate natural human conversation (Bilinski et al. [7]). In particular, end-to-end TTS synthesis has attracted wide attention due to the simplification in training and improved naturalness of synthetic utterances. Recent work has demonstrated the creation of multiple voices for context-aware conversational speech synthesis (Stanton et al. [8], Cong et al. [9]). Soleymanpour et al. [10] reported on synthesis of dysarthric speech based on a multi-speaker TTS framework. To the best of our knowledge, no literature has investigated how to leverage TTS for synthesizing different types of stuttering

Submitted to NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research. Do not distribute.

voices. For people who stutter, the natural flow of speech is interrupted by various irregular acoustic patterns, such as sound repetition, syllable prolongation, and long pausing. Consequently, it remains challenging to extend current TTS approaches to the production of realistic stuttering with high naturalness and fine-grained prosody control.

To address these limitations, we introduce Stutter-TTS, a novel TTS approach that achieves both naturalness and natural prosody in stuttered speech synthesis. We propose a novel prosody-control strategy for supervised learning by incorporating special tokens into the source text to represent different prosodic-phonetic characteristic of stutter, including phoneme repetition, dysrhythmic phonation, and blocks. By manipulating the input source text, Stutter-TTS can generate either fluent speech (without stutter) or specific types of stutter. We systematically produce 100 hours of diverse types of utterances containing stutter and quantify the generation performance by randomly sampling 400 utterances for evaluation.

2 Methods

A multi-speaker transformer-based TTS network has been used previously to model stuttering speech (Vaswani et al. [11]). The architecture is essentially similar as Chen et al. [12] and Li et al. [13], consisting of a transformer backbone, featuring a phonetic encoder, and an acoustic auto-regressive decoder. A scaled dot product (Kamath et al. [14]) attention mechanism is used to align the acoustic and phonetic features.

2.1 Stutter-TTS Architecture

Several modules are added on top of the base transformer. To condition the decoder on speaker identity information, a global audio reference encoder is included as a lightweight replacement of a speaker embedding model. This module consists of a Gated Recurrent Unit (GRU) (Chung et al. [15]) network that receives a set of randomly drawn frames from a reference Mel-spectrogram, and aggregates them into a time-independent representation. The reference is set to be the target Mel-spectrogram at training time, and the random sampling of frames is a mechanism to prevent the content of the target to be leaked to the decoder by destroying the time-dependent information while keeping an unbiased estimation of the frequency bins energy distributions. At inference time, a reference Mel-spectrogram of the desired speaker is used as input to the global audio reference encoder, as a prototype of the voice to use to synthesize the input sentence.

A stuttering disorder is often characterized by unintentional repetitions, prolongations, or interruption of sounds. It is very difficult to predict which phonemes in an utterance will be affected by stutter (Dash et al. [16]). From the speech generation perspective, this leads to a situation where the text-to-speech mapping is more ambiguous than for regular speakers. Following this line of reasoning, a probabilistic embedding is added as input to the phonetic encoder. Instead of modeling a constant embedding for each phoneme, the parameters of a diagonal Gaussian are used. This feature allows the model to learn the pronunciation uncertainty at phoneme level. Additionally, a learnable parameter α is used to weight the sum of the positional encoding, together with a layer normalization (Ba et al. [17]), which as described in Chen et al. [12], assures that both phonetic and positional information are preserved.

Auto-regressive decoders, especially when dealing with data that features local correlations as found in speech, often stall into a failure mode known as exposure bias (Arora et al. [18]): the decoder, instead of predicting the next step, copies its last input step. To prevent exposure bias, a prenet with a strong regularization is included before the decoder. This module is vital for the correct generalization of the model. It consists of a strong dropout (60%) that is kept active at inference time (Gal and Ghahramani [19]) followed by a strong bottleneck projection Chen et al. [12]. This regularization reduces the amount of information that is given to the decoder at each step, preventing it to stall into the exposure bias failure mode. Finally, after the decoder module, a postnet and a stop signal are included, similar to the Tacotron 2 architecture (Wang et al. [20]).

