Advancing Stuttering Detection via Data Augmentation, Class-Balanced Loss and Multi-Contextual Deep Learning

Shakeel A. Sheikh, Md Sahidullah, Fabrice Hirsch, Slim Ouni

Abstract—**Stuttering is a neuro-developmental speech impairment characterized by uncontrolled utterances (interjections) and core behaviors (blocks, repetitions, and prolongations), and is caused by the failure of speech sensorimotors. Due to its complex nature, stuttering detection (SD) is a difficult task. If detected at an early stage, it could facilitate speech therapists to observe and rectify the speech patterns of persons who stutter (PWS). The stuttered speech of PWS is usually available in limited amounts and is highly imbalanced. To this end, we address the class imbalance problem in the SD domain via a multi-branching (MB) scheme and by weighting the contribution of classes in the overall loss function, resulting in a huge improvement in stuttering classes on the SEP-28k dataset over the baseline (*StutterNet*). To tackle data scarcity, we investigate the effectiveness of data augmentation on top of a multi-branched training scheme. The augmented training outperforms the MB *StutterNet* (clean) by a relative margin of 4.18% in macro F1-score ($\mathcal{F}_1$). In addition, we propose a multi-contextual (MC) *StutterNet*, which exploits different contexts of the stuttered speech, resulting in an overall improvement of 4.48% in $\mathcal{F}_1$ over the single context based MB *StutterNet*. Finally, we have shown that applying data augmentation in the cross-corpora scenario can improve the overall SD performance by a relative margin of 13.23% in $\mathcal{F}_1$ over the clean training.**

Index Terms—**Stuttering detection, speech disorder, data augmentation, class balanced learning.**

I. INTRODUCTION

SPEECH impairments, often known as speech disorders, are difficulty in producing speech sounds. These speech difficulties usually take the form of dysarthria, cluttering (poorly intelligible speech), lisping, apraxia, and stuttering [1]–[5]. Only a few percentage (5-10%) of the world population can produce accurate speech units, the rest encounter some type of speech disorder in their life span [6]. Of these speech impairments, stuttering is the most predominant one [1]. Stuttering[1], is a neuro-developmental speech disorder, in which the flow of speech is disturbed by abnormally persistent and involuntarily speech sounds, which usually take the shape of *core behaviors:* including blocks, prolongations and repetitions [1]. Stuttering is complex and the several factors that lead to it are delayed childhood development, stress, and speech motor abnormalities [1]. In [7], Smith and Weber put forward the multifactorial dynamic pathway theory, where they argued that stuttering occurs due to the failure of the nervous system. People with stuttering (PWS) exhibit impairment in sensorimotor processes which are responsible for the production of speech, and its direction is influenced by emotional and linguistic aspects.

In conventional stuttering detection (SD) and therapy sessions, the speech therapists or speech-language pathologists manually analyze the PWS' speech [8]. The speech therapists observe and monitor the speech patterns of PWS to rectify them [1]. This convention of SD is very laborious and time-consuming and is also inclined toward the idiosyncratic belief of speech therapists. In addition, the automatic speech recognition systems (ASR) are working fine for normal fluent speech, however, they are unsuccessful in recognizing the stuttered speech [9], which makes it impractical for PWS to easily access virtual assistants like Apple Siri, Alexa, etc. As a result, interactive automatic SD systems that provide an impartial objective, and consistent evaluation of stuttering speech are strongly encouraged. The SD can also be used to adapt and improve ASR virtual assistant tools for stuttered speech. Despite the fact that having numerous potential applications, very little research attention has been given to the domain of SD and measurement. The detection and identification of stuttering events can be quite a difficult and complex problem due to several variable factors including language, gender, age, accent, speech rate, etc.

The main goal of this work is to build robust automatic stuttering detection systems capable of detecting multiple stuttering types based on speech modality, which later on can be deployed as a tool in real-world settings by providing a means to both PWS and speech therapists to keep track of the stutter speech. These systems can later on be further improved by providing a feedback mechanism to PWS and help them to rectify their stuttering.

A significant amount of work is done in the detection of other speech disorders [10] like dysarthria [11] and Parkinsons [12], but stuttering has not been addressed widely even though it is the most common one. In this paper, we propose a deep learning framework for robust SD. The automatic

Shakeel A. Sheikh, Md Sahidullah, and Slim Ouni are from Université de Lorraine, CNRS, Inria, LORIA, F-54000, Nancy, France (e-mail: shakeel-ahmad.sheikh, md.sahidullah, slim.ouni@loria.fr).

Fabrice Hirsch is from Université Paul-Valéry Montpellier, CNRS, Praxiling, Montpellier, France (e-mail: fabrice.hirsch@univ-montp3.fr).

[1]Stuttering is also called stammering. In this paper, we will use the terms disfluency, stuttering, and stammering interchangeably.

detection of stuttering can help in the treatment of stuttering if detected at an early age [1].

Most of the computer-based SD methods are based either on ASR systems [13], [14] or language models [15], [16]. These methods are two-stage approaches that first convert the acoustic speech signals into their corresponding spoken textual modality, and then detect stuttering by the application of language models. Even though this ASR-based two-stage approach for identifying stuttering has shown promising results, the dependence on the ASR unit makes it computationally costly and prone to error. Moreover, the adaption towards the ASR task results in the possible loss of stuttering relevant information such as prosodic and emotional content.

In recent decades, the applications of deep learning have grown tremendously in speech recognition [17], speaker recognition [18], speech synthesis [19], emotion detection [20], voice conversion [21], voice disorder detection [10] including Parkinson's disease detection [22] and dysarthric speech detection [11], [23]. Inspired by the human auditory temporal mechanism, Kodrasi *et al.* [23] recently proposed a convolutional neural network based dysarthric speech detection method by factoring the speech signal into two discriminative representations including temporal envelope (stress, voicing, and phonetic information) and fine structure (breathiness, pitch, and vowel quality) and reported the state-of-the-art results in dysarthria detection. However, the application of deep learning in SD is limited. The acoustic properties of speech disfluencies are different for different disfluencies which can help to discriminate from fluent voice. Due to the presence of these acoustic cues in the stuttered embedded speech, deep learning models can be used to exploit these acoustic cues in the detection and identification of stuttering events. Most of the SD existing methods employ spectral features including *mel-frequency cepstral coefficients* (MFCCs) and spectrograms or their variants that capture the stuttering-related information.

The earlier studies in this domain applied shallow deep learning approaches in SD. In 1995, Howell *et al.* [24] employed two fully connected artificial neural networks for the identification of two types of disfluencies, namely, repetition and prolongation. They extracted autocorrelation features, envelope parameters, and spectral information and used these features as an input to the artificial neural networks for SD. The network was trained on 12 speakers with 20 autocorrelation features and 19 vocoder coefficients. In 2009, Ravikumar et al [25] proposed multi-layered perceptron for the repetition type of stuttering. The network was trained by using MFCC input features on 12 different disfluent speakers. In 2019, B. Villegas *et al.* [26] trained multi-layer perceptron on 10-dimensional respiratory features for the block SD. They used a dataset of 68 Latin American Spanish speakers in their case study. In a recent study, Kourkounakis *et al.* [27] proposed residual network and bi-directional long short-term memory (ResNet+BiLSTM) based binary deep learning classifiers for the detection of six different types of disfluencies including prolongation, word repetition, sound repetition, phrase repetition, and false starts. They used spectrograms as input features and reported promising results on a small subset (25 speakers) from the UCLASS dataset [27]. In another study Lee *et al.* [28]

curated a large stuttering dataset (SEP-28k) and utilized the convolutional long short-term memory (ConvLSTM) model for the detection and identification of six types of stuttering, namely, blocks, prolongations, sound repetitions, word repetitions, and interjections. The model takes 40-dimensional input MFCCs, eight-dimensional articulatory features, 41-dimensional phoneme feature vector, and three-dimensional pitch features. Sheikh *et al.* [29] recently proposed a single multi-class time delay neural network (TDNN) based *Stutter-Net* classifier which is capable of detecting core behaviors and fluent speech segments and gives promising detection performance on a larg subset of UCLASS (100+) speakers compared to the state-of-the-art classifiers. The model solely takes 20-dimensional MFCC features as an input. In another recent study M. Jouaiti *et al.* [30] introduced phoneme-based bi-drectional long short-term memory (BiLSTM) model for SD by mixing the SEP-28k and UCLASS datasets. The model is trained on 20-dimensional MFCCs and 19-dimensional phoneme input features. The disfluencies considered in this work are prolongations, repetitions, and interjections. A detailed summary and comparison of various feature extraction methods and classifiers can be found in [31].

This work provides a complete deep learning framework for robust SD following our preliminary initial investigations [29] in which *StutterNet* yields state-of-the-art SD results. In this study, we identify the limitations of *StutterNet* and proposed further advancements to address those drawbacks. Our main contributions are summarized below:

- *Solution for class imbalance problem*: The standard stuttering datasets suffer from class imbalance problems. We introduced two strategies based on weighted loss and multi-branch architecture to tackle the class imbalance problem.
- *Introducing data augmentation*: The stuttering datasets have a limited amount of training data and this makes it difficult to apply advanced deep learning models with a large number of parameters. To address this limitation, this work introduces audio data augmentation. To the best of our knowledge, this is the first work to apply audio data augmentation in the SD problem.
- *Introducing multi-contextual architecture*: Stuttering detection is a special type of speech characterization problem where each class (i.e., stuttering types) has a varying duration. For example, block lasts for a shorter duration than prolongations and repetitions. Therefore, a fixed length of context in the basic *StutterNet* framework might not be the optimized one for detecting all types of stuttering. We introduce a multi-contextual architecture that uses different context lengths in a parallel fashion.

