



Frame-Level Stutter Detection

John Harvill¹, Mark Hasegawa-Johnson¹, Changdong Yoo²

¹University of Illinois at Urbana-Champaign, United States

²Korea Advanced Institute of Science and Technology, South Korea

{harvill12, jhasegaw}@illinois.edu, cd.yoo@kaist.ac.kr

Abstract

Previous studies on the detection of stuttered speech have focused on classification at the utterance level (e.g., for speech therapy applications), and on the correct insertion of stutter events in sequence into an orthographic transcript. In this paper, we propose the task of frame-level stutter detection which seeks to identify the time alignment of stutter events in a speech utterance, and we evaluate our approach on the stutter correction task. Limited previous work on stutter correction has relied on simple signal processing techniques and only been evaluated on small datasets. Our approach is the first large scale data-driven technique proposed to identify stuttering probabilistically at the frame level, and we make use of the largest available stuttering dataset to date during training. Predicted frame-level probabilities of different stuttering events can be used in downstream applications for Automatic Speech Recognition (ASR) as either additional features or part of a speech preprocessing pipeline to clean speech before analysis by an ASR system.

Index Terms: Stutter Detection, Speech Dysfluency, Sound Repetition, Word Repetition, Prolongation

1. Introduction

Stuttering is a type of irregularity in speech characterized by repetitions or prolongations of sounds, abrupt pauses, or interjections of sounds unrelated to the intended speech [1], prevalent in approximately 2% of adults [2]. ASR word error rates (WER) increase by a factor of three when test data contains stuttering [3]; stutter-augmented training can reduce the stuttering penalty slightly, but cannot overcome it entirely. Methods to identify and overcome stuttering are thus of great importance for people who stutter. Previous work in stutter detection has focused primarily on two tasks. First, several papers have focused on determining if stuttering is present anywhere in a given utterance. Classifications of this kind can be useful for speech therapists conducting stuttering assessments, because stuttering instances can be counted automatically instead of requiring human labels [4]. Stutter counts can then be used by speech pathologists to gauge the progress in improvement of communication skills for patients who stutter. The second class of prior work focuses on detecting stutter events as if they were words, and inserting them orthographically into the transcribed ASR output (e.g., [5, 6]). While automatic utterance-level information can assist in determining whether or not an utterance contains stuttering, they cannot be used to preprocess the signal in order to improve the performance of an ASR.

In this paper we propose a new task: *frame-level* stutter detection. The goal of this task is to determine where stuttering is present within a speech utterance. While this manuscript was in preparation, another group posted an arXiv paper with the first (to our knowledge) frame-level stutter detection, and demonstrated that the use of frame-level stutter detection significantly

improves ASR error rates [7]. The frame-level stutter detector in [7] is trained in a supervised fashion, using manually transcribed stutter start and end times, which require a great deal of human annotator time to acquire. Our approach avoids the necessity of manually labeled stutter start and end times by the use of (1) artificial stutter pretraining, followed by (2) multiple instance learning, implemented using max pooling.

We evaluate the quality of our model's predictions on the stutter correction task [8]. We remove sections of audio predicted to contain stuttering and have human evaluators judge the amount of stuttering in the processed speech and the degree to which the speech matches the intended transcription. The amount of stuttering judgement approximates recall by seeing how many instances of stuttering are missed by our system. The transcript matching judgement approximates precision by seeing how many instances of non-stuttered speech are removed and thus alter the transcript of the speech.

The main contributions of our paper are as follows: (1) We propose a pretraining technique that simulates different types of stuttering and allows us to pretrain our acoustic models with artificial frame-level labels. (2) We use the max-pooling technique from [9] to learn frame-level stutter event probabilities from utterance-level labels on a large collection of real stuttered speech. (3) We demonstrate improvement in stutter removal in our processed speech over the raw speech with little change to the perceived transcript.