To train the model, the L1-loss between the target and the predicted Mel spectrogram is minimized using stochastic gradient descent, similar to the original loss function of Tacotron 2 (Wang et al. [20]). An additional L1-loss is included to enforce that the decoder module produces an output signal in the same domain as its input (autoregression requirement). Equation 1 shows the full loss function,

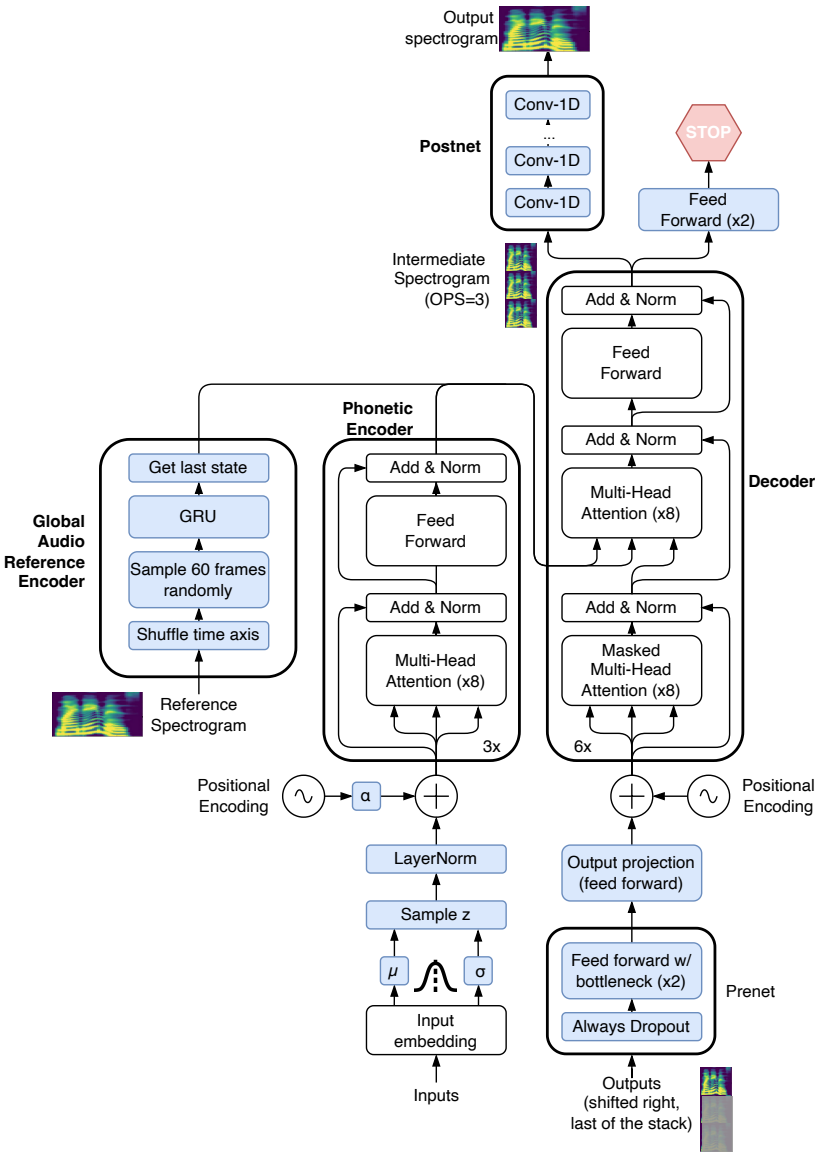

Figure 1: Diagram of Stutter-TTS architecture.

where $\hat{\mathbf{m}}_{\text{final}}$ is the mel-spectrogram after the postnet, $\hat{\mathbf{m}}_{\text{intermediate}}$ is the mel-spectrogram before the postnet and $\mathbf{m}$ is the target mel-spectrogram. The TTS model is train using *teacher-forcing* method (Williams and Zipser [21], Goodfellow et al. [22]). At inference time, the free-running mode is used, generating the samples one step at a time in an auto-regressive fashion. The auto-regressive loop contains the decoder and the prenet modules, but not the postnet module Wang et al. [20].

$$J(\mathbf{m}, \hat{\mathbf{m}}_{\text{intermediate}}, \hat{\mathbf{m}}_{\text{final}}) = \cdot||\hat{\mathbf{m}}_{\text{intermediate}} - \mathbf{m}||_1 + ||\hat{\mathbf{m}}_{\text{final}} - \mathbf{m}||_1 \tag{1}$$

2.2 Stutter Token

To replicate recurring prosodic-phonetic phenomena associated with stutter, we use a list of special tokens to denote different stuttering patterns and their location. Specifically, we insert stutter tokens immediately in front of the word where stuttering occurs in the corresponding audio. In this work, we mainly focus on three common stutter types as described in Table 1. During grapheme-to-phoneme (G2P) conversion, stutter tokens are treated as unique tokens that are directly concatenated to the

phoneme set. The TTS model will hence learn embedding vectors associated with each of the stutter tokens.