The remaining of the paper is organized as follows. Section II describes the *StutterNet* and analyzes its deficiencies. Section III discusses the proposed methodology for addressing the deficiencies. Section IV details the experimental design, metrics used and datasets. Section V discusses the experimental results on class balanced, data augmentation, MC *StutterNet* and cross corpora scenario. Finally, in Section VI, we conclude with the possible future directions.

II. STUTTERNET: OVERVIEW & LIMITATIONS

A. StutterNet

Most of the earlier work employed only a small set of disfluent speakers in their experimental studies, and has approached the SD problem as a binary classification problem: disfluent vs fluent identification or one vs other type [31]. The *StutterNet* we propose in our earlier work, is a time delay neural network based architecture that has been used to tackle the SD as a multi-class classification problem. The *StutterNet* takes 20 MFCCs as input features with a frame length of 20 ms, mean-normalized with a hop length of 10 ms. Usually, the initial layers in a standard deep neural network learn wider contexts when processing a temporal input signal. However in *StutterNet* network as shown in Table I, the initial layers learn and capture only smaller contexts and the deeper ones compute the activations from a wider context, thus the deeper/higher layers are able to capture and learn the longer temporal contexts. The network consists of five time delay layers with the first three focusing on $[t-2,\ t+2]$, $\{t-2,t,\ t+2\}$, $\{t-3,t,\ t+3\}$ and the other two on $\{t\}$ contextual frames, respectively. The TDNN layers are having a dilation of $(1, 2, 3, 1, 1)$ respectively. This is followed by a two-layered BiLSTM unit, mean and standard deviation pooling layer, three fully connected (FC) layers, and a softmax layer on top of the network that reveals the classification scores of stuttering disfluencies. A ReLU nonlinearity activation function and a 1D batch normalization are applied after each layer except the statistical pooling layer. We apply dropout of 0.3. in FC layers.

Consider an input speech sample with T frames. The first five layers of *StutterNet* focus on the small context of speech frames. For example, layer 2 takes as input the sliced output of layer 1 at time frames of $\{t-2,\ t,\ t+2\}$, which results in capturing a total temporal context of 9 with the help of the previous layer's context of $[t-2,\ t+2]$. Similarly, layer 3 sees a total context of 15 time frames. The statistical pooling layer computes and concatenates the mean and standard deviation by aggregating all T output frames at the end of BiLSTM output. The statistical pooling layer accumulates the information across the temporal dimension which makes it suitable for subsequent layers to operate on the entire temporal speech segment. The *StutterNet* is trained to classify 5 different types of stuttering including *core behaviors* and the fluent part of the PWS. For detailed *StutterNet* architecture, please refer to [29].

B. Limitations

Although this network has shown promising results on a SEP-28K dataset in SD, it has several deficiencies. First, it does not generalize quite well on unseen data and leads to overfitting due to the limited amount of data available for training. In addition to data scarcity, the SEP-28k dataset was collected from podcasts in a clean environment, which makes the trained *StutterNet* difficult to generalize in other environmental conditions. Second, obtaining class-balanced datasets is extremely difficult and very expensive in the speech domain and the SEP-28k dataset is no exception. Deep neural network (DNN) classifiers trained on highly class-imbalanced datasets are usually biased towards the majority class, which

TABLE I: STUTTERNET ARCHITECTURE, TC: TOTAL CONTEXT, TDNN: TIME DELAY NEURAL NETWORK LAYER, FC: FULLY CONNECTED LAYER, BILSTM: BIDIRECTINAL LONG SHORT-TERM MEMORY (2 LAYERS), LAYER CONTEXT OF [T-2, T+2] MEANS 5 FRAMES ARE TAKEN INTO CONSIDERATION TWO BEFORE THE CURRENT TIME STEP AND TWO AFTER THE CURRENT TIME STAMP.

Layer	Output Layer Size	Layer Context	TC
TDNN 1	64	$[t-2,\ t+2]$	5
TDNN 2	64	{t-2, t, t+2}	9
TDNN 3	64	{t-3, t, t+3}	15
TDNN 4	64	{t}	15
TDNN 5	3*64	{t}	15
BiLSTM	64 * 2	{t}	T
Statistical Pooling	$3*64*2{\times}1$	$[0,\ T)$	T
FC1	64	-	T
FC2	64	-	T
FC3	NumClasses	-	T

results in poor modeling of minority classes [32]. In addition to the above deficiencies, we found in our previous work *StutterNet* [29] that the context is very important in SD. Stuttering detection is a special type of speech characterization problem where each class (i.e., stuttering types) has a varying duration. For example, block lasts for a shorter duration than prolongations and repetitions. Therefore, a fixed length of context in the basic *StutterNet* framework might not be the optimized one for detecting all types of stuttering. A larger context increases the performance of prolongation and repetition types of disfluencies but decreases the recognition performance of fluent speech segments on the UCLASS dataset [29].

III. ADDRESSING DEFICIENCIES

In this section, we address the three above-mentioned issues.

A. Class Imbalance

In class imbalance learning, the distribution of samples across each class is not uniform. The DNN classifiers, trained on highly imbalanced datasets generally perform poorly for the minority class and favor the majority class [33]. In the worst case, where there is an extreme imbalance in the training set, the majority class dominates the learning, and samples among the minority class may go undetected, thus affecting performance [34].

The class-imbalance is one of the major problems in real-world applications, including multi-lingual speech recognition [35], and stuttering is no different as shown in Fig. 1. In fact, stuttering is extremely imbalanced across fluent and other speech disfluencies. The Fig. 1. shows that the interjections are the most common disfluency present in the SEP-28k dataset followed by repetitions, blocks, and prolongations and the overall distribution is approximately equivalent to the fluent distribution.

Collecting a balanced dataset is difficult and expensive for the stuttering detection task. Other datasets such as Kassel State of Fluency (KSoF) (not publicly accessible) [36], FluencyBank [28] also suffer from this issue.
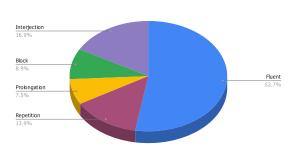
Fig. 1: Stuttering data distribution in SEP-28k dataset showing five classes consisting of single stuttering class.

Over the years, the class imbalance problem is one of the main concerns due to its prevalence, especially in the biomedical domain. Several methods have been proposed to tackle the class imbalance problem, which are mainly categorized into three groups: data-level, cost-level, and architecture-level [32]. Data-level approaches attempt to re-balance the class distribution by means of some re-sampling methods which include: under-sampling, over-sampling, or combined over-sampling and under-sampling [32], [33]. The architecture-level approaches attempt to modify the existing algorithm or develop a new one to tune and adapt it for imbalanced datasets [32], [33]. Cost-level approaches attempt to influence the loss/objective function by providing comparatively higher misclassification cost penalties to the minority class in order to force the model to learn about minority classes [32], [33]. In DNNs, addressing the class-imbalance problem by re-sampling may either get rid of sensitive speech samples that are extremely important in training when under-sampling. It can also add numerous quantities of duplicated speech samples under the over-sampling strategy, which eventually makes the training expensive and makes the DNN model likely to overfit [32]. Because of these limitations, we investigate cost-level and architecture-level approaches in this work.

For the cost-based approach, we modify the standard cross entropy loss by assigning weights to different classes [37]. We set the class weights inversely proportional to the number of samples. We define the weight for class i as $w_i = \frac{N}{C*N_i}$ where N is the number of training samples, C is the number of classes, N_i is the number of training samples for class i. Therefore, the weighted cross-entropy (WCE) over the train set can be defined as,

$$\mathcal{L}_{\mathrm{WCE}} = \frac{1}{\mathcal{B}} \sum_{b=1}^{\mathcal{B}} \frac{\sum_{i}^{M} w_i * \log(p_i)}{\sum_{i,\ i \in \mathcal{B}}^{M} w_i} \qquad (1)$$

where $\mathcal{B}$ is the number of batches, M is number of stuttered speech samples in a batch b_i, $p_i = \left(\frac{e^{c_i}}{\sum_{j=1}^{C} e^{c_j}} \right)$ is the predicted probability of class c_i of sample i.

For architecture-level, we propose a multi-branched approach similar to the work by C. Lea *et al.* [28] and M. Bader-El-Den *et al.* [38] to address the class imbalance issue in SD task. Inspired by the fact that the number of fluent class samples is almost equal to the total number of samples in disfluent classes, we simultaneously classify fluent vs disfluent classes in one output branch and subcategories of disfluent classes in another output branch. The multi-branched architecture with two output branches is shown in Fig. 3 (The figure is overall architecture of mulit-contextual *StutterNet* with two contexts, For single contextual MB *StutterNet*, only context = 5 is taken into consideration). This has one common encoder $\mathcal{E}$ (θ_{e}) section followed by two parallel branches referred as *FluentBranch* $\mathcal{F}$ (θ_{f}) and *DisfluentBranch* $\mathcal{D}$ (θ_{d}). The embeddings from the encoder are processed with both the branches parallelly, where the $\mathcal{F}$ is trained to distinguish between fluent and disfluent samples, and the $\mathcal{D}$ is trained to differentiate within the disfluent sub-categories.

