XSpeech: Multi-Branch Learning for Multi-Class Stuttering Classification with SpecAugment

Qianheng Xu

Received June 09, 2025
Accepted August 01, 2025
Electronic access August 31, 2025

Stuttering is a speech impairment characterized by disruptions and prolongations in speech fluency. These speech disfluencies can hinder communication and contribute to increased anxiety and depression. This paper introduces XSpeech, a novel deep learning framework designed to classify stuttered speech into five phonologically distinct stuttering types: sound repetitions, word repetitions, interjections, blocks, and prolongations. XSpeech uses data from the SEP-28K dataset, an annotated corpus of podcast recordings with interviews of people who stutter. XSpeech transforms speech signals into log-Mel spectrograms and applies the SpecAugment augmentation technique to increase generalizability and reduce overfitting. The main XSpeech architecture incorporates three key components: convolutional layers for spectrogram feature extraction, a Bidirectional Long Short-Term Memory (BiLSTM) for temporal sequence processing, and a multi-head self-attention mechanism to model global contextual relationships. Using a multi-branch learning pipeline, XSpeech decides between a binary classification of fluent or stuttered for each of the 5 types of stuttering. Each of the 5 binary classifiers has accuracies ranging from 90.0% to 96%. The predictions of each binary classifier are considered in the final prediction, allowing XSpeech to predict multiple forms of stuttering in a single speech sequence. It achieves an overall accuracy of 92.1% when classifying between all 5 types of stuttering. Furthermore, XSpeech demonstrates the improvement that SpecAugment offers, providing an overall 8.7% accuracy boost over non-augmented data. Future work will focus on expanding dataset diversity and optimizing for real-world deployment.

Keywords: Stuttering Classification, Deep Learning, Computational Linguistics, Audio Processing

1 Introduction

1.1 Background

Stuttering is a prevalent speech disorder impacting more than 70 million individuals globally[1]. As shown in Table 1, it manifests through interruptions in speech flow, including repetitions (e.g., ”a-a-and”), prolongations (e.g., ”ssssso”), and blocks (involuntary pauses)[2]. Such interruptions can significantly hinder effective communication, frequently causing social anxiety, depression, and a diminished quality of life[3]. In children, stuttering may lead to bullying and social isolation, exacerbating the emotional and psychological difficulties they encounter[4].

1.2 Problem Statement

Despite the prevalence of stuttering and the urgency for early intervention, the field of automatic detection and classification of stuttering still contains potential for research, especially with respect to the full diversity of stuttering types. Many prior studies have been limited to only two or three disfluency categories due to drawbacks around traditional feature extraction. Even when classification was attempted, models often showed strong performance in one category while underperforming in others, leading to biased or incomplete results[5].

Classification	**Phonological Pattern**
Sound Repetition (SoundRep)	“a-a-and”
Word Repetition (WordRep)	“and and”
Interjection	“I think that – uhmm”
Block	“I think...<pause>...that”
Prolongation	“sooooo”

Table 1 Classification Types and Phonological Patterns. A summary of the five stuttering categories considered in this study, with examples of each type. These categories are based on phonological patterns commonly used in speech pathology.

1.3 Significance and Purpose

This research proposes XSpeech, a deep learning-based framework for the automatic detection and classification of all five primary stuttering disfluencies. By leveraging advanced architectures such as convolutional neural network (CNN), bidirectional long short-term memory (BiLSTM) network, and the Attention mechanism, this work moves beyond the limitations of handcrafted features and limited-scope classifiers. It aims to demonstrate that modern deep learning (DL) methods can learn nuanced, context-aware patterns from raw or semi-processed audio inputs, significantly improving classification accuracy across all stutter types.

1.4 Context

While stuttering is a prevalent speech impediment, stutter detection and classification are fields which could greatly benefit from modern DL techniques. Previous computational methods for stutter detection involve signal processing methods that extract handcrafted features from specific speech components. These features are typically classified using Multi-Layer Perceptrons (MLPs) or alternative machine learning methods such as Radial Basis Function (RBF) networks, Support Vector Machines (SVMs), and Hidden Markov Models (HMMs). Commonly extracted features include Mel-Frequency Cepstral Coefficients (MFCCs), Linear Predictive Cepstral Coefficients (LPCCs), Linear Predictive Coding (LPC), and Perceptual Linear Prediction (PLP)[6–8].

Chia Ai et al. shows that LPCC slightly outperformed MFCC features by around 2% accuracy in classifying repetitions and prolongations[6]. On the contrary, Fook et al. shows MFCC features slightly outperform LPCC, WLPCC, LPC, and PLP features, though all approaches return relatively similar accuracies[7]. Furthermore, Mahesha and Vinod et al. benchmarked LPC, LPCC, and MFCC features with an SVM classifier to determine between syllable repetitions, word repetitions, and prolongations. They showed LPCC features were by far the most accurate for all three classifications. While previous research did not distinguish between syllable and word repetitions, Mahesha and Vinod et al. were able to distinguish between both categories and achieve an overall 92% accuracy with LPCC features. However, they were still unable to classify interjections and blocks due to dataset limitations[9]. Table 2 shows the general techniques and results found by previous work on feature extraction-based stutter classification.

Modern deep learning methods allow models to generalize and learn thousands of bespoke features, allowing the model to fit for all 5 categories of stuttering without compromising accuracy in any one category. Traditional MLP models may only consist of a few fully connected hidden layers due to computational limitations. DL models can take full advantage of modern hardware to train millions of parameters on more than one input dimension[5].