Table 1: The mapping rule from different types of stutter to corresponding tokens inserted in the source sentence, along with their relative frequencies in the annotated training dataset

Stutter Type	Stutter Token	Percentage (%)
Phoneme repetition	s_repetition	40.11
Dysrhythmic phonation	s_phonation	21.40
Block	s_block	15.59

As illustrated in Figure 2, stutter labels are introduced into the input sentence to denote certain prosodic-phonetic structure. It is worthwhile pointing out that the proposed processing approach achieves word-level prosody control in terms of where stuttering happens in the synthetic utterances. This design allows fine-grained control of stuttering occurrences at synthesis time. In the inference stage, we can simply place the token for the desired stuttering pattern prior to the word where we want the model to render with stutter. Subsequently, the resulting synthetic audio will produce a stutter at the designated position in the source sentence.

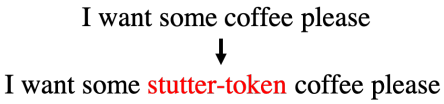

Figure 2: An illustration of how stuttered prosodies are included in the input transcript. Stutter token is precisely inserted prior to the stuttered word in the utterance. Stutter token can be customized as needed to represent different stutter types.

3 Experimental Results

3.1 Dataset Description and Model Training

The Stutter-TTS model is trained using a combination of two proprietary datasets, one containing close-talking microphone fluent speech (without stutter) and one with reference (golden) stuttering speech. The fluent speech dataset contains 10 professional speakers with 13,000 studio-recorded utterances per speaker (600 hours in total). The golden stuttering dataset contains 146 native English speakers who stutter with 125 utterances per speaker (40 hours in total). Utterances in both datasets are 6 to 12 seconds long.

We process all audio at 16 kHz and generate 80-dimensional Mel spectrograms. The length of a frame is 50 ms with an overlap of 12.5 ms. We employ the Universal Neural vocoder to synthesize audio samples using output spectrogram from Stutter-TTS (Lorenzo-Trueba et al. [23]).

3.2 Evaluation of Synthetic Stuttered Speech

To evaluate the synthesis of utterances with stutter, we compare the Mel spectrogram generated from Stutter-TTS with the associated recording, collected from speakers with stutter. We modify the original transcription by inserting the stutter tokens where the speaker stuttered, with the aim to reproduce the stutter pattern of the original recording. As shown in Figure 3, our model is able to mimic repetition patterns as highlighted in the red rectangle. More importantly, we observe that when eliminating the stutter token from the source text, the resulting synthetic utterance contains no stuttering, thus preserving the ability to produce fluent utterances with high naturalness.

In this experiment, we sample 20 reference recordings for each of 10 speakers, and paired with 10,000 sentences for each of the speaker. We randomly insert one stutter token into input sentences with equal probability on location over all words. We systematically synthesize 100 hours of speech containing three stutter types. To measure generation performance with stutter, we randomly sample

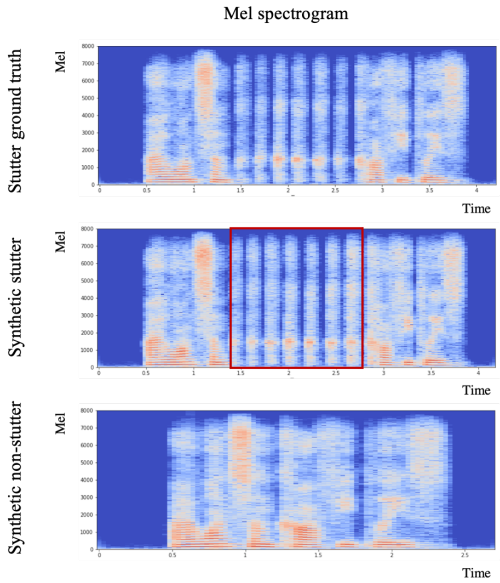

Figure 3: Comparison of Mel spectrograms of ground truth versus synthetic stuttered speech.

400 utterances containing phoneme repetition, dysrhythmic phonation, block and non-stutter. We evaluate the existence of specific stutter types in a subjective manner, by playing synthesized audio recordings to identify whether the desired stuttering characteristics occur.