2. Related Work

There exists a large body of literature on utterance-level stutter detection. Older works have relied on classical speech processing techniques such as analysis of vocal tone and formants [10] or use of classical signal processing features such as autocorrelation, signal envelope or Mel-Frequency Cepstral Coefficients (MFCC) as input for a neural network, k -nearest neighbor or Linear Discriminant Analysis (LDA) classifier [11, 12]. With the popularization of deep learning techniques, more recent work has focused on the use of neural networks to identify stuttering in speech. The most popular architectures are Long Short-Term Memory (LSTM) and Time Delay Neural Networks (TDNN) which are both used to model sequential data [13, 14, 15].

Stutter correction is a similar problem to stutter detection, except that the goal is not only to identify sections of stuttered speech but correct them such that speech no longer sounds dysfluent. Previous work on this task is limited, possibly due to the lack of a large dataset devoted to the problem¹. Dash et al. [17] use an amplitude threshold to remove prolongations under the assumption that prolonged sounds are of lower am-

¹UCLASS [16] has aligned transcriptions for a small number of audio files, but the stuttering events are indicated orthographically and not organized by dysfluency classes, making automatic analysis difficult.

plitude compared to surrounding speech. They rely on an ASR system to remove word or phrase repetitions by first converting the given speech to text, removing string repetitions, and then resynthesizing the speech using a text-to-speech (TTS) system. Arjun et al. [8] remove repetitions, long pauses, and prolongations through correlation analysis of MFCC and Linear Predictive Coefficients (LPC). The repetition removal algorithm relies on an amplitude-based word segmentation, and the methods are evaluated on a small private speech dataset.

3. Data

For our study, we use the recently released Stuttering Events in Podcasts dataset (SEP-28k) [14], which is the largest publicly-available stuttered speech dataset. SEP-28k has many more hours of speech data than previous stuttered speech datasets like UCLASS [16] and FluencyBank [18], with a total of 23 hours of annotated speech. The data are split into 3-second chunks, giving a total of 28,177 audio clips. Lea et al. [14] also release annotations for FluencyBank [18] while following the same annotation scheme as for SEP-28k, which adds an additional 3.5 hours of annotated speech data. When referring to SEP-28k for the remainder of the paper we mean the union of both SEP-28k and FluencyBank with annotations from [14].

SEP-28k Annotations: Each 3-second audio clip in SEP-28k is annotated by three raters with binary labels (presence or absence of speech dysfluency) for the stuttering dysfluencies of (1) Block (2) Prolongation (3) Sound Repetition (4) Word/Phrase repetition (5) No dysfluency (6) Interjection. These dysfluency types are defined as [14]: (1) Block - Gasps for air or stuttered pauses. (2) Prolongation - Extending a sound or syllable beyond its normal duration, often by a large factor: “v[vvvvv]ery much so.” (3) Sound repetition: Repeated syllables or sounds (not an entire word) “I like to [d-d-d]dance.” (4) Word Repetition - Repetition of whole words or multiword phrases “They went [went] to the store.” (5) No dysfluency - Audio does not contain any of the possible dysfluencies. (6) Interjection: Insertion of filler words like “um” or “uh.” Lea et al. [14] set the threshold for a positive label for any speech dysfluency at two (majority of the three raters). In this study we only consider Prolongation, Sound repetition, and Word repetition dysfluencies because of their importance for potential downstream speech preprocessing applications and the simplicity with which these dysfluencies can be simulated.

Partition: We make separate partitions for each dysfluency type. We balance each partition by assigning two thirds of the positive test samples to the train/dev set and one third to the test set. Due to the over-representation of negative samples, we up-sample the positives to occur as frequently as negatives during training. The held-out test samples are used for evaluation by human annotators (see Section 5).

Pretraining with LibriSpeech: Our proposed approach also relies on a large speech corpus for model pretraining, so we use LibriSpeech [19]. We only use the train-clean-360 partition, which contains 360 hours of speech. We require an alignment of the transcription to the audio at the phone level and use the existing alignment from <https://github.com/CoirentinJ/librispeech-alignments>.

