MANAS Journal of Engineering
ISSN 1694-7398 — e-ISSN 1694-7398

Volume 12 (Issue 2) (2024) Pages 208-214
https://doi.org/10.51354/mjen.1538494

Detection and Identification of Stuttering Types Using Siamese Network

Venera Adanova[1]*, **Maksat Atagoziev**[2]

[1] Department of Computer Engineering, TED University, Ankara, 06420, Türkiye, venera.adanova@tedu.edu.tr, ORCID: 0000-0001-7247-0288

[2] Department of Computer Engineering, OSTIM Technical University, Ankara, 06374, Türkiye, maksat.atagoziev@ostimteknik.edu.tr, ORCID: 0000-0001-7799-7636

ABSTRACT

Stuttering is a complex speech disorder characterized by disruptions in the fluency of verbal expression, often leading to challenges in communication for those affected. Accurate identification and classification of stuttering types can greatly benefit persons who stutter (PWS), especially in an era where voice technologies are becoming increasingly ubiquitous and integrated into daily life. In this work, we adapt a simple yet effective Siamese network architecture, known for its capability to learn from paired speech segments, to extract novel features from audio speech data. Our approach leverages these features to enhance the detection and identification of stuttering events. For our experiments, we rely on a subset of the SEP-28k stuttering dataset, initially implementing a single-task model and gradually evolving it into a more sophisticated multi-task model. Our results demonstrate that transitioning the network from a single-task learner to a multi-task learner, coupled with the integration of auxiliary classification heads, significantly improves the identification of stuttering types, even with a relatively small dataset.

ARTICLE INFO

Research article

Received: 25.08.2024
Accepted: 04.12.2024

Keywords: Stuttering, dysfluency detection, multi-task learning

**Corresponding author*

1. Introduction

Stuttering, a.k.a. stammering, is a complex speech disorder that negatively affects the communication ability of 1% of the population. Persons who stutter (PWS) often know what they want to say, however the speech is interrupted by involuntary pauses and word or sound repetitions. Identification of stuttering in a speech is a challenging problem involving multiple disciplines such as pathology, psychology, acoustics, and signal processing.

With the advance of machine and deep learning the research done on speech domain have dramatically developed. Thus, current Automatic Speech Recognition (ASR) systems have good accuracy leading to voice assistants such as Alexa, Siri or Google. However, these systems are built based on fluent speech, and they fail to recognize speech accompanied with pauses and repetitions. Considering that voice technologies are becoming ubiquitous, if the developers continue assuming ideal speech scenarios, the future world seems to be the place where people with speech disorders will feel greatly deprived.

The studies on the stuttered speech have gained a speed recently as Apple released SEP28k stuttering dataset[14]. The data was collected from podcast where PWS are interviewed and is the first largest annotated dataset.

In this work, we perform our experiments on the subset of SEP28k dataset. We build a simple single task model which learns to differentiate between stuttering types, and then gradually convert the model into multi-task learner with multiple heads. We then observe improvements that these transformations bring into the classification task.

2. Related Work

Though speech recognition systems have evolved dramatically over the last decade, with the development of machine and deep learning, there is a scarce number of studies involving stuttering detection and identification. The majority of studies conducted on stuttering data aim to detect and identify the dysfluency types in audio recordings. These types of dysfluency are generally defined as: blocks, prolongations, sound/word/phrase repetitions, and interjections [14, 18]. Blocks are defined as involuntary pauses before words. Prolongations are elongated syllable, like *I am s[sss]ory*. Repetitions involve sound, word or phrase repetitions. For example, *I made [made] dinner* represents a word repetition. In order to avoid above defined stuttering types a person who stutters learns to use filler words like *um, uh, you know, etc*. These filler words are known as interjections. Note that the dysfluency types might be named differently in different studies.

2.1. Datasets

The main reason for the deficiency of studies in stuttered speech is the lack of data. Just like any speech related problem, detecting stuttering requires lots of data for accurate learning. Typically, works conducted on stuttering detection and identification are done based on some datasets, either in-house or public, and learn to classify between fluent and dysfluent speech (stuttering detection), and distinguish dysfluency types (identification).