The objective is to optimize the sum of *FluentBranch* loss $\mathcal{L}_{\mathrm{f}}$ and *DisfluentBranch* loss $\mathcal{L}_{\mathrm{d}}$. For simplicity, a simple sum of the two losses has been taken into consideration and it works well. Thus, the overall objective function is define as:

$$\mathcal{L}(\theta_{\mathrm{e}}, \theta_{\mathrm{f}}, \theta_{\mathrm{d}}) = \mathcal{L}_{\mathrm{f}}(\theta_{\mathrm{e}}, \theta_{\mathrm{f}}) + \mathcal{L}_{\mathrm{d}}(\theta_{\mathrm{e}}, \theta_{\mathrm{d}}). \qquad (2)$$

During the evaluation step, if the *FluentBranch* predicts the sample as fluent, then *FluentBranch* predictions are considered otherwise *DisfluentBranch* predictions are taken into consideration to reveal the stuttering category.

B. Data Augmentation

Deep learning has achieved rapid progress in the domain of speech processing tasks including speech recognition [17], speaker recognition [18], emotion recognition [20], speech disorder detection [29]. However, as a drawback, deep learning based models are hungry for the data and require a substantial amount of annotated data for training, and *StutterNet* is no exception. The model may require more stuttering data than other speech processing domains, as the presence of disfluencies in stuttering speech corpus is not frequent.

Data augmentation is a popular technique that increases the quantity and diversity of the existing annotated training data, improves robustness, and avoids the overfitting of DNNs. For normal speech recognition, data augmentation demonstrates to be an effective approach for dealing with data scarcity and enhancing the performance of various DNN acoustic methods [39]. Several data augmentation techniques have been investigated including pitch adjust [40], spectral distortion [41], tempo perturbation [42], speed perturbation, cross-domain adaptation [43], adding noise to clean speech [42], spectrogram deformation with frequency and time masking [39], mixspeech [40], etc.

On the contrary, so far, very limited attention has been given to data augmentation targeting the speech disorder domain.

In [44], speed and tempo perturbation based data augmentation were used to convert the normal speech to dysarthric speech impairment. In [45], a voice conversion adversarial training based framework was used to simulate dysarthric speech from healthy speech. In [46], normal speech samples (also called out-of-domain) were used as a data augmentation for dysarthric speech in the bottleneck feature extraction stage. In [47], the speaker-dependent parameter was computed, which was then used for augmentation of scarce dysarthric speech in tempo adjustment. In [48], several data augmentation techniques such as noise, time stretching, pitch shifting, time shift, masking, etc., were analysed in Dementia detection. There are some studies on data augmentation targeting text based stuttering detection [13], however, in the case of audio based stuttering/disfluency detection, this has not been studied and analysed deeply [49] .

In the stuttering domain, the employment of data augmentation is not straightforward, because most of the data augmentations like time stretch, speed perturbation, etc, alter the underlying structure of the stuttering speech sample completely. Our approach employs reverberation and additive noises because it reflects the real-world scenario and does not change significantly the underlying stuttering in the speech sample as shown in Fig. 2. Reverberation consists of convolving speech samples with room impulse responses. We utilize the simulated room impulse responses as described in [50]. For additive noises, we utilize the MUSAN dataset, comprised of 60 hours of speech from 12 languages, 42 hours of music from various genres, and 900 hours of noises [51].

To augment the original speech samples, we combine the training "clean" set with below mentioned augmented copies which results in 5 times increase in training samples.
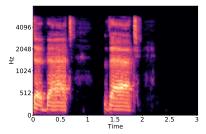
1) *music:* A single music sample file randomly chosen from MUSAN is added to the original clean stuttering speech sample (SNR: 5-15dB) (The music sample file is trimmed or repeated as required to match the duration of the clean stuttered speech sample).
2) *noise:* Throughout the stuttered speech, samples from MUSAN noises are added at 1 sec intervals (SNR: 0-15dB).
3) *babble:* Speech samples from randomly three to seven speakers are summed together, then added to the original clean stuttered speech sample (SNR: 13-20dB).
4) *reverb:* The "clean" train set is convolved with simulated room impulse responses.

All the data augmentation types shown in Fig. 2 were performed using Kaldi [52] tool.
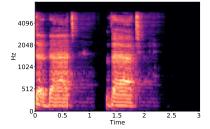
C. Multi-contextual StutterNet

The multi-contextual framework is based on the way humans perceive speech. In the cochlea, the input acoustic speech signal is partitioned into several frequency bands so that the information in each band can be filtered independently and thus processed in parallel in the human brain [53].
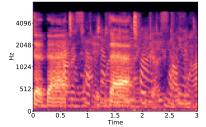
Multi-contextual has been studied for action recognition in videos [54], robust ASR [55]–[59], speech separation [60], where the input speech signal is processed in multiple
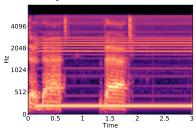

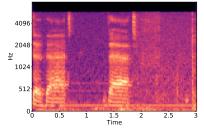(a) Repetition (clean speech)

(b) Repetition with reverberation

(c) Repetition with babble noise

(d) Repetition with white noise

(e) Repetition with music

Fig. 2: Repetition stuttering with utterance *"said that, that"* and the effect of various data augmentations (From 0 to 0.25 seconds, the speaker is saying *"said"*, then followed by two repetitions *"that"* from 0.25 to 0.6 and 1.4 to 1.75).

streams/contexts (multiple time or frequency resolutions), etc. K.J Han *et al.* [55] recently proposed a multi-stream[2] convolutional neural network for robust acoustic modeling. Chiba *et al.* [61] recently proposed multi-stream attention-based BiLSTM network for speech emotion recognition. Li *et al.* [62] extracted deep features by training multi-stream hierarchical DNN for acoustic event detection. Moreover Sheikh *et al.* [29] found that settings like context frame size optimized for one stuttering class are not good for other stuttering types.

Exploiting this fact, we investigate how the multi-contextual neural networks will impact classification performance in the speech disorder domain, and in particular stuttering identification. In our preliminary study, we found that the context window improves the identification performance of two types of disfluencies on the UCLASS dataset [29]. As the context frame size increases in the *StutterNet*, the performance detection of prolongation and repetition also increases, but decreases for fluent speech segments, and almost remains unaffected for a block type of stuttering. The prolongation and repetition last longer than other types of disfluencies. To address this issue, we exploit the variable contexts of MB *StutterNet* by training the model jointly on different contexts as shown in Fig. 3. The pseudo-code of multi-contextual (MC) *StutterNet* is provided in *Algorithm* 1.

IV. EXPERIMENTAL SETUP

We evaluate our proposed architecture thoroughly on the newly released SEP-28k stuttered dataset [28], and for cross copora, we use the FluencyBank and LibriStutter datasets.

A. Datasets

SEP-28k: The SEP-28k stuttering dataset was curated from a set of 385 podcasts. The original podcast recordings have varying lengths. From each podcast, 40 to 250 segments (each of length 3 seconds) are extracted which resulted in a total of 28,177 segments. The original SEP-28k dataset contains two types of labels: stuttering and non-stuttering. The stuttering labels include blocks, prolongations, repetitions, interjections, and fluent segments, whereas the nonstuttering labels include unintelligible, unsure, no speech, poor audio quality, and music which are not relevant to our study. In our case study, we use only the stuttering single labeled samples. Out of 28,177 clips, we use only 23573 segments, among which 3286 are repetitions, 1770 are prolongations, 2103 are blocks and 12419 are fluent segments, and 3995 are interjections. This resulted in a total of 19.65 hours of data which includes 2.74 hours of repetition, 1.48 hours of prolongation, 1.75 hours of block, 10.35 hours of fluent speech, and 3.34 hours of interjections. After labeling, each 3-sec sliced speech segment is downsampled to 16 kHz. We randomly select 80% of podcasts (without mixing podcasts) for training, 10% of podcasts for validation, and the remaining 10% of the podcast for evaluation in a 10-fold cross-validation scheme. The speaker information is missing from the SEP-28k dataset, so we divide the dataset based on podcast

---

Algorithm 1 Pseudo-code for MC *StutterNet*

Output: Predicted label set $\hat{y} \in (R, P, B, In, F)$
Input: Stutter dataset with $\mathcal{D} = (X_i, f_i, d_i)$, where each sample X_i is $\mathcal{R}^{20 \times T}$ MFCC input sequence

d: Stutter label	L: Total loss
f: Pseudo label for FluentBranch	b: Sample batch
K: Number of epochs	C: Context
CE: Cross entropy loss function	λ: Learning rate
p_f: Prediction of FluentBranch	$\theta = [\theta_e, \theta_f, \theta_d]$: Network parameters
p_d: Prediction of DisfluentBranch	θ_e : Encoder parameters
L_f: Loss of FluentBranch	θ_d : DisfluentBranch parameters
L_d: Loss of DisfluentBranch	θ_f : FluentBranch parameters

Ensure: MC *StutterNet* weight initialization
1: **for** i in K **do**
2:    **for** b in $\mathcal{D}$ **do**
3:      **Forward Pass:**
4:      b = ApplyKaldiAugmentation(b)
5:      out_5 = StutterNetBase(b, C = 5)
6:      out_9 = StutterNetBase(b, C = 9)
7:      $stat_5$ = StatisticalPooling(out_5)
8:      $stat_9$ = StatisticalPooling(out_9)
9:      $stat_{embed}$ = Concat($stat_5$, $stat_9$)
10:     p_f = FluentBranch($stat_{embed}$)
11:     p_d = DisfluentBranch($stat_{embed}$)
12:     $L_f = CE(p_f, f_b)$ // f_b: Pseudo label of a batch
13:     $L_d = CE(p_d, d_b)$ // d_b: Stutter label of a batch
14:     $L(\theta_e, \theta_f, \theta_d) = L_f(\theta_f) + L_d(\theta_d)$
15:     **Backward Pass:**
16:     $\nabla\theta_b$ = backward_propagation($L(\theta)$, $\hat{y}$, d)
17:     **Parameter Updates:**
18:     $\theta_b \leftarrow \theta_b - \lambda * \nabla\theta_b$
19:    **end for**
20:    Compute CE loss on validation set
21:    **if** $CE_{val} \leq CE_{val}^{prev}$ **then**
22:      **if** patience ≤ 7 **then**
23:       Continue training $\rightarrow$ Go to step 2
24:      **else**
25:       Stop training
26:      **end if**
27:    **end if**
28: **end for**

---

identities (assuming each podcast is having a unique speaker)[3].