Modern DL methods include the convolutional neural network (CNN), long short-term neural network (LSTM), and Attention mechanism. CNNs are especially useful for processing 2D data like images or audio spectrograms. At the core of a CNN are convolutional layers, which use small trainable filters (called kernels) to scan across the input. At each position, the CNN computes a dot product between the kernel and the corresponding section of the input. As the CNN trains, these kernel weights are updated so that the CNN learns to detect important features—such as edges, patterns, or other relevant visual structures in the data[10].

Long short-term neural networks solve the vanishing gradient problem, where the gradients used for updating weights during backpropagation become approach 0, halting any model improvement. Previous recurrent neural networks suffered from the vanishing gradient as input sequences became longer. LSTMs mitigate this by using memory cells and a “forget” gate, allowing the model to retain and discard data. More importantly, LSTMs can handle long-term contextual features if they are pertinent to the model’s accuracy. This works well for speech sequences as different words can have an impact on the grammar and word choice used several words later[11,12].

More recently, the Attention mechanism is an improvement upon the encoder-decoder model originally developed for machine translation. The encoder-decoder model condenses the entire input into a fixed-length vector, which is restrictive for longer input sequences. Previous studies show that encoder-decoder model accuracy decreases as the length of input sequence increases. The Attention mechanism encodes each token in the input as a vector. Every token is assigned a Query (Q), Key (K), and Value (V) vector. To compute attention for a token, we take its Query and compare it to the Keys of all tokens using dot products. These comparisons produce attention scores, which are normalized (typically with softmax) to become attention weights. Attention weights represent how much attention the current token should give to each other token. Finally, the model uses these attention weights to compute a weighted sum of the Value vectors. This result is a new vector that captures the most relevant information from the other tokens, tailored to the current token[13,14].

More recently, there has been a shift into deep learning as a power way to classify stutters. Sheikh et al. presents StutterNet, a multi-class Time Delay Neural Network (TDNN). Because TDNNs use windows of time-delayed inputs, they are especially well-suited to temporal dependencies such as MFCC speech features[15]. Additionally, Kourkounakis et al. introduced FluentNet, a Squeeze-and-Excitation Residual Network (SE-ResNet) combined with a Bidirectional Long Short-Term Memory (BiLSTM) layer. The SE-ResNet learned effective spectral frame-level representations by emphasizing important features through residual and channel-wise attention mechanisms. The spectral embeddings were passed into the BiLSTM layer, which learned temporal dependencies and contextual information from speech sequences in both forward and backward directions[2]. Table 3 explores some of the most recent work on deep learning-based stutter classification.

By harnessing modern DL techniques such as convolutional networks, BiLSTMs, and the attention mechanism, XSpeech shows the untapped potential of DL in stuttering detection and

Author	Dataset	Classifications	Features	Classifier	Best Accuracy
Chia Ai et al., 2012[6]	39 speech samples from UCLASS	Prolongations, SoundRep, WordRep	MFCC	Linear Discriminant Analysis (LDA)	90%
Fook et al., 2013[7]	UCLASS	Prolongations, SoundRep, WordRep	15 MFCC features	SVM	95.67%
Mahesha & Vinod, 2013[9]	UCLASS	Prolongations, SoundRep, WordRep	LPCC	SVM	92.0%

Table 2 Summary of Prior Work Using Feature-Based Approaches for Stuttering Classification. This table presents a comparative overview of selected studies that employed handcrafted acoustic features and classical machine learning classifiers to identify stuttering types. All studies used datasets derived from UCLASS and focused on three disfluency types: prolongations, sound repetitions (SoundRep), and word repetitions (WordRep). The features used include Mel-Frequency Cepstral Coefficients (MFCCs) and Linear Predictive Cepstral Coefficients (LPCCs), with classification performed using Linear Discriminant Analysis (LDA) or Support Vector Machines (SVMs). Reported best-case classification accuracies range from 90% to 95.67%.

Author	Dataset	DL Method	Best Accuracy
Sheikh et al., 2021[15]	UCLASS	Time-delay neural network	50.79%
Kourkounakis et al., 2020[2]	UCLASS, LibriStutter	SE-ResNet and BiLSTM	96.6% for WordRep
Basak et al., 2023[16]	SEP-28K, FluencyBank	Transformer	88.1%

Table 3 Summary of Recent Deep Learning Approaches to Stuttering Classification. This table highlights recent advances in the application of deep learning (DL) architectures for classifying stuttered speech. Studies utilized diverse datasets, including UCLASS, LibriStutter, SEP-28K, and FluencyBank, and explored a range of neural models such as Time Delay Neural Networks (TDNNs), Squeeze-and-Excitation Residual Networks (SE-ResNets) combined with Bidirectional Long Short-Term Memory (BiLSTM) layers, and Transformer-based architectures. Each model targeted specific disfluency types and reported varying levels of accuracy. Kourkounakis et al. (2020) achieved the highest task-specific performance with 96.6% accuracy for word repetitions (WordRep).

classification.