Features: For both LibriSpeech and SEP-28k, we resample audio to 16kHz. We extract F0 (pitch, pitch-delta, voiced/unvoiced features) using pysptk [20] and the log Mel spectrogram using Librosa [21]. For F0, we limit the pitch range from 60-240Hz. For the log Mel spectrogram, we limit the dynamic range to 120db by clipping amplitudes lower than -120db

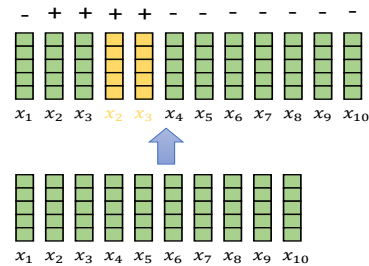


Figure 1: *Artificial stutter transformation and frame-level labeling scheme. In this toy SoundRep example, acoustic feature frames x_2 and x_3 together represent a phone and are jointly repeated once. A positive or negative label is indicated above each frame with ‘+’ or ‘-’, respectively, and repeated frames are colored yellow. The positive frame-level label is applied to both copies of the repeated phone.*

and setting them to -120db. We then normalize the log features to lie in the range zero to one where zero corresponds to -120db and one corresponds to 0db. We use a window size of 1024, hop length of 160 (10ms) and 80 Mel frequency bins.

4. Methods

Given that SEP-28k only contains utterance-level dysfluency labels, we have no ground truth alignment of the labels to specific sections of the audio for frame-level supervision. One approximate solution to this problem is to artificially introduce stuttering dysfluencies, because we know exactly where the dysfluencies occur and can thus create frame-level labels. Another solution is to use max-pooling as in [9]. Max-pooling encourages the label of the most likely stuttered frame in the audio sample to match the utterance-level label. Given these motivations, we discuss the details of each technique in the following sections.

Artificial Stutter Pretraining: There are two main goals for artificial stutter pretraining. The first is to pretrain our stutter detection model on a much larger dataset than just the labeled stuttering data alone. This allows our model to learn a rich set of acoustic features relevant for speech prior to training with task-specific data, which can improve generalization [22]. The second goal is to train the model with frame-level stuttering event labels, which are not available in current stuttered speech datasets. Since we artificially add in stuttering events, we know exactly where they occur. For each of the pretraining schemes discussed next, we supervise our model using the cross-entropy loss at each frame to encourage it to classify stuttered vs. non-stuttered frames (see Figure 1).

All artificial stuttering events are introduced into the spectrogram/F0 features and are performed on-the-fly during pretraining of the models. Pretraining schemes are discussed next: (1) Artificial Sound Repetition (SoundRep): Given a phone-level forced alignment, repeat phone segments anywhere from 2-5 times when a phone is selected (at random) to be repeated. We use $p = 0.3$ for phone selection probability. (2) Artificial Prolongation (Pro): Given a phone-level forced alignment, stretch the phone anywhere from 2-4 times the original length by interpolating the spectrogram and pitch features. We select phones with $p = 0.2$. (3) Artificial Word Repetition (WordRep): Given a word-level forced alignment, repeat word segments anywhere from 2-5 times. We select words to be repeated with $p = 0.3$. (4) Combination (Combo): Pretrain

with all three of the previous artificial dysfluencies, i.e. randomly select one of the three dysfluency types for each training sample. This model should predict a positive label for *any* of the three possible dysfluency types, whereas the previous approaches (SoundRep, Pro, WordRep) only predict a positive label for one particular type. Given that artificial dysfluencies injected into spectrogram/F0 features may produce sharp discontinuity artifacts that make their prediction trivial, we also randomly inject discontinuities into previously unmodified portions of the signal, after adding artificial stuttering, by randomly removing small chunks of audio. Specifically, for each frame, we choose whether or not to remove the next 5 frames (50ms) with probability $p = 0.005$.