The works that use in-house datasets [1, 8, 9, 11, 15] collect their own data, label them manually and report the stuttering detection and identification accuracy based on their own data. This type of dataset is very small and is not shared publicly.

There is only a handful number of publicly available stuttering datasets. The very first and also the smallest one is the UCLASS dataset [10]. It contains 457 audio recordings of monologues, conversations and readings, and only small amount of them has transcriptions. The dataset is not labeled according to dysfluency types.

The FluencyBank dataset [16] contains audio and video files with transcriptions for the interviews conducted for 32 adults and children who stutter. However, the dataset is not labeled.

The scarcity of labeled data led to the creation of synthetic dataset LibriStutter [12], which consists of 50 speakers (approximately 20 hours). The dataset was generated by injecting random stuttering to LibriSpeech dataset, which consists of fluent speech. The audio signals were segmented into four-second windows, and for every window either one of the stuttering events, as sound, word and phrase repetitions, prolongations, and interjections, were injected, or left untouched.

The largest dataset, *Stuttering Events in Podcasts* (SEP-28k) was released recently by [14]. SEP-28k is the first publicly available annotated dataset. It contains about 28000 3-second clips from podcast recordings. The SEP28k corpus also has 4144 3-second annotated clips from the FluencyBank dataset. [6] subsequently introduced an extended SEP-28k, which contains also the gender and speaker information. Along with the extended data they proposed a possible partitioning of data into train and test set.

[4] suggest their own dataset namely, Kassel State of Fluency (KSoF), which consists of 5500 clips of stuttered speech in German. The clips were labeled with the six stuttering event types: blocks, prolongations, sound/word/phrase repetitions, interjections and speech modifications. The last type is therapy specific and indicates whether the speaker's speech is modified after the therapy. The dataset also has some metadata, like the gender of a speaker, therapy status, type of microphone used, etc.

2.2. Stuttering Identification

One of the studies that aimed to identify all dysfluency types in stuttered speech was conducted by [13]. The work was based on UCLASS and LibriStutter datasets. They build a deep neural network, named FluentNet, consisting of Squeeze-and-Excitation Residiual Network and bidirectional LSTM. The four-second long audio clips from the dataset are converted to spectrograms using Short Time Fourier Transform (STFT), and these spectrograms formed the inputs to FluentNet. Though the work demonstrates promising results it has some limitations. The results were reported for the small subset of speakers, probably because they needed to label the data manually first. Also, the work does not consider fluent speech and their models learn to classify between stuttering types only. The other limitation is that they trained a model for every dysfluency type leading to computationally expensive system due to large number of parameters.

As an alternative to the FluentNet, [17] proposed their own time delay neural network, called StutterNet. The model was trained and tested on UCLASS. The audio recordings for over 100 speakers were manually labeled as in [13]. Only core behaviors (prolongations, blocks, and repetitions) and fluent part of the speech were considered. Each audio recording was sliced into four-second clips and for each clip mel-frequency cepstral coefficients (MFCCs) were computed. The MFCC features are then fed to StutterNet as features. In contrast to the aforementioned approach a single model was trained for identification of all types of dysfluency and a slight improvement was observed.

In [7] KSoF is used for stuttering detection. Each audio

recording is sliced into three-second clips. For each clip the features are extracted from pretrained wav2vec 2.0 [3] network. Wav2vec network is learned on large amount of fluent speech, and takes raw data as an input and produces a feature vector describing each audio data. In [7] before using the wav2vec features the network was fine-tuned using SEP-28k dataset. The features obtained from the fine-tuned network are subsequently fed to SVM model for dysfluency identification purposes.

The work in [19] reports on the dysfluency identification results based on multi-task and adversarial leaning using SEP-28k dataset. Each clip in the dataset was represented by 20 MFCC features. Instead of learning a single task in the network, a multi-task learning was used. A network consists of three parts. First part learns to classify between fluent and dysfluent speech, second part learns to classify between dysfluency types, and the third part is just an auxiliary part which prevents the network from overfitting and learns to classify between show names from which the clips were taken. All three parts share a single encoder, meaning that they share weight in the encoder part.