2 Methods

2.1 Data collection and Evaluation

The study utilizes the Stuttering Events in Podcasts (SEP-28K) dataset, which comprises of 28,000 manually annotated audio clips from individuals who stutter[17]. All audio clips labeled with “PoorAudioQuality,” “Music,” or “NoSpeech” labels were filtered out, leaving 16,293 audio clips with clear human speech. While the study reports 92% accuracy based on 16,293 clean audio clips from SEP-28K, it’s important to note that the filtered-out clips were removed solely due to the absence of discernible speech. These excluded clips did not contain usable speech content and were not removed based on audio quality or attributes[17]. Therefore, the filtering process did not introduce bias against any specific type of stuttering behavior; rather, it ensured that the model was trained on clips where speech was actually present. Labels such as “Unintelligible” where there was difficult-to-understand speech were kept in the dataset and used in this work. The dataset is labeled with five primary stuttering types: sound repetition, word repetition, interjection, block, and prolongation. To effectively capture stuttering events while maintaining computational efficiency, each audio recording was segmented into 3-second fragments. This duration was selected based on prior research, which suggests that most stuttering disfluencies occur within 1–5 seconds. A 3-second window provides sufficient temporal context for encapsulating stutters while minimizing the inclusion of irrelevant speech surrounding it[18,19]. Each segmented recording was resampled to 16 kHz which is a widely adopted standard in speech processing, as it preserves relevant frequency components (up to 8 kHz) while maintaining computational efficiency. Poor-quality clips, such as those with significant background noise or incomplete speech, were removed to improve the reliability of the training dataset.

2.2 Feature Extraction

Before the model can be trained, features must be extracted from the raw input audio to prepare and clean the data for training. For each audio fragment, the following four-step process is used to clean and extract features, as shown in Figure 1:

1. A raw audio waveform is generated as an image. The waveform plots the amplitude of the sound at any given timestep.

2. The amplitude-time waveform is converted to a spectrogram by applying a Short-time Fourier Transform (STFT). STFT is derived from the original Fourier Transform, an algorithm to decompose a signal into its constituent sinusoidal frequencies and amplitudes. In practice, this converts the signal from the time domain to the frequency domain. The resulting frequency vs. amplitude graph is called the spectrum. However, the Fourier Transform only applies to a static signal that doesn't change over time. Unfortunately, speech signals are always changing as we speak. So, the STFT algorithm is applied by computing the Fourier Transform spectrum for overlapping windows cut from the signal. Finally, each spectrum is layered on top of each other to create a spectrogram.

3. The spectrogram is converted into a log-Mel spectrogram by mapping the frequencies from the "vanilla" spectrogram onto the Mel scale. This is done because human hearing is not linearly sensitive towards pitch – In fact, we are more sensitive towards changes in lower pitch than high pitch. In practice, this means that while humans can easily distinguish between a 500 Hz and a 750 Hz tone, the difference between a 10,000 Hz and a 10,250 Hz tone is far less noticeable, even though both pairs are separated by the same 250 Hz interval. The Mel scale, first introduced by Stevens et al. (1937)[20], was a novel way to model how humans perceive pitch. To convert a spectrogram into a log-Mel spectrogram, the linear frequency spectrogram is passed through a Mel filter bank, which applies a series of triangular filters to aggregate spectral energy within perceptually relevant frequency bands. After converting a spectrogram to a log-Mel spectrogram, equal distances on the Mel scale correspond to equal perceived distances in pitch. The functional impact is that Mel spectrograms emphasize frequencies of human speech while also reducing the strength of background noise signals, helping the model to focus only on the speech[10,16,17]. Specifically, the log-Mel spectrograms were created with Fast Fourier Transform (FFT) windows of 2048 samples long, with a new FFT window being taken every 512 samples. Previous research shows that log-Mel spectrograms are becoming more popular and effective than mel frequency cepstral coefficients (MFCC) due to advancements in convolutional neural networks and image processing[5,20,21].

4. Lastly, the SpecAugment data augmentation algorithm was applied to create a random mask of a range of values on both the time and frequency domain[22].

The result of the data preparation process is a dataset of augmented mel spectrogram images for every 3-second fragment. The XSpeech model was trained on 80% of the total spectrogram images and tested on the 20% remaining images.

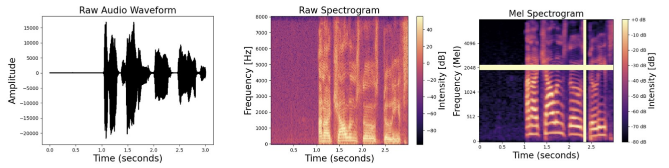

Fig. 1 Converting Raw Audio into a Log-Mel Spectrogram with SpecAugment. This figure shows the four main steps used to prepare the audio for deep learning. A raw waveform is first generated, then converted into a spectrogram using STFT. The spectrogram is then mapped to the Mel scale and turned into a log-Mel spectrogram. Finally, SpecAugment is applied to randomly mask parts of the spectrogram in time and frequency to improve model training.

2.3 Model Architecture

XSpeech uses a multi-branch architecture, using a "divide and conquer" approach to stutter classification. Instead of a single multi-class classifier, the multi-branch architecture (Figure 2) employs five independent binary classifiers, each dedicated to distinguishing one stuttering type from fluent speech. The prediction of each binary classifier is independently considered, allowing XSpeech to simultaneously predict more than one type of stutter for any speech sequence. That is, the logits of each binary classifier are individually considered. If more than one classifier returns a "positive" result, all positive results are counted in the final prediction. This multi-branch method decreases the workload that any single model must handle and distributes the computation across several specialized models. Figure 2 shows the ensemble classification pipeline.