FluencyBank: The actual FluencyBank AudioVisual dataset was created by Nan Bernstein Ratner (University of Maryland) and Brian MacWhinney (Carnegie Mellon University) for the study of fluency development. In our case study, we use the annotation done by the Apple similar to SEP-28k [1]. This stuttering dataset was curated from 33 podcasts with 23 males and 10 females, which resulted in a total of 4,144 segmented clips. Out of which, we only use 3355 samples (ignoring the non-stuttering and multiple samples), among which 542 are repetitions, 222 are prolongations, 254 are blocks and 1584 are fluent segments, and 753 are Interjections. This results in a total of 2.80 hours of data which includes 0.45 hours of repetition, 0.19 hours of prolongation, 0.21 hours of block, 0.63 hours of interjection samples, and 1.32 hours of fluent samples. We have considered those samples where at least two annotators agree with the same labeling of the segment.

Simulated LibriStutter: The LibriStutter (English) consists of 50 speakers (23 males and 27 females), is approximately

---

[2]multi-stream, multi-scale, multi-resolution are the different names of multi-context.

[3]The details about the train, validation, and test set splits about the SEP-28k dataset is not publicly available. We create a protocol by ensuring no overlap of podcasts between train, validation, and test sets. The details are available in `https://shakeel608.github.io/protocol.pdf`

During evaluation, **FluentBranch** predictions are considered only for fluent speech samples. If the **prediction** for speech sample in **FluentBranch** is "**stutter**", then **predictions** of **DisfluentBranch** are taken into **consideration** to reveal the **type** of stuttering class. Otherwise a speech utterance sample could be **predicted** as **fluent** in **FluentBranch** and **stutter** in **DisfluentBranch**.
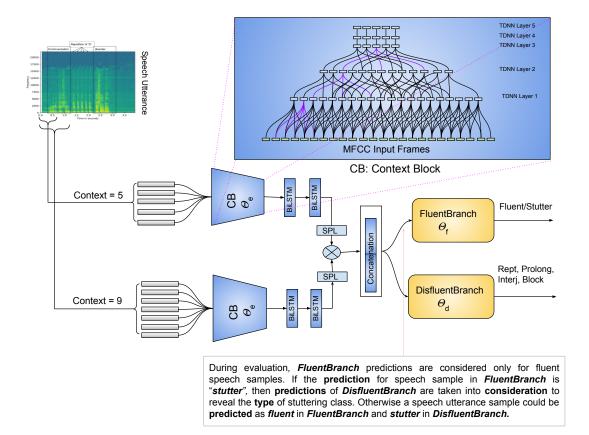
Fig. 3: A schematic diagram of Multi-contextual *StutterNet*, which is a multi-class classifier that exploits different variable contexts of (5, 9) in SD. The FluentBranch and DisfluentBranch are composed of 3 fully connected layers followed by a softmax layer for prediction of different stuttering classes, CB: Context Block, SPL: Statistical Pooling Layer. The context C (5, 9) here does not mean the TDNN layers, but rather it means the kernel-size which is number of frames taken into account when processing speech frames (C=5 means a context of 5 frames are taken at a time and C =9 means a context of 9 frames are taken at a time). These two contexts are jointly exploited in MC *StutterNet* as shown in the left hand side of the figure.

20 hours and includes synthetic stutters for repetitions, prolongations, and interjections [63]. Random stuttering was inserted within the four-second window of each speech signal. The original LibriStutter is having 6 classes including fluent, interjection[4], prolongation, sound, word and phrase repetitions. To make our experiments consistent with the SEP-28k dataset, we treat all repetitions as one class. After extracting samples based on the class label, we did not find any interjection samples in the dataset, so we only train with three classes including fluent, prolongation, and repetitions. For splitting of the dataset, please refer Table 3 in the response sheets.

B. Training Setup

We implement models using the PyTorch library. The acoustic input features used in this case study are 20-dimensional MFCC features, which are generated every 10 ms using a 20 ms window and extracted using the Librosa library [64]. For 3-dim pitch and phoneme features, we use Pykaldi [65] and Phonexia [66] tools respectively. For training, we use Adam optimizer and cross-entropy loss function with a learning rate of 10^{-2} and batch size of 128. All the results reported in this

paper are the average of the 10-fold validation technique and all the training experiments were stopped by early stopping criteria with patience of 7 on validation loss.

C. Evaluation metrics

To evaluate the model performance, we use the following metrics: macro F1-score and accuracy which are the standard and are widely used in the stuttered speech domain [27], [28], [31], [36], [67]. The macro F1-score ($\mathcal{F}_1$) (which combines the advantages of both precision and recall in a single metric unlike unweighted average recall which only takes recall into account) from equation (3) is often used in class imbalance scenarios with the intention to give equal importance to frequent and infrequent classes, and also is more robust towards the error type distribution [68].

$$\mathcal{F}_1 = 1/C \sum_k F1_k = 1/C \sum_k \frac{2.P_k R_k}{P_k + R_k} \quad (3)$$

where C is the number of classes and P_k, R_k, and $F1_k$ denotes the precision, recall, and F1-score with respect to class k.

---

[4]https://borealisdata.ca/dataset.xhtml?
persistentId=doi:10.5683/SP3/NKVOGQ

D. Experiments

This sections describes briefly the experiments carried out in this paper.

- We carry out experiments using *StutterNet* with the two state-of-the art SD baselines including ResNet+BiLSTM and ConvLSTM models in the same settings to have a fair comparison.
- We perform experiments using weighted loss and multi-branched training schemes for addressing class-imbalance problem in stuttering domain, and also exploit the advantage of both the schemes in freezing parts of network.
- We experiment with the data augmented training to evaluate its performance in stuttering detection.
- We experiment with MC *StutterNet* on top of data augmentation, and also analyse its performance in cross-corpora settings.

V. RESULTS

In this section, we discuss the results with class balanced training, data augmentation, and multi-contextual learning. We propose two modifications to the vanilla *StutterNet* to address the class imbalance issue and one of them involves the loss function on the SEP-28k dataset. We further present the results of cross-corpora experiments on the FluencyBank and LibriStutter datasets. Table II shows the results of state-of-the-art baselines (1st part are baselines) and impact of class balanced training. Table III depicts the results using data augmentation on top of class balanced training. Table IV shows the results using MC *StutterNet*.

Baselines: In addition to our single branch *StutterNet* baseline (BL5), we first implement the state-of-the-art ConvLSTM and ResNet+BiLSTM as our baseline models for comparison purposes in the same settings. Our baseline model *StutterNet* is performing well in almost all the disfluent classes except blocks as compared to the baseline model used in SEP-28k paper [28] ConvLSTM (MFCC features) (referred to as BL1), ConvLSTM (phoneme features) (referred to as BL2), and ConvLSTM (pitch+MFCC features) (referred to as BL3). In addition, we also use one more model i.e, ResNet+BiLSTM classifier (referred to as BL4) from Kourkounakis *et al.* [27]. Comparing our single branch baseline *StutterNet* to BL4, the model performs well only in interjection and prolongation classes as shown in Table II. For subsequent comparison, we select BL1 (best among BL1, BL2 and BL3) and BL4.

A. Class Imbalance

1) Weighted cross entropy: This section discusses the impact of applying weighted cross entropy to *StutterNet* (which we call *StutterNet*$_{\text{WCE}}$) in order to improve the detection performance of minority classes including repetitions, blocks, and prolongations. Figure 4. illustrates the class-wise training loss curves for normal and weighted cross-entropy loss functions. We can observe that for the standard cross entropy loss function, the majority class (i.e., fluents) exhibit higher loss values and it dominates the overall loss.