[2] proposed automatic detection and correction for three types of dysfluencies: repetitions, prolongations and long pauses, using signal processing methods. In this research MFCCs and Linear Predictive Coefficients (LPC) are used to extract the features. The downside of this approach is that it largely relies on empirical thresholds to detect dysfluency types. Thus, to detect repetitions the MFCC and LPC features of consecutive words are extracted and their correlation factor is computed. If the correlation is above a certain threshold then the words are considered to be similar, and one of the words is deleted.

Another two approaches [1, 8] also use empirical threshold to detect prolongations in speech. These studies deal with detection and correction of prolongations and repetitions. To correct the prolongations, amplitudes of the audio signal are compared against predefined threshold value, and the values below that threshold are deleted. To detect repeated words the audio signal is first converted to text, in the text format the repeated words are detected and deleted. Subsequently, the text is converted back to speech.

3. Dataset

In our work, we use a subset of SEP28k. The audio data was collected from eight podcast shows and every episode was divided into 3 second clips. The total number of clips in the dataset is 28177 (approx. 23.5 hours) and every clip was annotated by three annotators. The annotations for every clip have two types: stuttering and non stuttering. Stuttering types include *prolongation*, *block*, *interjection*, *sound/word repetition* and *no stuttered word*, where former five types represent dysfluency types and the last one is the fluent type. Non stuttering types include *unintelligible*, *natural pause*, *unsure*, *music* and *poor audio quality*. We are mainly interested in the stuttering types.

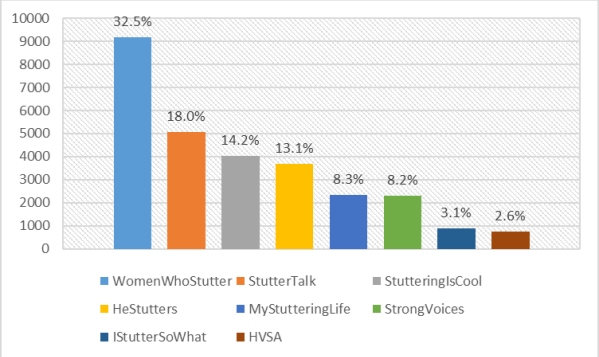

Figure 1: The distribution of podcast shows in the dataset. WomenWhoStutter and HeStutters have common host.

The SEP28k is a challenging dataset and these challenges come in three ways. First, it is highly imbalanced. More than half of the data contains fluent speech, and approximately 10% is given for a particular dysfluency type. Second, each clip might have several annotations. Thus, a single clip might contain both prolongation and block dysfluency types, while being also annotated by one of the annotators as a fluent speech. Lastly, it is also imbalanced in terms of speaker. Thus, host speech dominates 60% of the data. Also, the distribution of podcast shows is imbalanced as shown in Figure 1. The number of clips for one of the shows, Women Who Stutter, form 33% of overall clips. Considering that Women Who Stutter and He Stutters share the same host (Pamela Mertz), a large amount of clips is dominated by the speech of a single person.

As has been discussed previously, each clip in the dataset might have multiple labels. Typically, it is not clear how much the annotators should be trusted. For example, consider a clip where three annotators identify it as having prolongation, two annotators also note that some part of it has blocks, and one annotator states that it has no other disfluency. We do not know how much weight should be given to that single label. It might also be the case that each annotator votes for a different stuttering type, further increasing our confusion. All studies that have used this dataset have somehow neglected to mention their approach to dealing with these labels. We, on the other hand, decided to take a more cautious approach by constructing a smaller subset from SEP-28k, which we refer to as the confidence list. This list consists of clips that were consistently assigned a single type by all three annotators. However, the *interjection* type was never annotated alone, as it always co-occurs with the *no*

stuttered word type. Therefore, we also include clips where all three annotators selected both *no stuttered word* and *interjections*. Moreover, clips where all three annotators selected both *no stuttered word* and *natural pause* are also included. We also formed a confidence list for the FluencyBank dataset. However, only clips with disfluency types were included, as the percentage of fluent clips already constitutes a large portion of the dataset. In total, our dataset contains 3,901 clips, and the distribution of different types is illustrated in Fig. 2. It is important to note that the dataset is highly imbalanced, with 65% of the clips consisting of fluent speech.