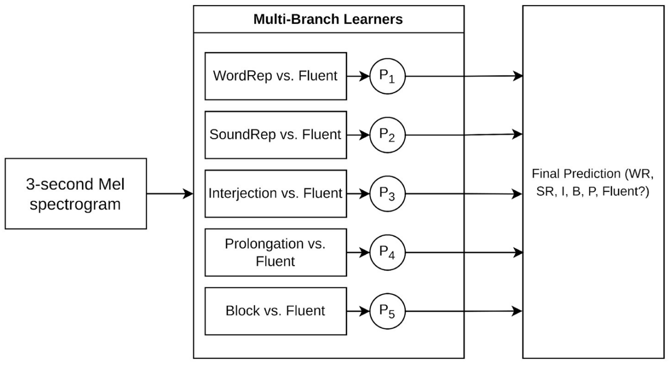

Fig. 2 High-level architecture of XSpeech. The model uses five independent binary classifiers, each trained to detect one specific disfluency type. The architecture supports multi-label prediction: multiple classifiers can return positive outputs simultaneously, reflecting the real-world co-occurrence of stuttering types.

Each binary classifier is built on a hybrid deep learning architecture that integrates convolutional neural networks (CNNs), Bidirectional LSTM (BiLSTM) networks, and a multi-head self-attention mechanism. Figure 3 shows the high-level organization of one of the binary classification models.

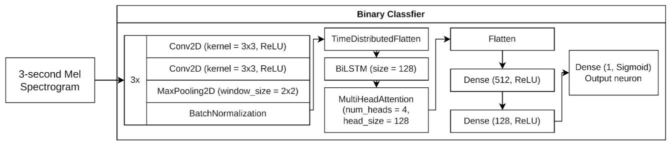
Binary Classfier
3-second Mel Spectrogram
3x
Conv2D (kernel = 3x3, ReLU)
Conv2D (kernel = 3x3, ReLU)
MaxPooling2D (window_size = 2x2)
BatchNormalization
TimeDistributedFlatten
BiLSTM (size = 128)
MultiHeadAttention (num_heads = 4, head_size = 128
Flatten
Dense (512, ReLU)
Dense (128, ReLU)
Dense (1, Sigmoid) Output neuron

Fig. 3 Binary Classifier Architecture in XSpeech. Each stuttering type is handled by its own classifier. The input spectrogram passes through CNNs, a BiLSTM layer, and a self-attention layer before producing a binary output.

The model begins with an input layer that accepts images of shape 224×224×1, corresponding to the single-channel, 2D array of the log-Mel spectrogram. There are three convolutional blocks, each with two 2D convolutional layers. The first convolutional block has two Conv2D layers with 64 filters, filter size 3x3, and ReLU activation. The second and third convolutional blocks have similar parameters but with 128 and 256 filters, respectively.

The CNNs extract spatial patterns from spectrogram images through a hierarchical structure of convolutional layers. CNNs have demonstrated superior ability to analyze and extract features from complex spectrograms and other types of two-dimensional data. Unlike handcrafted feature extraction techniques, CNNs automatically learn feature representations, improving robustness to variations in speech patterns and background noise. By adding augmented data into the training set, the CNN layers are forced to learn which parts of the audio have been "corrupted" by SpecAugment and learn more general features of the audio.

To explore how the CNN layers respond to augmented input, I extracted and visualized activation maps from the Conv2D_2 layer in the second convolutional block across multiple test clips. Activation maps highlight the regions in the input spectrogram that most strongly activate particular filters, thereby revealing the features the network deems salient. As shown in Figure 4, several maps consistently display low activation (dark blue or black) in regions affected by SpecAugment masking bands, while maintaining high activation in unmasked areas. This recurring pattern across clips suggests that the CNN is learning to down-weight or ignore artificially distorted segments during feature extraction.

The output of the final convolution block is flattened into 1D vector for every timestep.

Following convolutional feature extraction, XSpeech uses a bidirectional long short-term memory (BiLSTM) network with 128 units to capture temporal dependencies in speech sequences. The long short-term memory network can capture long-term dependencies in sequential and time series tasks since it mitigates the vanishing and exploding gradient problems. Unlike standard LSTMs, BiLSTMs process information in both forward and backward directions, enabling the model to understand both past and future context within a speech segment[23]. This dual processing improves classification accuracy, being particularly useful for stuttering classification, where certain disfluencies depend on both preceding and following phonemes[24].

To further enhance feature selection, a multi-head self-attention mechanism with 4 heads with size 128 for each query and key head was used. This mechanism assigns higher importance to key temporal and spectral regions within the spectrogram, allowing the model to focus on distinctive stuttering cues while ignoring background noise or non-informative speech patterns. This is crucial for stuttering classification because certain disfluencies (e.g., blocks and prolongations) occur in localized regions of speech signals. The attention-weighted feature representations are then passed through two fully connected layers (512 and 128 neurons each, respectively) with dropout layers of 0.5 in between to prevent overfitting. The final layer uses sigmoid activation on a single neuron, producing a binary classification output for each stuttering type. An L2 regularizer was used on the convolution and dense layers to discourage exploding weights and reduce overfitting[25].

Rather than using a single multi-class classifier, this study employs an ensemble of five binary classifiers, each specializing in a specific stuttering type. The predictions from each classifier are aggregated and any positive results are counted. This allows XSpeech to simultaneously predict more than one type of stutter for any audio sample as each binary classifier operates independently of another.

2.4 Training and Implementation

The XSpeech model was implemented using TensorFlow (Keras API) and trained using the Adam optimizer with an initial learning rate of 1e-4, which was dynamically adjusted to 1e-8 using the ReduceLROnPlateau function to enhance convergence. An EarlyStopping callback was also used to prevent overfitting and save the most accurate weights from the training session. A binary cross-entropy loss function was used for individual classifiers. Each binary classifier was trained for a maximum of 150 epochs with a batch size of 36. All parameters were tuned manually Dropout. layers with a probability of 0.4 were included to reduce overfitting by randomly deactivating a fraction of neurons during training. All hyperparameters were tuned in consideration of hardware limitations and accuracy.