Evaluation results are detailed in Table 2. It is vital to perform model training using a combination of both fluent and stuttering speech. Moreover, we experiment with a range of sampling weights to optimize the model's performance producing both fluent and stuttering utterances. It is beneficial to increase the proportion of stuttering samples as it leads to improved generation of diverse stuttering patterns. However, oversampling stuttering data hurts model performance on fluent speech (dropping from 0.733 to 0.575). We would attribute the variation in synthesis accuracy to the biased distribution of stutter types in the training dataset.

Table 2: F1 scores corresponding to diverse stutter types with variations on ratio of fluent speech versus stuttered speech.

Ratios(%)	Phoneme Repetition	Dysrhythmic Phonation	Block	Non-Stutter
95:5	0.692	0.503	0.720	0.647
90:10	**0.786**	**0.633**	0.837	**0.733**
85:15	0.773	0.615	**0.853**	0.575

4 Conclusion

We present a novel Stutter-TTS system that can produce voice profiles that can generate stutter in a highly controlled manner. We incorporate a list of special tokens to denote characteristics of stuttering patterns in the source text. For training Stutter-TTS, it is critical to fine-tune the sampling ratio between fluent and stuttering speech. Stutter-TTS achieves faithful synthesis of artificial utterances with stutter types including phoneme repetition, dysrhythmic phonation, and blocks. In addition, systematic speech synthesis demonstrates the ability to create new voices with specified stuttering structure. In future work, we will explore the potential of Stutter-TTS to improve the recognition of stuttered speech via generation of matched ASR training data.

References

[1] Veton Kepuska and Gamal Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, pages 99–103. IEEE, 2018.

[2] Liam Barrett, Junchao Hu, and Peter Howell. Systematic review of machine learning approaches for detecting developmental stuttering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

[3] Sebastian P Bayerl, Dominik Wagner, Elmar Nöth, and Korbinian Riedhammer. Detecting dysfluencies in stuttering therapy using wav2vec 2.0. *arXiv preprint arXiv:2204.03417*, 2022.

[4] Melanie Jouaiti and Kerstin Dautenhahn. Dysfluency classification in stuttered speech using deep learning for real-time applications. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6482–6486. IEEE, 2022.

[5] Colin Lea, Vikramjit Mitra, Aparna Joshi, Sachin Kajarekar, and Jeffrey P Bigham. Sep-28k: A dataset for stuttering event detection from podcasts with people who stutter. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6798–6802. IEEE, 2021.

[6] Xianrui Zheng, Yulan Liu, Deniz Gunceler, and Daniel Willett. Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5674–5678. IEEE, 2021.

[7] Piotr Bilinski, Thomas Merritt, Abdelhamid Ezzerg, Kamil Pokora, Sebastian Cygert, Kayoko Yanagisawa, Roberto Barra-Chicote, and Daniel Korzekwa. Creating new voices using normalizing flows.

[8] Daisy Stanton, Matt Shannon, Soroosh Mariooryad, RJ Skerry-Ryan, Eric Battenberg, Tom Bagby, and David Kao. Speaker generation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7897–7901. IEEE, 2022.

[9] Jian Cong, Shan Yang, Na Hu, Guangzhi Li, Lei Xie, and Dan Su. Controllable context-aware conversational speech synthesis. *arXiv preprint arXiv:2106.10828*, 2021.

[10] Mohammad Soleymanpour, Michael T Johnson, Rahim Soleymanpour, and Jeffrey Berry. Synthesizing dysarthric speech using multi-talker tts for dysarthric speech recognition. *arXiv preprint arXiv:2201.11571*, 2022.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[12] Mingjian Chen, Xu Tan, Yi Ren, Jin Xu, Hao Sun, Sheng Zhao, Tao Qin, and Tie-Yan Liu. Multispeech: Multi-speaker text to speech with transformer. *arXiv preprint arXiv:2006.04664*, 2020.

[13] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713, 2019.

[14] Uday Kamath, John Liu, and James Whitaker. *Deep learning for NLP and speech recognition*, volume 84. Springer, 2019.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[16] Ankit Dash, Nikhil Subramani, Tejas Manjunath, Vishruti Yaragarala, and Shikha Tripathi. Speech recognition and correction of a stuttered speech. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1757–1760. IEEE, 2018.

[17] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[18] Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. *arXiv preprint arXiv:2204.01171*, 2022.

[19] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[20] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[21] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[23] Jaime Lorenzo-Trueba, Thomas Drugman, Javier Latorre, Thomas Merritt, Bartosz Putrycz, Roberto Barra-Chicote, Alexis Moinet, and Vatsal Aggarwal. Towards achieving robust universal neural vocoding. *arXiv preprint arXiv:1811.06292*, 2018.