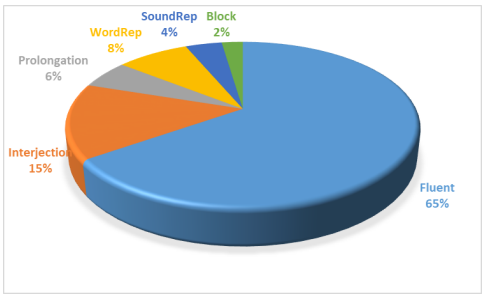

Figure 2: The distribution of stuttering types in our dataset. Observe that fluent data forms 65% of the dataset.

4. Proposed Framework

Given the audio clips, we initially extract their Mel Frequency Cepstral Coefficients, MFCCs. The extracted MFCCs are further fed to our baseline model, training which we learn new embeddings (features) for the clips. The baseline model that we use to extract embeddings is the Siamese network with contrastive loss shown in Fig. 4(gray region). The choice of this network is not random, as we favor it because of its ability to learn well the data which consists of small number of representatives from each class. This is indeed the case for our data.

There are two inputs to the network and two identical subnetworks that share the weights. For each pair the two subnetworks produce embeddings which are then used to compute the Euclidean distance between a pair of inputs. The main goal of the network is not to learn to classify different stuttering types but to differentiate between them.

The subnetwork consist of three blocks. Each block contains a convolutional layer followed by max pooling and dropout layers. The last layer does global averaging which returns the desired 64x1 dimensional vector. The details on input and output dimensions are illustrated in Fig. 3.

As a loss function for the baseline model we use contrastive loss which is defined as following:

$$L_{baseline} = y \cdot d^2 + (1 - y) \cdot \max(margin - d, 0)^2 \quad (1)$$

where, y is a true label, 1 if the audio pairs are of the same class, 0 otherwise, and d is the Euclidean distance between the outputs of twin network embeddings. Margin is 1.

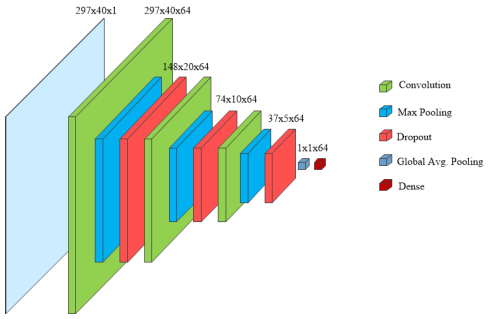

Figure 3: The architecture of our subnetwork.

Our baseline model does a single task learning (STL). During our experiments we extend the model to do multi task learning (MTL). Thus, we add a classification head to the model so that the model can also learn to classify between the six stuttering types. In our other experiment, we add another classification head, which also forces the network to differentiate between show types. These extension are shown in Fig. 4, where each of the classification heads is fed with the new features computed for the first input of the baseline model. For both of these classification heads we use sparse categorical crossentropy loss function. The addition of auxiliary tasks as this, are suggested in the literature [5, 19] for generalization and regularization purposes.