Training was conducted on an NVIDIA RTX 3080 GPU (10GB VRAM) to accelerate computations. The entire model contains approximately 12.3 million parameters and takes around 0.005159 seconds to run inference on a 3-second clip of audio. The dataset was split into 80% training data (13,034 spectrograms) and 20% testing data (3,258 spectrograms) with an episode-level split. Out of all the episodes, 80% were allocated to the training set and 20% for the testing set. This meant

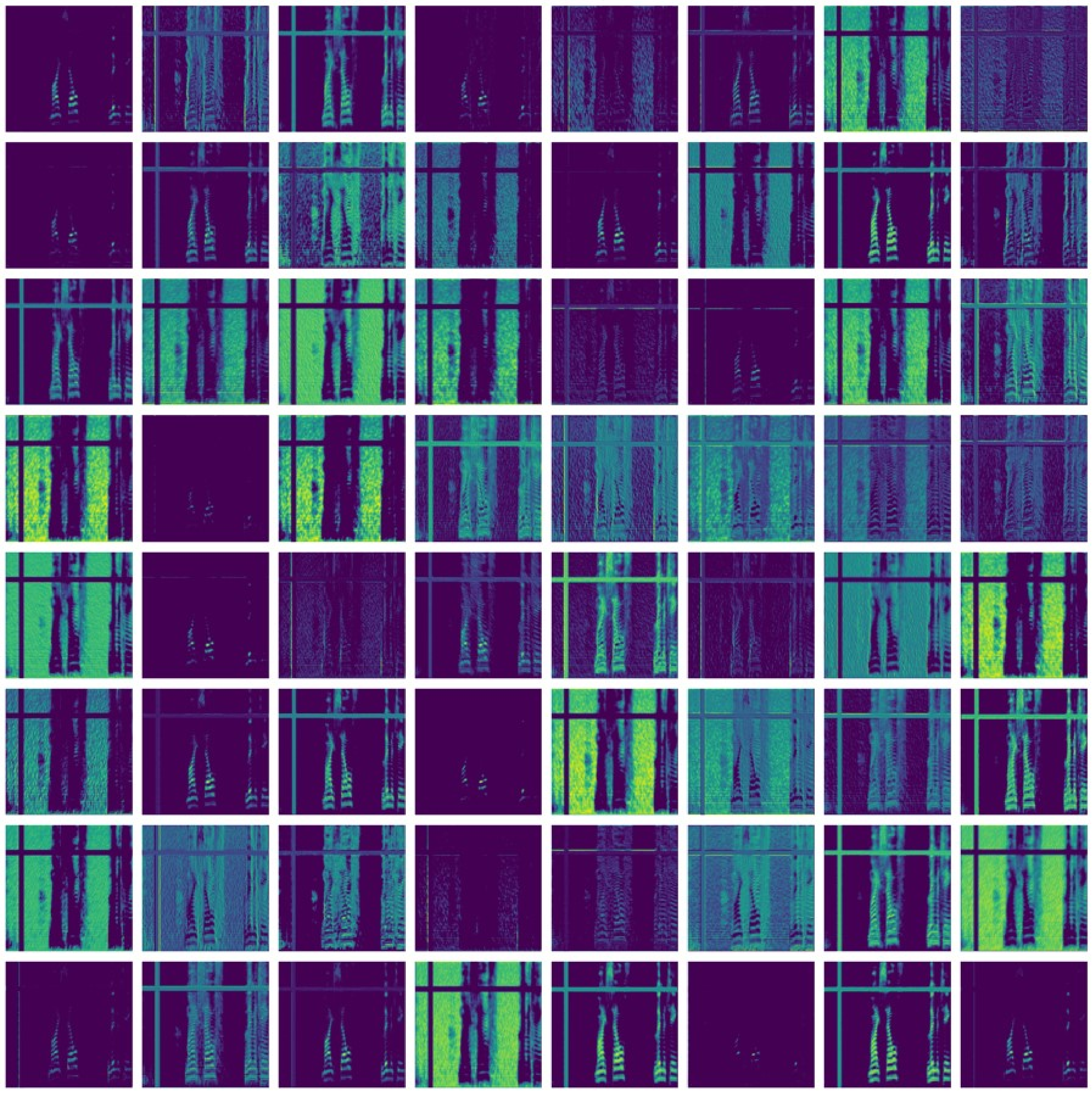

Fig. 4 A set of activation maps extracted from the second convolutional block with 64 filters. These maps highlight salient regions of the input spectrogram for each specific filter. Some filters have learned to ignore the SpecAugment-warped bands, showing the convolutional block has learned to ignore the augmentation during its prediction.

that all clips from a given podcast episode appear in only one of the two sets. This strategy helps prevent the model from memorizing speaker or microphone characteristics that are consistent within an episode. Without such a split, adjacent or similar clips could appear in both training and testing sets, leading to inflated performance metrics due to shared acoustic or speaker-specific cues. Each binary classifier was trained independently.

3 Results

3.1 Performance Evaluation

XSpeech was evaluated using standard classification metrics, including accuracy, precision, recall, F1-Score, and Cohen's kappa coefficient to measure classification performance across different stuttering types. Accuracy refers to the proportion of correctly predicted samples across the entire dataset, while precision measures the proportion of predicted positives that are truly positive. Recall evaluates how effectively the model identifies actual positive cases, and F1-score provides a harmonic mean between precision and recall, offering a balanced measure of model performance. Cohen's kappa coefficient accounts for the possibility of chance agreement, making it particularly useful for datasets with class imbalance. Given that speech segments in SEP-28K often contain more than one type of stuttering, this evaluation adopts a multi-label classification framework. A prediction was counted as correct only when the model successfully identified all stuttering types present in the audio segment. To avoid inflating results due to class imbalance, all reported metrics are macro-averaged, ensuring that each stutter type contributes equally to the overall evaluation.