Comparison to LibriStutter: Kourkounakis et al. [13] first introduced the idea of simulating artificial stuttering events by injecting dysfluencies into the LibriSpeech dataset [19] and created a corpus called LibriStutter. Here we note several distinctions between our approach and that of LibriStutter: (1) We add artificial dysfluencies to the spectrogram/F0 features (not the waveform) on-the-fly during pretraining instead of during one fixed preprocessing step, creating many more artificial training examples. (2) Instead of trying to disguise artifacts produced by artificial stuttering by smoothing the boundaries (which may still leave behind small artifacts), we simply add in distracting artifacts so that the model must learn to recognize repetitions or extensions of sounds instead of the artifact. (3) We use a phone-level alignment instead of a word-level alignment, making our sound repetition and prolongation dysfluencies correspond exactly to specific syllables, instead of approximating these by taking a fraction of the word, as is done in LibriStutter.

Frame-level Prediction Finetuning: Even though we lack ground truth frame-level labels for the authentic stuttered speech, we can encourage the model to learn to identify specific frames that correspond to stuttering by using a type of multiple-instance learning [23], implemented with max-pooling as in [9]. Given a sequence of acoustic feature frames x_1, x_2, \dots, x_N , where N is the sequence length, our model outputs a sequence of binary probability mass functions q_1, q_2, \dots, q_N where $q_n^- = p(l(x_n) = 0)$ and $q_n^+ = p(l(x_n) = 1)$. If $l(\cdot) = 1$ corresponds to a positive stutter label, we choose the frame t_{max} with maximum positive class probability as:

$$t_{max} = \operatorname{argmax}_n q_n^+ \quad (1)$$

Then we compute the cross-entropy loss L with respect to the utterance label distribution p_{utt} :

$$L = -p_{utt}^- \log q_{t_{max}}^- - p_{utt}^+ \log q_{t_{max}}^+ \quad (2)$$

and backpropagate the error. We initialize our model with one of the four pretrained models (SoundRep, Pro, WordRep, Combo), depending on which dysfluency type we want to recognize.

Model Architecture: We map the sequence of acoustic features to a sequence of binary probability distributions using a Bi-directional Long Short-Term Memory (BLSTM) plus Fully-Connected (FC) neural network (see Figure 2). The network consists of two BLSTM layers followed by three FC layers, where the output of the last FC layer has dimension two and is turned into a probability distribution using the softmax function. The BLSTM layers have a hidden size of 200, and the FC layers have a hidden size of 300. We use a dropout probability of 0.1 in all network layers, a learning rate of 0.0001, and the hyperbolic tangent activation function in the linear layers. Network parameters are updated using the Adam optimizer.

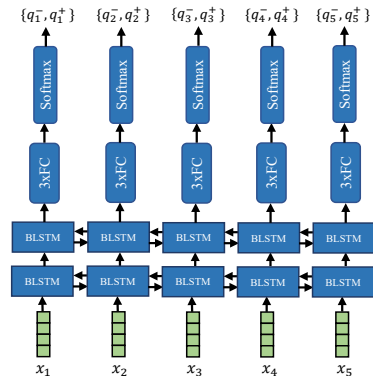


Figure 2: Neural network architecture used to compute binary distributions for each acoustic frame representing the probability that a frame contains a particular dysfluency.

No Pretraining Setting: To examine the importance of artificial stutter pretraining on frame-level stutter prediction, we train a model on the max-pooling objective using a randomly-initialized set of neural network weights instead of those from one of the pretrained models.

5. Experiments and Evaluation

To demonstrate the effectiveness of our approach on frame-level stutter classification, we rely on human evaluations of SEP-28k test samples for which we have removed stuttered speech. To remove stuttered speech, we rely on a thresholding approach. First, we smooth the predicted framewise probabilities and then search for frames where the probability is above a set threshold T_{up} . For each frame whose probability is above T_{up} , we find the first frame to the left and right whose probabilities fall below a second threshold T_{down} and remove the audio from all frames in between (in the time domain). We show results for $T_{up} = 0.5$ and $T_{up} = 0.25$, and we use $T_{down} = 0.1$ for all experiments.