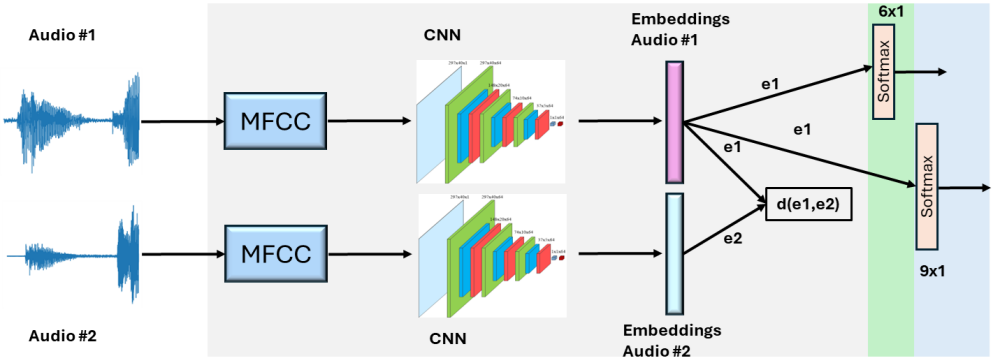

Figure 4: Proposed framework. The model within the gray box is the baseline model (BM). Then we add one classification head that learns to classify between six stuttering types (gray plus green regions) (BSM). The third model is formed by adding one more classification head to the second model (gray plus green plus blue regions), which also learns to classify shows (BSSM).

Hence, the overall loss of an MTL model is given by:

$$L = \lambda_{baseline} \cdot L_{baseline} + \lambda_{auxiliary} \cdot L_{auxiliary} \quad (2)$$

As was mentioned before, our model learns new features (embeddings) for the audio data, which are subsequently fed to machine learning models for classification purposes.

5. Experimental Results

5.1. Features

We first compute MFCC features for every audio clip, which are then fed to our model. All clips are read with the sampling frequency of 16000. The MFCC features are computed using *speechpy* library, with the frame length of 0.025, frame stride of 0.01 and number of filters of 40. For every 3 seconds long clip, using these parameters, we produce 297x40 features.

5.2. Data Augmentation

The data is highly imbalanced, so we produce new samples using augmentation techniques. The augmented data is used only during the training process. Since fluent speech already takes up to 65% of the data we only augmented the clips with dysfluency labels.

Data augmentation is done using *audiomentations* library. For every dysfluent clips in the training set we add Gaussian noise with min. amplitude of 0.001 and max. amplitude of 0.015, time stretch up/down to 25%, and shift pitch up/down 4 semitones.

5.3. Data Splitting

The data is divided into three: train, validation and test sets. The train set contains clips of 10 most frequent speakers, plus the dysfluency clips from FluencyBank, which gives us overall 2434 clips. What is left is divided between validation and test sets. Thus validation set consist of 734 clips of next most frequent speakers, and the test set contains 733 clips of less frequent speakers.

5.4. Training

We train three different models: baseline model, baseline model with stuttering classification head, baseline model with stuttering classification and show classification head. We call them BM, BSM and BSSM (See Figure 4), respectively. The latter two are MTL models. For all three models we use Adam optimizer with $learning\_rate = 0.001$, batch size is 32 and the number of epochs is 20. For the first MTL model the weights of losses are equal, thus $\lambda_{baseline} = 0.5$ and $\lambda_{stuttering} = 0.5$. For the second MTL model we pay less importance to the show classification head as it is used more like regularization, hence the weights are distributed as $\lambda_{baseline} = 0.4$, $\lambda_{stuttering} = 0.4$, $\lambda_{show} = 0.2$.

5.5. Experimental Results

After training the models, we extract new features for the data and perform classification on Support Vector Machine (SVM) and K-Nearest Neighbor (KNN). The SVM with polynomial kernel of degree 7, and the C parameter of 100 is found to be most optimal one for the validation set. For KNN model the k is 1 in all our results. All performance evaluations in this section are conducted on test data, which was not used during the training of either the feature extraction network or the SVM and KNN models.

Table 1 presents the results of the classification done using different model combinations. Observe that, when we switch to MTL models (BSM and BSSM) there is

a significant improvement in the f1-scores of different stuttering types. Although switching to BSSM does not introduce improvements to SVM model's classification performance, it has major impact on KNN model. Recall that, our data has only 2% of *block* dysfluency types. So the models find hard to learn it, hence, we observe such small number of detected block types.