Therefore, during training with backpropagation, the number of updates for fluent class dominates the gradient values
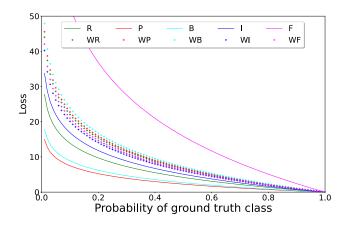

Fig. 4: Class-wise loss values for different probabilities of ground truth classes. Here block, fluent, repetition, prolongation, and interjection classes are correspondingly denoted by B, F, R, P, and I. The standard cross entropy (CE) is shown by solid curves and its weighted version is shown by dashed ones (WX represents the weighted loss of class X).

that in turn forces the model to focus mainly on correctly classifying/predicting the majority class. Thus, the minority classes including the blocks, prolongations, and repetitions are given less importance during training, which leads to their poor detection performance. Table II confirms that the detection performance of blocks, prolongations, and repetitions is very poor, as they are mostly predicted as fluent samples due to the class imbalance nature of the problem. The loss functions for the weighted cross entropy are shown by dashed curves in Fig. 4. The figure indicates that applying weights to standard cross entropy loss function by equation (1) forces the model to give balanced importance to each disfluency class while optimizing the parameters of the network during backpropagation. This, in turn, helps in boosting the gradient updates for minority classes during training, and thus increases their detection performance in baselines including BL4, and BL5 as shown in Table II. The *StutterNet*$_{\text{WCE}}$ gives a relative improvement of 33.49%, 83.50%, 1,185%, and 5.98% over BL5 and 58.69%, 462%, 500%, and 7.84% over BL1 for detecting repetitions, prolongations, blocks, and interjections, respectively. Table II also demonstrates the suitability of WCE with competitive ResNet+BiLSTM method and results in 54%, 57.35%, 517%, and 10.23% relative improvement in repetitions, prolongations, blocks, and prolongations over BL4.
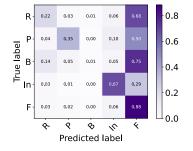
2) Multi-branch training: Moreover, we address the class imbalance problem via a multi-branch network (referred to as *StutterNet*$_{\text{MB}}$). This has two output branches: *FluentBranch* and *DisfluentBranch* as shown in Fig. 3. (For *StutterNet*$_{\text{MB}}$, only a single context of five is taken into consideration) This method improves the detection performance of repetitions, prolongations, blocks by a relative margin of 29.92%, 0.65%, 144% respectively over BL5, and 54.45%, 208%, 13.72%,
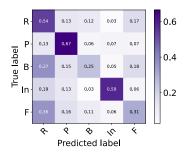
TABLE II: RESULTS WITH BASELINES (BL) AND USING CLASS IMBALANCE LEARNING (CLEAN TRAINING) (B: BLOCK, F: FLUENT, R: REPETITION, P: PROLONGATION, IN: INTERJECTION, TA: TOTAL ACCURACY, $\mathcal{F}_1$: MACRO F1- SCORE), F_{MFCC}: MFCC INPUT FEATURES, F_{F0}: 3-DIM (PITCH, PITCH-DELTA, VOICING) FEATURES, F_{phone}: PHONEME FEATURES, MB: MULTI BRANCH, WCE: WEIGHTED CROSS ENTROPY, $\mathcal{M}_{enc}^{frz}$:FREEZING ENCODER, $\mathcal{M}_{enc,disf}^{frz}$: FREEZING ENCODER AND DISFLUENTBRANCH, $\mathcal{M}_{enc,fluent}^{frz}$: FREEZING ENCODER AND FLUENTBRANCH.

Method	Accuracy						
	R	P	B	In	F	TA	$\mathcal{F}_1(\%)$
	Baselines						
ConvLSTM + F_{MFCC} (BL1) [28]	22.83	10.61	06.34	56.74	72.35	52.68	34.00
ConvLSTM + F_{phone} (BL2) [28]	10.18	01.06	00.35	43.88	74.48	48.43	24.00
ConvLSTM + $F_{F0+MFCC}$ (BL3) [28]	19.28	09.55	08.51	51.78	66.60	48.47	30.80
ResNet+BiLSTM (BL4) [27]	18.76	41.24	5.47	57.18	88.19	62.36	43.12
StutterNet (BL5) [29]	27.14	32.55	2.96	57.74	87.60	62.57	42.84
	Class Imbalance						
ResNet+BiLSTM + WCE [27]	28.90	64.89	33.79	63.03	46.90	47.42	41.00
MB ResNet+BiLSTM [27]	34.79	30.19	5.92	49.26	75.47	55.62	39.20
StutterNet + WCE (*StutterNet*$_{WCE}$)	36.23	59.73	38.05	61.19	41.59	45.26	41.02
MB StutterNet (*StutterNet*$_{MB}$)	35.26	32.76	7.21	56.04	77.27	58.56	42.26
$\mathcal{M}_{enc}^{frz}$	39.82	37.91	10.45	60.57	73.49	58.58	44.42
$\mathcal{M}_{enc,disf}^{frz}$	29.25	45.85	18.11	56.88	74.49	58.18	44.80
$\mathcal{M}_{enc,fluent}^{frz}$	31.15	27.62	05.01	57.64	73.64	55.83	38.60

TABLE III: RESULTS USING DATA AUGMENTATION (B: BLOCK, F: FLUENT, R: REPETITION, P: PROLONGATION, IN: INTERJECTION, TA: TOTAL ACCURACY, $\mathcal{F}_1$: MACRO F1- SCORE), A4: BABBLE + REVERBERATION + MUSIC + NOISE AUGMENTATION, MB: MULTI BRANCH, WCE: WEIGHTED CROSS ENTROPY, $\mathcal{M}_{enc}^{frz}$:FREEZING ENCODER, $\mathcal{M}_{enc,disf}^{frz}$: FREEZING ENCODER AND DISFLUENTBRANCH, $\mathcal{M}_{enc,fluent}^{frz}$: FREEZING ENCODER AND FLUENTBRANCH.

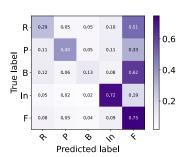| Method	Accuracy						
	R	P	B	In	F	TA	$\mathcal{F}_1(\%)$
StutterNet + A4	28.67	32.53	3.69	64.24	88.70	64.69	45.30
StutterNet + WCE (*StutterNet*$_{WCE}$) + A4	43.06	61.43	38.18	65.33	43.73	48.30	44.34
MB ResNet+BiLSTM + A4	33.27	32.85	0.99	65.95	70.92	55.75	39.44
MB StutterNet (*StutterNet*$_{MB}$) + A4	34.05	33.93	4.32	68.74	78.88	61.35	44.03
$\mathcal{M}_{enc}^{frz}$ + A4	28.87	30.94	4.26	64.12	85.29	62.85	44.06
$\mathcal{M}_{enc,disf}^{frz}$ + A4	27.63	36.33	11.52	62.31	83.05	62.14	45.76
$\mathcal{M}_{enc,fluent}^{frz}$ + A4	31.19	25.14	04.48	61.30	77.56	58.32	39.80

(a) *StutterNet* (BL)

(b) *StutterNet* (WCE) (*StutterNet*$_{WCE}$)

(c) MB *StutterNet* (*StutterNet*$_{MB}$)

Fig. 5: Accuracy confusion matrices showing the confusion of fluent speech with repetitions and blocks. F: Fluent, R: Repetition, B: Block, P: Prolongation, In: Interjection, BL: Baseline, WCE: Weighted cross-entropy, MB: Multi-branch.

6.80% in repetitions, prolongations, blocks, and fluents respectively over BL1, and 88%, 31.81% in repetitions, blocks respectively over BL4, however, the BL4 is performing better in prolongations and interjection classes. We also applied multi-branch training in ResNet+BiLSTM which results in a relative improvement of 85.45% in repetition class only as compared to the baseline BL4.

In applying class balanced training, we found that there is drop in macro $\mathcal{F}_1$ score. Using *StutterNet*WCE and WCE based ResNet+BiLSTM, the macro $\mathcal{F}_1$ score drops from 42.84% and 43.12% to 41.02% in *StutterNet* and 41.02% in ResNet+BiLSTM respectively. By employing multi-branch training, the macro $\mathcal{F}_1$ score drops very slightly from 42.84% to 42.26% in *StutterNet* but it drops remarkably from 43.12%

to 39.20% in ResNet+BiLSTM.

3) Analysis of confusion matrix: Using the WCE training scheme, the detection performance of minority classes improved remarkably at the cost of fluent accuracy. From the Fig. 5 (a), we analyze that most of the repetitions and blocks are being classified as fluent speech. Initially, we hypothesize that it is most likely because of the imbalanced nature of the problem. Despite addressing the class imbalance problem in the stuttering domain using WCE and multi-branched training, we found that the block and repetition type of stuttering disfluencies are the ones that are still getting confused with the fluent class as depicted in Fig. 5 (b) and Fig. 5 (c). This makes intuitive sense because the blocks are closely related with fluent segments having just different initial silence or gasp followed by fluent utterances like fluent speech. The repetitions, on the other hand, contain some word or phrasal repetitions, which are actually fluent utterances, if we carefully analyze their individual parts. Consider an utterance *he he is a boy*. The word he is being repeated twice but is a fluent part if two *he's* can be analyzed on an individual basis.

4) Exploiting advantage of WCE and MB StutterNet*:* Since *StutterNet*$_{\text{WCE}}$ and *StutterNet*$_{\text{MB}}$ address the class-imbalance issue differently, we combine them to exploit both of their advantages. We first pre-train the *StutterNet*$_{\text{WCE}}$ and use it as a *DisfluentBranch* in our multi-branched *StutterNet*. After the pre-training step, we freeze parameters in two ways. First, we freeze the parameters of the contextual encoder only and fine-tune only the two output branches. We label this training scheme as $\mathcal{M}_{\text{enc}}^{\text{frz}}$. By exploiting this method, we achieve an overall detection improvement of 5.11% in $\mathcal{F}_1$ over *StutterNet*$_{\text{MB}}$. Second, we freeze the base encoder and *StutterNet*$_{\text{WCE}}$ (*DisfluentBranch*) and append with one more FluentBranch (to distinguish between fluents and stutter samples). We refer this as $\mathcal{M}_{\text{enc,disf}}^{\text{frz}}$ and it results in an overall improvement of 6.01% in $\mathcal{F}_1$ over *StutterNet*$_{\text{MB}}$.