On the 20% test split of SEP-28K, XSpeech achieved strong results across all categories. Accuracy ranged from 90.0% in block classification to 96.3% for word repetition. The model attained F1-scores of 0.88 for block, 0.93 for prolongation, 0.92 for interjection, 0.96 for word repetition, and 0.95 for sound repetition. These results suggest that XSpeech is capable of robust and reliable classification across all five major types of disfluency, even when multiple stutter types occur simultaneously within a single utterance. Notably, performance was consistently strong across categories, indicating that the model does not disproportionately favor more frequent disfluency types. This reinforces the effectiveness of XSpeech's multi-branch architecture in supporting real-world, multi-label stuttering detection tasks.

3.2 Robustness Evaluation via Five-fold Cross Validation

Five-fold cross-validation is a widely adopted method for assessing the robustness and generalizability of machine learning models. In this procedure, the full dataset is randomly shuffled and partitioned into five equal subsets, or "folds." For each of the five iterations, the model (in this case, XSpeech) is trained on four folds and evaluated on the remaining one. This rotation continues until every fold has served as the test set exactly once.

Each of XSpeech's five binary classifiers—Block, Interjection, Prolongation, Sound Repetition, and Word Repetition—was evaluated using this cross-validation strategy. For every fold, I generated a confusion matrix to capture true positives, false positives, true negatives, and false negatives. Across all folds, the classifiers consistently demonstrated high accuracy and low variance, indicating stable performance and strong generalization. The confusion matrices for each classifier are shown in Figure 5.

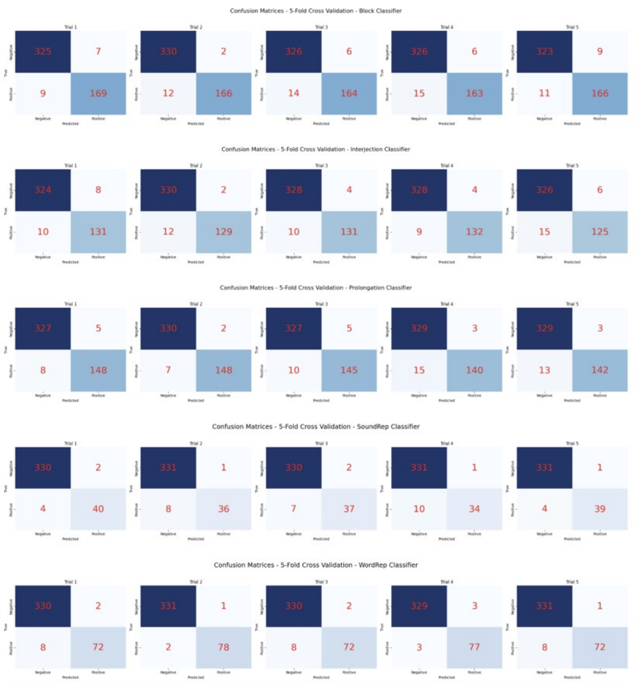

Fig. 5 Confusion Matrices from 5-Fold Cross-Validation. This matrix shows how often the model correctly predicted each stuttering type versus how often it made errors. Darker diagonal cells represent correct predictions; off-diagonal cells indicate confusion between types.

These matrices illustrate the distribution of true positives, false positives, true negatives, and false negatives, providing a detailed view of classifier performance and consistency across folds.

3.3 Comparative Analysis with Existing Models

To benchmark XSpeech against prior works, its performance was compared with FluentNet and ResNet+BiLSTM. Although these baseline models were originally trained on the UCLASS dataset, they share similar architectural strategies and evaluation metrics, including class-wise accuracy on specific stutter types. This allows for a qualitative comparison of architectural effectiveness, even if direct numerical benchmarking is not strictly valid. However, it is important to acknowledge that the UCLASS dataset differs substantially from SEP-28K in terms of size, speaker diversity, language style, and annotation quality. Additionally, UCLASS is no longer publicly available, and its limited scale may constrain generalizability. These cross-dataset differences introduce several confounding variables, including potential discrepancies in preprocessing and data distributions, making quantitative comparisons imprecise.

Classification	Accuracy	Precision	Recall	F1-Score	Cohen's Coefficient
Block vs. Fluent	90.0%	0.88	0.89	0.88	0.7619
Prolongation vs. Fluent	93.1%	0.92	0.92	0.93	0.834
Interjection vs. Fluent	92.4%	0.89	0.90	0.92	0.8086
WordRep vs. Fluent	96.3%	0.93	0.95	0.96	0.861
SoundRep vs. Fluent	95.0%	0.94	0.95	0.95	0.713
Overall	92.1%	0.91	0.93	0.92	0.8021

Table 4 Classification performance of XSpeech on test split. This table shows how well XSpeech classified each type of stuttering using five evaluation metrics: accuracy, precision, recall, F1-score, and Cohen's kappa Coefficient. Results are based on a multi-label setup, where predictions had to match all stuttering types in a clip. All values are macro-averaged to give equal weight to each stutter type.