There are two dimensions across which we must evaluate the utterances with stuttering removed: (1) How much stuttered speech is removed, and (2) How much non-stuttered speech is removed. To evaluate along these two dimensions, we perform Mean Opinion Score (MOS) trials by asking three raters on Amazon Mechanical Turk (AMT) to answer two questions about each test sample: (1) How much *dys.type* stuttering is present in the speech audio sample? Here *dys.type* refers to Prolongation, Sound Repetition or Word Repetition. The possible ratings are (1) None (no stuttering), (2) Very little (barely noticeable), (3) Some (noticeable), (4) Much (very obvious stuttering), and (5) Significant (severe stuttering). (2) How well does the speech audio match the given transcript? This question addresses how much non-stuttered speech is removed. The possible ratings are (1) Bad - Speech does not match transcript (2) Poor - Several extra/missing words compared to transcript (3) Fair - Some extra/missing words compared to transcript (4) Good - Speech mostly matches transcript (5) Excellent - Speech exactly matches transcript.

For the second question above, we specifically instruct evaluators to evaluate how well the *intended* speech matches the transcript such that they ignore dysfluencies. By phrasing the instructions in this way, we encourage raters to mark the original test utterances as an exact match for the transcript, thereby focusing the results of this question on utterances in which our

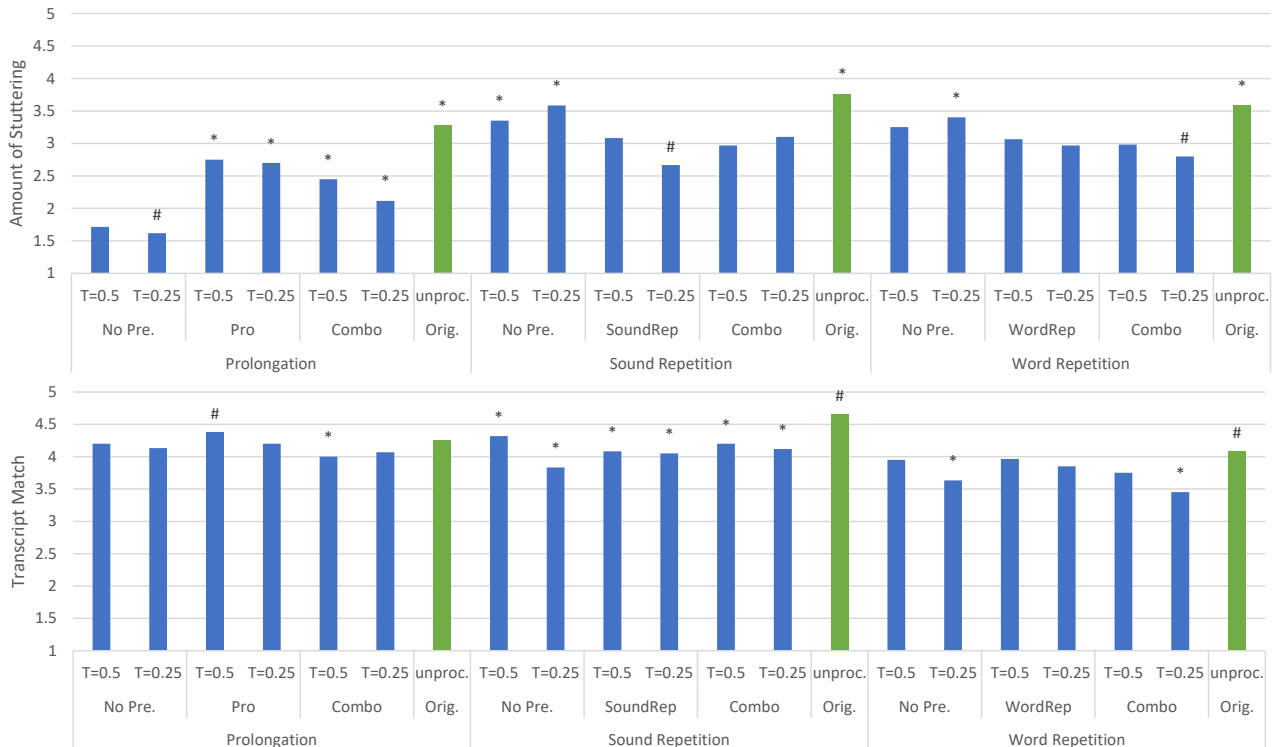


Figure 3: Evaluation results for amount of stuttering perceived and how well the audio transcript matches the intended transcript. ‘#’ indicates the best performing approach within each speech type. Methods marked with ‘*’ differ significantly ($p < 0.05$) from the best approach. The unprocessed speech is highlighted in green for easy visual comparison.

model has identified non-stuttered speech as stuttered and removed it. Our supplemental instructions for this task on AMT are as follows: “If the transcript is ‘they went home’ and the speaker utters ‘they w-w-w-went home’ (sound repetition) or ‘they went home went home’ (word repetition), the speech exactly matches the transcript. If the speaker were to utter ‘they home’ or ‘they went to home,’ the speech does NOT exactly match the transcript and the score should not be 5.”