We also experiment by first training the model using weighted cross entropy in *FluentBranch*, and then by fine-tuning the *DisfluentBranch* by freezing the parameters of the encoder and *FlunetBranch*. We refer to it as $\mathcal{M}_{\text{enc,fluent}}^{\text{frz}}$ and the results for this configuration are shown in Table II and III. However, this training scheme degrades performance in almost all the disfluent classes in comparison to $\mathcal{M}_{\text{enc}}^{\text{frz}}$ (freezing encoder only) and $\mathcal{M}_{\text{enc,disf}}^{\text{frz}}$ (freezing encoder and *DisfluentBranch*). This is possibly due to the reason that the *base encoder* is trained only to distinguish between fluent and disfluent classes via *FluentBranch*. Then freezing its parameters in a fine-tuning step further inhibits it more in learning sub-classes (repetitions, prolongations, interjections, and blocks) of the disfluent category which makes their overall detection performance lower.

B. Data augmentation

The main experimental results obtained with various data augmentation techniques are shown in Table III, where we compare the detection performance obtained with data augmented training to the baseline clean dataset. We first separately train the *StutterNet*$_{\text{MB}}$ on different data augmentation
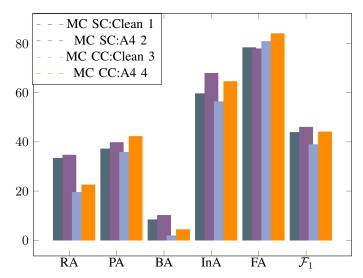

Fig. 6: Impact of data augmentation (A4) in cross corpora FluencyBank dataset with MC *StutterNet* (R: Repetition, P: Prolongation, B: Block, In: Interjection, F: Fluent, XA: X: Disfluency Class and A: Accuracy, MC: Multi contextual *StutterNet*, SC: Same corpora, CC: Cross corpora, A4: Augmentation). The bar plot clearly shows that the model MC *StutterNet* trained on clean SEP-28k dataset fails to generalize on FluencyBank cross corpora data. Applying data augmentation improves the stuttering detection in cross domain corpora as shown orange bars (4th column in each disfluency).

techniques which are described in Section III. We found that the training MB ResNet+BiLSTM and *StutterNet*$_{\text{MB}}$ with Kaldi augmentation increases the overall $\mathcal{F}_1$ performance. From Table III, it can be seen that the data augmentation does help in improving the $\mathcal{F}_1$ in almost all the cases. Applying data augmentation with a single branched and *StutterNet*$_{\text{WCE}}$, there is a relative improvement of 5.74% and 3.50% in $\mathcal{F}_1$ respectively, over the clean versions of the training. When data augmentation is applied to MB training, there is a relative improvement of 4.17% and 0.61% in $\mathcal{F}_1$ using *StutterNet*$_{\text{MB}}$ and ResNet+BiLSTM over clean training, respectively.

Moreover, applying Kaldi data augmentation on top of the $\mathcal{M}_{\text{enc}}^{\text{frz}}$ and $\mathcal{M}_{\text{enc,disf}}^{\text{frz}}$ training scheme, there is a relative improvement of 7.28%, and 6.81% in overall accuracy respectively. In addition to Kaldi augmentation, we also apply pitch scaling and bandpass filter as data augmentation, however, we did not achieve much improvement in SD.

C. Multi-contextual StutterNet

Different contexts show different optimized class accuracies. In order to exploit these different variable contexts and to improve the detection performance further, we propose a multi-contextual (MC) *StutterNet* for SD as shown in Fig. 3. We jointly train and optimize the MC *StutterNet* on the clean and augmented data using variable contexts of 5 and 9. The embeddings extracted from each context as depicted by $\mathcal{CB}$ block are passed to a two-layered BiLSTM unit, and then concatenated after applying statistical pooling layer (SPL), resulting in a $1 \times 2 \times (2 \times N)$-dimensional feature vector (where

TABLE IV: RESULTS USING MC *StutterNet* WITH CLEAN AND DATA AUGMENTATION. THE MC *StutterNet* ALSO CONTAINS MB TRAINING. (B: BLOCK, F: FLUENT, R: REPETITION, P: PROLONGATION, IN: INTERJECTION, TA: TOTAL ACCURACY, BB: BABBLE, RV:REVERBERATION, MU:MUSIC, NO:NOISE, A4: BB + MU + NO + RV AUGMENTATION)

Method	Accuracy						
	R	P	B	In	F	TA	$\mathcal{F}_1$(%)
MC *StutterNet* (Clean)	33.36	37.15	08.34	59.63	78.34	59.77	43.86
MC *StutterNet* + Bb	32.29	37.11	09.85	66.50	74.54	58.71	44.40
MC *StutterNet* + Mu	32.08	36.57	07.98	65.68	77.67	59.96	44.00
MC *StutterNet* + No	28.54	32.39	04.75	64.20	80.42	59.99	42.80
MC *StutterNet* + Rv	31.98	37.02	05.83	67.63	77.41	60.08	44.20
MC *StutterNet* + A4	34.65	39.75	10.12	67.91	77.89	61.72	46.00

N is layer size), which is then fed parallelly to two different branches including *FluentBranch* and *DisfluentBranch* for class predictions. This results in a relative improvement of 13.40%, 15.67%, 6.41%, and 1.38% in prolongation, block, interjection, and fluent classes over *StutterNet*$_{\text{MB}}$ (clean), thus an improvement of 3.79% in macro $\mathcal{F}_1$ score, however, employing multi-contextual training, we see a drop from 35.26% to 33.36% in repetition accuracy over the *StutterNet*$_{\text{MB}}$. In comparison to baseline BL5 (vanilla single branch *StutterNet*), there is a relative improvement of 22.92%, 14.13%, 181.81%, 3.27% in repetition, prolongation, block, and interjection classes respectively. Thus a relative improvement of 2.38% in macro $\mathcal{F}_1$ score. The MC *StutterNet* also performs better in macro $\mathcal{F}_1$ score in comparison to state-of-the-art baselines BL1 and BL4. Applying data augmentation on top of the MC *StutterNet*, we found that except noise augmentation, all other data augmentation types help in improving the macro $\mathcal{F}_1$ score in comparison to MC *StutterNet* (clean) The noise augmented samples help in improving the detection accuracy of fluent class. For interjections, we found that all the data augmentation helps, and for blocks, only babble augmentation helps in improving their detection accuracy. We also found applying all four data augmentation techniques in MC *StutterNet* results in an accuracy improvement in prolongation and repetition classes, however, with individual data augmentation, a drop in their accuracies can be observed in Table IV. By applying all four data augmentation in MC *StutterNet* training, there is an overall improvement of 1.76%, 17.15%, and 134% in repetitions, prolongations, and blocks respectively as shown in the Table IV, which is a 4.48% relative gain in macro $\mathcal{F}_1$ score over the augmented *StutterNet*$_{\text{MB}}$ training.

D. Summary of proposed methods

This work advances the basic *StutterNet* by addressing its limitations with three modifications. In Table V, we present a summary of the results demonstrating systematic improvements. We observe that all the proposed modifications help to gradually improve the performance and we achieve 7.37% overall relative improvement in terms of macro $\mathcal{F}_1$ score.

E. Cross corpora evaluation

Table VI shows the results on cross corpora datasets including FluencyBank and simulated LibriStutter. By optimizing on SEP-28k dataset in terms of data augmentation and multi-contextual, we aim to evaluate our proposed methodology MC *StutterNet* on a cross corpora scenario. We train *StutterNet*

TABLE V: SUMMARY OF RESULTS OF PROPOSED METHODS (A4: ALL FOUR AUGMENTATION (BB + MU + NO + RV)

Method	$\mathcal{F}_1$ (%)
StutterNet (BL5) (Clean) [29]	42.84
Class Imbalance	
StutterNet$_{\text{WCE}}$ (Clean)	41.02
StutterNet$_{\text{MB}}$ (Clean)	42.26
$\mathcal{M}_{\text{enc}}^{\text{frz}}$ (Clean)	44.42
$\mathcal{M}_{\text{enc,disf}}^{\text{frz}}$ (Clean)	44.80
Data Augmentation	
StutterNet + A4	45.30
StutterNet$_{\text{WCE}}$ + A4	44.34
StutterNet$_{\text{MB}}$ + A4	44.03
$\mathcal{M}_{\text{enc}}^{\text{frz}}$ + A4	44.06
$\mathcal{M}_{\text{enc,disf}}^{\text{frz}}$ + A4	45.76
Multi-Contextual	
MC StutterNet (Clean)	43.86
MC StutterNet + A4	**46.00**

on SEP-28k dataset and evaluate it on *FluencyBank* dataset which comprises samples from 33 podcasts. We found that the model trained on one corpus (SEP-28k) fails to generalize and performs poorly on cross-domain corpora. As can be seen from the Table VI and Fig. 6, the $\mathcal{F}_1$ detection performance decreases remarkably from 43.86% to 38.92% employing clean training. We hypothesize that the performance drop is due to the domain mismatch due to the difference in speaker accent and recording environment between the SEP-28k and FluencyBank datasets. The repetitions and block classes show more degradation in their performance in cross corpora scenarios. Furthermore, we apply data augmentation in cross corpora evaluation, and we found that it boosts the detection performance of all the classes, which results in an improvement of 13.23% in $\mathcal{F}_1$.