Classification	XSpeech (This Work)	FluentNet (Kourkounakis et al., 2020)	ResNet+BiLSTM (Kourkounakis et al., 2020)
Block vs. Fluent	90.0%	N/A	N/A
Prolongation vs. Fluent	93.1%	94.9%	94.1%
Interjection vs. Fluent	92.4%	81.9%	81.4%
WordRep vs. Fluent	96.3%	96.6%	96.6%
SoundRep vs. Fluent	95.0%	84.4%	84.1%

Table 5 Comparative performance analysis of XSpeech with prior models. This table compares the classification performance of XSpeech with two models from Kourkounakis et al. (2020): FluentNet and ResNet+BiLSTM. XSpeech performs similarly or better across most stuttering types, especially for interjections and sound repetitions. Results are shown for each stuttering type. "N/A" indicates the previous models did not report results for that category.

As such, the results presented in Table 5 are illustrative only and are intended to highlight the potential advantages of the XSpeech architecture rather than to serve as conclusive benchmarks. For example, XSpeech shows improved performance on interjections (+10.5%) and sound repetitions (+10.6%) compared to FluentNet.

3.4 Impact of SpecAugment on Performance

To assess the impact of the SpecAugment data augmentation step, XSpeech was retrained using the same model hyperparameters but with two different data augmentation techniques: time-stretching and pitch-shifting. The data is time-stretched by factor of 1.25, slightly speeding up the audio. Unlike simply increasing the playback speed, time stretching does not modify the pitch of the audio. XSpeech is also tested with pitch-shifted data, where all frequencies are raised by a whole tone. Finally, XSpeech is also trained on data with no augmentation applied. The accuracy comparison between XSpeech trained with all data augmentation techniques is shown in Table 6.

As shown in Table 6, any sort of augmentation introduced slight improvements in accuracy, though time stretching and pitch shifting offer minor improvements compared to SpecAugment. This confirms that SpecAugment effectively enhances model generalization by introducing variability in spectrogram patterns, allowing the model to better adapt to unseen speech variations and generalize between differing speaking conditions.

3.5 Ablation Experiments

To evaluate the contribution of individual architectural components in XSpeech, ablation experiments were conducted under controlled conditions. Each binary classifier was retrained using SpecAugment-augmented data for 150 epochs with a batch size of 36, keeping all other hyperparameters constant, including the learning rate and ReduceLROnPlateau scheduler. The experiments involved selectively removing the BiLSTM layer, the multi-head self-attention layer, and both together, to observe their individual and combined effects on classification accuracy.

The results, presented in Table 7, show that removing either the BiLSTM or the attention mechanism leads to a noticeable decrease in performance across all five stuttering types. The largest drop in accuracy occurs when both components are removed, confirming their complementary roles in modeling temporal and contextual dependencies in speech. For example, in the Block vs. Fluent classifier, accuracy dropped from 90.0% to 83.8% without the BiLSTM, to 81.4% without attention, and further to 77.4% when both were removed. Similar patterns were observed across the other classifiers, with especially pronounced declines in the Interjection and Prolongation tasks, where accurate classification likely relies on understanding more nuanced acoustic cues over time.

Classification	With SpecAugment	Time-stretched (factor of 1.25)	Pitch-shifted (1 tone)	No data augmentation	Improvement vs. No Augmentation
Block vs. Fluent	90.0%	80.2%	79.8%	75.91%	14.29%
Prolongation vs. Fluent	93.1%	83.3%	86.2%	80.53%	12.57%
Interjection vs. Fluent	92.4%	84.6%	83.1%	87.96%	4.44%
WordRep vs. Fluent	96.3%	84.3%	85.5%	87.6%	8.70%
SoundRep vs. Fluent	95.0%	84.5%	85.0%	91.79%	3.21%
Overall	92.1%	83.2%	84.0%	83.4%	8.7%

Table 6 Comparison of Data Augmentation Methods. This table compares SpecAugment with other common techniques like pitch shifting and time stretching. SpecAugment gave the best overall results, helping the model generalize better to new data.

Classification	No Ablation	BiLSTM Removed	Attention Removed	Both BiLSTM and Attention Removed
Block vs. Fluent	90.0%	83.8%	81.4%	77.4%
Prolongation vs. Fluent	93.1%	85.2%	81.1%	80.3%
Interjection vs. Fluent	92.4%	86.9%	82.6%	78.7%
WordRep vs. Fluent	96.3%	89.5%	84.8%	80.2%
SoundRep vs. Fluent	95.0%	87.3%	83.6%	79.9%

Table 7 Ablation Study on Key XSpeech Components. This table shows how removing parts of the model—BiLSTM, self-attention, or both—affects overall classification accuracy. The full model performs best, demonstrating that each component contributes meaningfully to performance.

4 Conclusion

4.1 Key Findings

The results of XSpeech demonstrate the effectiveness of multi-branch learning using SpecAugment for data augmentation to increase the model's generalizability. Through ablation, we also show the importance of all three convolution, BiLSTM, and Attention components to the performance of XSpeech. Removal of the BiLSTM component resulted in a noticeable performance degradation across all tasks, with an average accuracy drop of approximately 6.7 percentage points. This highlights the critical role of the BiLSTM in capturing temporal dependencies and contextual information within the acoustic features. Exclusion of the Attention mechanism led to a consistent decline in performance, albeit slightly more severe in some cases (e.g., Block vs. Fluent and Prolongation vs. Fluent), with an average reduction of 9.1 percentage points. The most significant degradation was observed when both BiLSTM and Attention components were removed simultaneously. The resulting classifiers showed the lowest accuracies across all tasks, with performance dropping by up to 18.9 percentage points in the Word Repetition vs. Fluent task. This confirms that the BiLSTM and Attention modules contribute complementary strengths to the model's capacity to model sequential and contextually nuanced acoustic patterns.