For our evaluations, we choose 20 positive stuttered utterances each for Prolongation, Sound Repetition, and Word Repetition, making sure that only one stuttering type is present in each utterance. We then transcribe the speech in each utterance ourselves. We evaluate each original sample and multiple processed versions, each with a different removal threshold (T_{up}) or pretraining scheme. For each stutter type (Prolongation, Sound Repetition, Word Repetition), we test two thresholds ($T_{up} \in \{0.25, 0.5\}$) and three pretraining schemes: no pretraining (No Pre.), stutter-specific (for example, Prolongation uses the Pro pretrained model), and Combo pretraining.

6. Results

Results from our MOS tests are given in Figure 3. There are four trends in the data to notice. (1) Less stuttering is perceived in all processed speech compared to unprocessed speech. This implies our stutter detection algorithm correctly identifies the frames in which stuttering occurs. (2) There is a small reduction in the match of the perceived transcript in most processed speech. This tradeoff is expected since the system is imperfect and will inevitably remove non-stuttered speech. Experimental results show that the decrease in amount of stuttering is

larger than the decrease in transcript match for most approaches, suggesting a reasonable level of accuracy of our system for this task. (3) Model pretraining helps remove more stuttering for both sound and word repetition, but not prolongation. Removal of prolongation using a system with no pretraining (No Pre.) is an outlier compared to all other experiments, performing extremely well by removing almost all stuttering with no statistically-significant change in the perceived transcript compared to the unprocessed speech. It is possible that our representation of prolongation during pretraining is inaccurate, causing the Pro. and Combo systems to have lower accuracy than the No Pre. system. Since max-pooling is able to train a system to remove all prolongations, even without pretraining, it is possible that pretraining with artificial prolongations might bias the system against the natural prolongation examples it would otherwise learn. This could be explored in future work. (4) In general, the choice of threshold T_{up} does not have much effect on the performance.

7. Conclusions

This paper demonstrates that frame-level stutter detection can be trained using (1) artificial stutter pretraining for sound and word repetition, and (2) multiple-instance learning implemented via max-pooling. Future work will use frame-level stutter probabilities to improve the automatic transcription of stuttered and dysarthric speech, by preprocessing the signal using the methods of this paper and/or by using the method of [7] which provides frame-level stutter probabilities as an auxiliary feature to the ASR.