Experimental evaluation of MC *StutterNet* on simulated LibriStutter dataset [63] results in a macro $\mathcal{F}_1$ score of $\approx 91\%$ (shown in Table VI). However, the performance considerably drops when evaluated in cross-corpora setting in comparison to real stuttering dataset FluencyBank. The MC *StutterNet* trained on SEP-28k dataset shows extremely poor performance when tested on simulated LibriStutter dataset. Applying data augmented training, we see there is minimal improvement of 1% and 2% in repetition and prolongation, respectively. The LibriStutter simulated dataset does not reflect the actual nature and characteristics of stuttered speech. The results also confirms that a model trained on simulated stuttered datasets should not be used in a real clinical condition.

TABLE VI: RESULTS ON CROSS CORPORA FLUENCYBANK AND SIMULATED LIBRISTUTTER DATASETS. THE FIRST ROW IS FROM TABLE IV WHICH SHOWS THE RESULTS ON THE SAME CORPORA SEP-28K DATASET. THE LAST TWO ROWS IN THE FIRST HALF OF THE TABLE SHOW THE RESULTS, WHERE THE MODEL IS TRAINED ON SEP-28k AND TESTED ON FLUENCYBANK IN CROSS CORPORA SETTING. (B: BLOCK, F: FLUENT, R: REPETITION, P: PROLONGATION, IN: INTERJECTION), BB: BABBLE, RV: REVERBERATION, MU: MUSIC, NO: NOISE, A4: ALL FOUR BB + RV + MU + NO AUGMENTATION. THE MULTI-BRANCHED TRAINING IS ALSO CONSIDERED IN MC *StutterNet*.

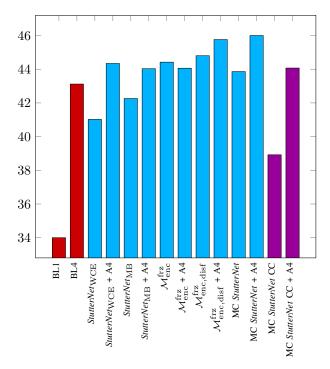| Method	TrainSet	TestSet	Accuracy						$\mathcal{F}_1(\%)$
			R	P	B	In	F	TA	
MC *StutterNet*	SEP-28k	SEP-28k	33.36	37.15	08.34	59.63	78.34	59.77	43.86
MC *StutterNet* (Clean)	SEP-28k	FluencyBank	19.48	35.80	01.83	56.36	80.85	56.48	38.92
MC *StutterNet* + A4	SEP-28k	FluencyBank	22.54	42.22	4.36	64.56	84.04	60.92	44.07
MC *StutterNet*	LibriStutter	LibriStutter	93.36	76.26	NA	NA	98.19	96.11	91.00
MC *StutterNet*(clean)	SEP-28k	LibriStutter	00.65	00.48	NA	NA	99.95	77.25	30.00
MC *StutterNet* + A4	SEP-28k	LibriStutter	01.11	02.39	NA	NA	97.35	75.43	31.00
MC *StutterNet*	LibriStutter	SEP-28k	24.24	55.11	NA	NA	60.60	53.21	41.00

Fig. 7: Macro $\mathcal{F}_1$ score summary of proposed and baseline models (BL1 and BL4). The red, light blue and purple bars indicate the baseline, same-corpora setting, and cross-corpora settings. (CC: Cross Corpora with training on SEP-28k and evaluated on FluencyBank dataset, A4: All four data augmentation. The second last bar shows cross-corpora performance on FluencyBank when trained using a clean SEP-28k dataset and the last bar shows the same with data augmentation).

proposed *MC StutterNet* is a multi-class classifier that exploits different variable contexts trained jointly on different contexts (5, 9) using CE. More importantly, the experiments using data augmentation over the FluencyBank dataset revealed that our methodology generalizes better in the cross corpora domain. For class imbalance, we have used only a simple weighting scheme in the cross-entropy loss function, which results in an accuracy trade-off between majority and minority classes.

In general, the data augmentation helps in stuttering domain, however, the use of data augmentation in the stuttering domain is not straightforward, thus, is limited because most data augmentations, such as time stretch, speed perturbation, and so on, completely alter the underlying structure of the stuttering speech sample. In order for data augmentation to be more effective, a domain stuttering specific data augmentation is required to be developed. In addition, the stuttering detection domain has not matured enough, so a single metric like other speech domains which reflects the overall performance of a model is yet to be developed. In addition to accuracy metric, we have also used a macro F1-score ($\mathcal{F}_1$) which gives a good indication for the better evaluation of proposed methods. Moreover, we use joint training over multi contexts in this work, and it is possible that one context can dominate the training. A visualisation summary of macro $\mathcal{F}_1$ score of models in stuttering detection is shown in Fig. 7.

The proposed methodology show promising results and it can detect if the stuttering is present in the speech sample or not, however, it cannot predict where exactly the stuttering occurs in the speech frames. For future study, we would like to explore combining the other different types of neural networks in SD to predict the frames where exactly the stuttering occurs. In addition to varying context, the investigation of varying depth and different number of convolutional kernels is also an interesting topic to study in SD. Moreover, the temporal information captured by recurrent neural networks can also be investigated in a multi-stream fashion for the identification of disfluent speech frames.

The performance comparison of the proposed systems with two state-of-the-art systems demonstrate that even though we achieve a noticeable advancements, the automated stuttering detection requires further research for developing a clinically usable system. The stuttering detection is fundamen-

VI. CONCLUSION

This paper addresses the problem of class imbalance in the stuttering domain. We address the class imbalance problem via two strategies including weighted cross entropy and a multi-branch training scheme. The weighted cross entropy loss function forces the *StutterNet* classifier to give more attention to minority classes. We also investigate the effectiveness of data augmentation in the SD domain. For data augmentation, we employ reverberations and additive noises from the MUSAN dataset [51]. Additionally, we propose a MC *StutterNet*, time delay based neural network for SD. The

tally a challenging task due to inter-person variations, language/accent/dialect variability and other speaking variations. The scopes of the current work are limited to addressing basic problems related to stuttering detection. This work can be extended with speaker-adaptive training, domain adaptation to further improve the stuttering detection problem.

Regarding cognitive aspects of stuttering are well modeled or not as the explainability analysis of the used deep models along with auxiliary data (e.g., functional magnetic resonance imaging or fMRI data) is yet to be explored. However, we think the base architecture (i.e., MC *StutterNet*) developed in this work could still be useful for such explainability analysis. The stuttering detection can be improved with multimodality by integrating with visual cues related to head nodding, lip tremors, unusual lip shapes, quick eye blinks, facial expressions, etc. We think that the same base acoustic model can be used in a fusion framework.

We have found that the blocks are the most difficult to detect. It would be interesting to analyze the disfluencies of the speakers which are hardest to identify by doing more ablation analysis. The future work includes exploring self-supervised models that exploit unlabelled audio data.

REFERENCES

[1] B. Guitar, *Stuttering: An Integrated Approach to its Nature and Treatment*. Lippincott Williams & Wilkins, 2013.

[2] J. R. Duffy, *Motor Speech Disorders-E-Book: Substrates, Differential Diagnosis, and Management*. Elsevier Health Sciences, 2013.

[3] N. B. Ratner and B. MacWhinney, "Fluency bank: A new resource for fluency research and practice," *Journal of Fluency Disorders*, vol. 56, p. 69, 2018.

[4] D. Ward, *Stuttering and Cluttering: Frameworks for Understanding and Treatment*. Psychology Press, 2008.

[5] T. D. Kehoe and W. Contributors, *Speech Language Pathology-Stuttering*. Kiambo Ridge, 2006.

[6] I. Hurjui, S. Pete, and I. Bostan, "Spatial distribution and the prevalence of speech disorders in the provinces of Iran," *Journal of Medicine and Life*, vol. 9, no. 1, p. 56, 2016.

[7] A. Smith and C. Weber, "How stuttering develops: The multifactorial dynamic pathways theory," *Journal of Speech, Language, and Hearing Research*, vol. 60, no. 9, pp. 2483–2505, 2017.

[8] E. Nöth *et al.*, "Automatic stuttering recognition using hidden Markov models," in *Proc. ICSLP*, 2000.

[9] V. Mitra *et al.*, "Analysis and tuning of a voice assistant system for dysfluent speech," in *Proc. Interspeech 2021*, 2021, pp. 4848–4852.

[10] L. Verde, G. De Pietro, and G. Sannino, "Voice disorder identification by using machine learning techniques," *IEEE Access*, vol. 6, pp. 16 246–16 255, 2018.

[11] N. Narendra and P. Alku, "Dysarthric speech classification from coded telephone speech using glottal features," *Speech Communication*, vol. 110, pp. 47–55, 2019.

[12] C. Quan, K. Ren, and Z. Luo, "A deep learning based method for parkinson's disease detection using dynamic features of speech," *IEEE Access*, vol. 9, pp. 10 239–10 252, 2021.

[13] S. Alharbi *et al.*, "A lightly supervised approach to detect stuttering in children's speech," in *Proc. Interspeech 2018*, 2018, pp. 3433–3437.