Table 1: F1-score for stuttering classification. (P: Prolongation, B: Block, SR: Sound Repetition, WR: Word Repetition, I: Interjection, F: Fluent, BM: Baseline Model, BSM: Baseline with stuttering classification head, BSSM: BSM with show classification head).

F1-Score						
Model	P	B	SR	WR	I	F
BM_SVM	0.14	0.03	0.08	0.06	0.11	0.72
BM_KNN	0.10	0.00	0.11	0.09	0.14	0.66
BSM_SVM	0.31	0.04	0.13	0.11	0.33	0.74
BSM_KNN	0.21	0.08	0.12	0.11	0.29	0.64
BSSM_SVM	0.26	0.00	0.15	0.08	0.30	0.74
BSSM_KNN	0.36	0.04	0.11	0.10	0.34	0.68

The classification results for the case when only dysfluency types are considered is reported in Table 2. Observe that the *block* type classification accuracy improves in this case. While the baseline model learns the embeddings that differentiate between different stuttering types, adding stuttering classification head enforces the model to learn the embeddings that also represent the stuttering types themselves. Once again, we can observe the improvement in classification that bring the MTL models. The embedding computed using BSM model brings almost 100% improvement to *prolongation*, *block* and *sound repetition* types' f1-score, and almost 400% improvement for *word repetition* type in SVM, in contrast to BM model. The BSSM embedding brings significant improvements to KNN's classification performance.

Table 2: Results of classification dysfluency types only. (P: Prolongation, B: Block, SR: Sound Repetition, WR: Word Repetition, I: Interjection, BM: Baseline Model, BSM: Baseline with stuttering classification head, BSSM: BSM with show classification head).

F1-Score					
Model	P	B	SR	WR	I
BM_SVM	0.19	0.09	0.15	0.08	0.40
BM_KNN	0.13	0.10	0.21	0.18	0.40
BSM_SVM	0.35	0.21	0.27	0.27	0.61
BSM_KNN	0.26	0.11	0.22	0.19	0.60
BSSM_SVM	0.39	0.12	0.21	0.27	0.59
BSSM_KNN	0.43	0.16	0.22	0.31	0.58

We combine the five dysfluency types into one group and name the group as non fluent type and perform the binary classification. The f1-scores are illustrated in Table 3. It turns out that MTL models improve fluent data classification, while non fluent type has small improvement. Thus, BSM embeddings bring 55% improvement to fluent data classification in SVM, while it is 9.7% for non fluent data.

Table 3: F1-score for binary classification.

F1-Score		
Model	Fluent	Non Fluent
BM_SVM	0.49	0.41
BM_KNN	0.66	0.41
BSM_SVM	0.76	0.44
BSM_KNN	0.64	0.44
BSSM_SVM	0.73	0.45
BSSM_KNN	0.68	0.45

Table 4: Accuracy results for binary, dysfluency type only and total classification.

Accuracy			
Model	Binary	Dysfluency	Total
BM_SVM	0.45	0.25	0.52
BM_KNN	0.56	0.25	0.45
BSM_SVM	0.66	0.43	0.57
BSM_KNN	0.56	0.35	0.47
BSSM_SVM	0.64	0.42	0.57
BSSM_KNN	0.59	0.40	0.51

The accuracy of classification results are reported in Table 4. The accuracy of binary classification do not change much for KNN. However, SVM achieves 47% improvement with the MTL models. We observe significant accuracy improvement for the dysfluency type classification (when only 5 dysfluent categoris considered) both in KNN and SVM. We observe that the total accuracy does not change while the dysfluency type classification improves.

6. Conclusion

Stuttering detection is a complex problem. The complexity increases with the scarcity of the data. The existing datasets are highly imbalanced which leads to the learned models that favor the majority class. Since the data size is too small it is usually impossible to build complex networks because of the overfitting. By constructing networks that learn multiple tasks we can regularize the weights of the network. In this work, we built and learnt rather simple network on small and highly imbalanced dataset. Our findings show that by converting the network from a single task learner to a multi task learner,

and adding some auxiliary classification heads, we can significantly improve the identification of stuttering types.