8. References

- [1] M. E. Wingate, "A standard definition of stuttering," *Journal of speech and hearing disorders*, vol. 29, no. 4, pp. 484–489, 1964.
- [2] A. R. Porfert and D. B. Rosenfield, "Prevalence of stuttering," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 41, no. 10, pp. 954–956, 1978.
- [3] V. Mendeleev, T. Raissi, G. Camporese, and M. Giollo, "Improved robustness to disfluencies in RNN-transducer based speech recognition," in *Proc. ICASSP*, 2021, pp. 6878–6882.
- [4] L. S. Chee, O. C. Ai, and S. Yaacob, "Overview of automatic stuttering recognition system," in *Proc. International Conference on Man-Machine Systems*, no. October, Batu Ferringhi, Penang Malaysia, 2009, pp. 1–6.
- [5] S. Alharbi, A. J. H. Simons, S. Brumfitt, and P. Green, "Automatic recognition of children's read speech for stuttering application," in *WOCCI: Workshop on Child Computer Interaction*, 2017.
- [6] S. Alharbi, M. Hasan, A. J. Simons, S. Brumfit, and P. Green, "Sequence labeling to detect stuttering events in read speech," *Computer Speech and Language*, vol. 62, pp. 101 052:1–13, 2020.
- [7] O. Shonibare, X. Tong, and V. Ravichandran, "Enhancing asr for stuttered speech with limited data using detect and pass," 2022. [Online]. Available: <https://arxiv.org/abs/2202.05396>
- [8] K. Arjun, S. Karthik, D. Kamalnath, P. Chanda, and S. Tripathi, "Automatic correction of stutter in disfluent speech," *Procedia Computer Science*, vol. 171, pp. 1363–1370, 2020.
- [9] J. Zhu, M. Hasegawa-Johnson, and N. L. McElwain, "A comparison study on infant-parent voice diarization," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7178–7182.
- [10] A. Czyzewski, A. Kaczmarek, and B. Kostek, "Intelligent processing of stuttered speech," *Journal of Intelligent Information Systems*, vol. 21, no. 2, pp. 143–171, 2003.
- [11] P. Howell and S. Sackin, "Automatic recognition of repetitions and prolongations in stuttered speech," in *Proceedings of the first World Congress on fluency disorders*, vol. 2. University Press Nijmegen Nijmegen, The Netherlands, 1995, pp. 372–374.
- [12] O. C. Ai, M. Hariharan, S. Yaacob, and L. S. Chee, "Classification of speech dysfluencies with mfcc and lpcc features," *Expert Systems with Applications*, vol. 39, no. 2, pp. 2157–2165, 2012.
- [13] T. Kourkounakis, A. Hajavi, and A. Etemad, "Fluentnet: End-to-end detection of stuttered speech disfluencies with deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2986–2999, 2021.
- [14] C. Lea, V. Mitra, A. Joshi, S. Kajarekar, and J. 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)*. IEEE, 2021, pp. 6798–6802.
- [15] S. A. Sheikh, M. Sahidullah, F. Hirsch, and S. Ouni, "Stutternet: Stuttering detection using time delay neural network," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 426–430.
- [16] P. Howell, S. Davis, and J. Bartrip, "The uclass archive of stuttered speech," *J. Speech Lang Hear Res*, 2009.
- [17] A. Dash, N. Subramani, T. Manjunath, V. Yaragarala, and S. Tripathi, "Speech recognition and correction of a stuttered speech," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 1757–1760.
- [18] N. B. Ratner and B. MacWhinney, "Fluency bank: a new resource for fluency research and practice," *J Fluency Disord*, 2018.
- [19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.
- [20] R. Yamamoto, J. Felipe, and M. Blaauw, "r9y9/pysptk: 0.1. 14," URL: <https://github.com/r9y9/pysptk>, 2019.
- [21] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.
- [22] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" in *Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*, 2010, pp. 201–208.
- [23] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Peréz, "Solving the multiple-instance problem with axis-parallel rectangles," *Artificial Intelligence Journal*, vol. 89, 1997.