[14] ——, "Sequence labeling to detect stuttering events in read speech," *Computer Speech & Language*, vol. 62, p. 101052, 2020.

[15] V. Zayats, M. Ostendorf, and H. Hajishirzi, "Disfluency detection using a bidirectional LSTM," in *Proc. Interspeech 2016*, 2016, pp. 2523–2527.

[16] Q. Chen, M. Chen, B. Li, and W. Wang, "Controllable time-delay transformer for real-time punctuation prediction and disfluency detection," in *Proc. ICASSP 2020*, 2020, pp. 8069–8073.

[17] A. B. Nassif and *et al.*, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019.

[18] S. Latif and *at al.*, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," *arXiv preprint arXiv:2001.00378*, 2020.

[19] Y. Ning *et al.*, "A review of deep learning based speech synthesis," *Applied Sciences*, vol. 9, no. 19, p. 4050, 2019.

[20] M. B. Akçay and K. Oğuz, "Speech emotion recognition: Emotional models, databases, features, preprocessing methods, supporting modalities, and classifiers," *Speech Communication*, vol. 116, pp. 56–76, 2020.

[21] B. Sisman *et al.*, "An overview of voice conversion and its challenges: From statistical modeling to deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

[22] J. S. Almeida *et al.*, "Detecting Parkinson's disease with sustained phonation and speech signals using machine learning techniques," *Pattern Recognition Letters*, vol. 125, pp. 55–62, 2019.

[23] I. Kodrasi, "Temporal envelope and fine structure cues for dysarthric speech detection using CNNs," *IEEE Signal Processing Letters*, vol. 28, pp. 1853–1857, 2021.

[24] P. Howell and S. Sackin, "Automatic recognition of repetitions and prolongations in stuttered speech," in *Proc. of the first World Congress on Fluency Disorders*, vol. 2. University Press Nijmegen Nijmegen, The Netherlands, 1995, pp. 372–374.

[25] K. Ravikumar, B. Reddy, R. Rajagopal, and H. Nagaraj, "Automatic detection of syllable repetition in read speech for objective assessment of stuttered disfluencies," in *Proc. of World Academy of Science, Engineering and Technology*, vol. 36. Citeseer, 2008, pp. 270–273.

[26] B. Villegas *et al.*, "A novel stuttering disfluency classification system based on respiratory biosignals," in *Proc. IEEE EMBC 2019*. IEEE, 2019, pp. 4660–4663.

[27] T. Kourkounakis, A. Hajavi, and A. Etemad, "Detecting multiple speech disfluencies using a deep residual network with bidirectional long short-term memory," in *Proc. ICASSP 2020*, 2020, pp. 6089–6093.

[28] C. Lea *et al.*, "SEP-28k: A dataset for stuttering event detection from podcasts with people who stutter," in *Proc. ICASSP 2021*, 2021.

[29] S. A. Sheikh, M. Sahidullah, F. Hirsch, and S. Ouni, "StutterNet: Stuttering detection using time delay neural network," in *Proc. EUSIPCO 2021*, 2021, pp. 426–430.

[30] M. Jouaiti and K. Dautenhahn, "Dysfluency classification in stuttered speech using deep learning for real-time applications," in *Proc. ICASSP 2022*, 2022, pp. 6482–6486.

[31] S. A. Sheikh, M. Sahidullah, F. Hirsch, and S. Ouni, "Machine learning for stuttering identification: Review, challenges and future directions," *Neurocomputing*, vol. 514, pp. 385–402, 2022.

[32] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Springer, 2018, vol. 10.

[33] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[34] G. M. Weiss, "Foundations of Imbalanced Learning," *Imbalanced Learning: Foundations, Algorithms, and Applications*, pp. 13–41, 2013.

[35] G. I. Winata, G. Wang, C. Xiong, and S. Hoi, "Adapt-and-adjust: Overcoming the long-tail problem of multilingual speech recognition," *arXiv preprint arXiv:2012.01687*, 2020.

[36] B. W. Schuller *et al.*, "The ACM multimedia 2022 computational paralinguistics challenge: vocalisations, stuttering, activity, & mosquitos," in *Proc. ACM Multimedia 2022*, 2022.

[37] Y. Cui *et al.*, "Class-balanced loss based on effective number of samples," in *Proc. IEEE CVPR*, 2019, pp. 9268–9277.

[38] M. Bader-El-Den, E. Teitei, and M. Adda, "Hierarchical classification for dealing with the class imbalance problem," in *Proc. 2016 IJCNN*, 2016, pp. 3584–3591.

[39] D. S. Park *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.

[40] L. Meng, J. Xu, X. Tan, J. Wang, T. Qin, and B. Xu, "Mixspeech: Data augmentation for low-resource automatic speech recognition," in *Proc. ICASSP 2021*, 2021, pp. 7008–7012.

[41] N. Kanda, R. Takeda, and Y. Obuchi, "Elastic spectral distortion for low resource speech recognition with deep neural networks," in *Proc. IEEE ASRU 2013*, 2013, pp. 309–314.

[42] T. Ko *et al.*, "Audio augmentation for speech recognition," in *Proc. Interspeech 2015*, 2015.

[43] P. J. Bell *et al.*, "Transcription of multi-genre media archives using out-of-domain data," in *Proc. IEEE SLT 2012*, 2012, pp. 324–329.

[44] B. Vachhani, C. Bhat, and S. K. Kopparapu, "Data augmentation using healthy speech for dysarthric speech recognition," in *Proc. Interspeech 2018*, 2018, pp. 471–475.

[45] Y. Jiao *et al.*, "Simulating dysarthric speech for training data augmentation in clinical speech applications," in *Proc. ICASSP 2018*, 2018, pp. 6009–6013.

[46] H. Christensen *et al.*, "Combining in-domain and out-of-domain speech data for automatic recognition of disordered speech." in *Proc. Interspeech 2013*, 2013, pp. 3642–3645.

[47] F. Xiong, J. Barker, and H. Christensen, "Phonetic analysis of dysarthric speech tempo and applications to robust personalised dysarthric speech recognition," in *Proc. ICASSP 2019*, 2019, pp. 5836–5840.

[48] D. Woszczyk *et al.*, "Data augmentation for Dementia detection in spoken language." in *Proc. Interspeech 2022*, 2022, pp. 2858–2862.

[49] T. Grósz *et al.*, "Wav2vec2-based paralinguistic systems to recognise vocalised emotions and stuttering," in *Proc. ACM Multimedia*, 2022, p. 7026–7029.

[50] T. Ko *et al.*, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP 2017*, 2017, pp. 5220–5224.

[51] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[52] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE ASRU 2021*, 2011.

[53] J. B. Allen, "How do humans process and recognize speech?" in *Modern methods of speech processing*. Springer, 1995, pp. 251–275.

[54] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. NIPS 2014*. Cambridge, MA, USA: MIT Press, 2014.

[55] K. J. Han *et al.*, "Multistream CNN for robust acoustic modeling," in *Proc. ICASSP 2021*, 2021, pp. 6873–6877.

[56] H. Bourlard and S. Dupont, "A new ASR approach based on independent processing and recombination of partial frequency bands," in *Proc. ICSLP*, 1996.

[57] H. Bourlard *et al.*, "Towards subband-based speech recognition," in *Proc. EUSIPCO 1996*. IEEE, 1996, pp. 1–4.

[58] H. Hennansky, S. Tibrewala, and M. Pavel, "Towards ASR on partially corrupted speech," in *Proc. Fourth International Conference on Spoken Language Processing. ICSLP '96*, vol. 1, 1996, pp. 462–465 vol.1.

[59] N. Naderi and B. Nasersharif, "Multiresolution convolutional neural network for robust speech recognition," in *Proc. 2017 Iranian Conference on Electrical Engineering (ICEE)*, 2017, pp. 1459–1464.

[60] X.-L. Zhang and D. Wang, "A deep ensemble learning method for monaural speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 967–977, 2016.

[61] Y. Chiba, T. Nose, and A. Ito, "Multi-stream attention-based BLSTM with feature segmentation for speech emotion recognition," in *Proc. Interspeech 2020*, 2020.

[62] Y. Li, X. Zhang, H. Jin, X. Li, Q. Wang, Q. He, and Q. Huang, "Using multi-stream hierarchical deep neural network to extract deep audio feature for acoustic event detection," *Multimedia Tools and Applications*, vol. 77, no. 1, pp. 897–916, 2018.

[63] 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.

[64] B. McFee *et al.*, "Librosa: Audio and music signal analysis in python," in *Proc. 14th Python in Science Conference*, 2015.

[65] D. Can, V. R. Martinez, P. Papadopoulos, and S. S. Narayanan, "PyKaldi: A Python wrapper for Kaldi," in *Proc. ICASSP*. IEEE, 2018.

[66] R. Fer *et al.*, "Multilingually trained bottleneck features in spoken language recognition," *Computer Speech & Language*, vol. 46, no. Supplement C, pp. 252 – 267, 2017.

[67] S. A. Sheikh, M. Sahidullah, F. Hirsch, and S. Ouni, "End-to-end and self-supervised learning for ComParE 2022 stuttering sub-challenge," in *Proc. ACM Multimedia 2022*, 2022.

[68] J. Opitz and S. Burst, "Macro F1 and macro F1 a note," *arXiv preprint arXiv:1911.03347*, 2019.