Acknowledgements

This work is funded by TEDU BAP Grant No. T-22-B2010-90108.

References

[1] Amruth, V., Lavanya, K., Manoj, N., Umme, H., and Deepika, M. B. (2020). A novel approach for stutter speech recognition and correction. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, 8:544–547.

[2] Arjun, K. N., Karthik, S., Kamalnath, D., Chanda, P., and Tripath, S. (2020). Automatic correction of stutter in dysfluent speech. *Procedia Computer Science*, 171:1363–1370.

[3] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, pages 12449–12460.

[4] Bayerl, S., von Gudenberg, A. W., Hönig, F., Nöth, E., and Riedhammer, K. (2022a). KSoF: The kassel state of fluency dataset – a therapy centered dataset of stuttering. In *Proceedings of the Language Resources and Evaluation Conference LREC*, pages 1780–1787. European Language Resources Association.

[5] Bayerl, S. P., Gerczuk, M., Batliner, A., Bergler, C., Amiriparian, S., Schuller, B., Nöth, E., and Riedhammer, K. (2023). Classification of stuttering – the compare challenge and beyond. *Computer Speech & Language*, 81.

[6] Bayerl, S. P., Wagner, D., Nöth, E., Bocklet, T., and Riedhammer, K. (2022b). The influence of dataset partitioning on dysfluency detection systems. In Sojka, P., Horák, A., Kopeček, I., and Pala, K., editors, *Text, Speech, and Dialogue*, pages 423–436, Cham. Springer International Publishing.

[7] Bayerl, S. P., Wagner, D., Nöth, E., and Riedhammer, K. (2022c). Detecting dysfluencies in stuttering therapy using wav2vec 2.0. In *Interspeech 2022*. ISCA.

[8] Dash, A., Subramani, N., Manjunath, T., Yaragarala, V., and Tripathi, S. (2018). Speech recognition and correction of a stuttered speech. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1757–1760.

[9] Heeman, P., Lunsford, R., McMillin, A., and Yaruss, J. S. (2016). Using clinician annotations to improve automatic speech recognition of stuttered speech. In *Interspeech*, pages 2651–2655.

[10] Howell, P., Davis, S., and Bartrip, J. (2009). The UCLASS archive of stuttered speech. *Journal of Speech, Language, and Hearing Research*, 52:556–596.

[11] Howell, P. and Sackin, S. (1995). Automatic recognition of repetitions and prolongations in stuttered speech. In *Proceedings of the First World Congress on Fluency Disorders*, pages 372–374.

[12] Kourkounakis, T., Hajavi, A., and Etemad, A. (2020). Detecting multiple speech dysfluencies using a deep residual network with bidirectional long-short-term memory. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2020, pages 6089–6093.

[13] Kourkounakis, T., Hajavi, A., and Etemad, A. (2021). Fluentnet: End-to-end detection of stuttered speech dysfluencies with deep learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2986–2999.

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

[15] Nöth, E., Niemann, H., Haderlein, T., Decher, M., Eysholdt, U., Rosanowski, F., and Wittenberg, T. (2000). Automatic stuttering recognition using hidden markov models. In *INTERSPEECH*.

[16] Ratner, N. B. and MacWhinney, B. (2018). Fluency bank: A new resource for fluency research and practice. *Journal of Fluency Disorders*, 56:69–80.

[17] Shakeel, A. S., Md, S., Fabrice, H., and Slim, O. (2021). Stutternet: Stuttering detection using time delay neural network. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 426–430.

[18] Shakeel, A. S., Md, S., Fabrice, H., and Slim, O. (2022a). Machine learning for stuttering identification: Review, challenges and future directions. *Neurocomputing*, 514:385–402.

[19] Shakeel, A. S., Md, S., Fabrice, H., and Slim, O. (2022b). Robust stuttering detection via multi-task and adversarial learning. In *European Signal Processing Conference (EUSIPCO)*.