Another key finding is the significant impact of SpecAugment on model performance. Given the relatively small nature of the SEP-28k dataset, data augmentation plays a critical role in improving generalization. Through testing different data augmentation technique, we confirm SpecAugment significantly outperforms time-stretching and pitch-shifting as possible data augmentation techniques. This effect is most evident in block (+14.29%) and prolongation classification (+12.57%).

4.2 Limitations and Future Work

Despite achieving state-of-the-art accuracy, several limitations remain. First, there are significant dataset restrictions. Although the SEP-28k corpus is large and well-annotated, it contains only English-language speech, which inherently limits the generalizability of XSpeech to non-English speakers or even to English speakers with strong regional or non-native accents. To address this limitation, future research should explore multilingual stuttering datasets and investigate transfer learning approaches to adapt the model across languages and dialects. Additionally, unsupervised or semi-supervised learning on unlabeled or minimally labeled foreign-language corpora may provide a promising avenue for expanding XSpeech's cross-linguistic capabilities.

Furthermore, two widely cited datasets for stuttering research—UCLASS and FluencyBank—were not accessible at the time of this study. UCLASS has become private, and FluencyBank is protected, offering access to only a limited number of audio recordings via password. Validating XSpeech on Fluency-

Bank, in particular, remains a key goal for future work aimed at assessing cross-dataset generalization and clinical relevance.

Another key limitation involves the audio quality of the training data. While XSpeech was trained on the 16,293 speech clips filtered from SEP-28K, this subset excluded clips labeled "NoSpeech," "Music," or "PoorAudioQuality," which contained little to no intelligible speech. However, we emphasize that the retained clips still reflect a wide variety of speaker traits, background noise levels, and recording conditions, including spontaneous speech with disfluencies in semi-structured podcast settings. Nevertheless, we acknowledge that real-world clinical environments often present harsher acoustic conditions than those found in curated datasets. While we report macro-averaged metrics to ensure equal weighting across classes and apply SpecAugment to improve generalization, we acknowledge that additional techniques such as focal loss, class-weighted loss functions, or resampling of underrepresented classes could help mitigate performance disparities across stutter types. Incorporating these strategies into future versions of XSpeech may lead to further improvements, particularly in the recall of rarer disfluency types.

XSpeech has not been tested in real-time speech environments or with spontaneous conversational speech. Future research should implement real-time streaming models and conduct experiments in clinical speech therapy settings. Also, the multi-branch model introduces additional computational overhead. XSpeech has only been tested on a powerful NVIDIA GPU in an x86 system and may not perform as well on ARM or mobile devices. Future work will explore lightweight architectures optimized for mobile and edge deployment.

While preliminary activation map visualizations suggest that the CNN layers may learn to suppress augmented or noisy regions introduced by SpecAugment, these results remain qualitative. The analysis was based on a limited set of representative clips and layers, and thus cannot be taken as statistically conclusive evidence of consistent model behavior. Future work will involve a more rigorous interpretability study using techniques such as Gradient-weighted Class Activation Mapping (Grad-CAM), saliency maps, or attention heatmaps to quantify how the model distributes its focus across the input spectrogram. Additionally, aggregating activation trends across a larger sample of clips and layers could reveal whether the suppression of masked regions generalizes across different speakers, stutter types, and acoustic conditions. Such analysis would deepen the understanding of how XSpeech achieves robustness and would help ensure that model decisions are based on linguistically meaningful features rather than dataset artifacts or augmentation biases.

4.3 Conclusion

This study introduced XSpeech, a multi-branch, hybrid deep learning architecture for automatic stuttering classification that integrates convolutional neural networks, bidirectional LSTM networks, and a multi-head self-attention mechanism. By employing a "divide and conquer" approach with five independent binary classifiers, XSpeech enables simultaneous detection of multiple stuttering types within a single speech sequence.

Experimental results on the SEP-28k dataset demonstrate that XSpeech achieves high performance across all stuttering types, with an overall accuracy of 92.1% and an average F1-score of 0.92. Compared to prior models, XSpeech exhibits significant improvements, particularly in detecting interjections and sound repetitions. Through ablation studies, we confirmed that the BiLSTM and attention modules play a critical role in temporal and contextual modeling, and their removal leads to substantial performance degradation. Furthermore, the use of SpecAugment as a data augmentation strategy was shown to significantly enhance generalization performance, outperforming alternative techniques such as time-stretching and pitch-shifting.

While the proposed model achieves state-of-the-art accuracy, several avenues remain for future work, including cross-lingual generalization, real-time implementation, and deployment on resource-constrained devices. The findings of this work highlight the potential of multi-branch, attention-augmented neural architectures in robustly classifying disfluent speech and contribute to the development of clinically relevant tools for speech disorder assessment.

References

1 B. Ghai and K. Mueller, *Fluent: An AI Augmented Writing Tool for People who Stutter*, https://doi.org/10.1145/3441852.3471211.

2 T. Kourkounakis, A. Hajavi and A. Etemad, *Detecting Multiple Speech Disfluencies Using a Deep Residual Network with Bidirectional Long Short-Term Memory*, https://doi.org/10.1109/ICASSP40776.2020.9053893.

3 L. Iverach, S. O'Brian, M. Jones, S. Block, M. Lincoln, E. Harrison, S. Hewat, R. Menzies, A. Packman and M. Onslow, *Prevalence of anxiety disorders among adults seeking speech therapy for stuttering*, https://doi.org/10.1016/j.janxdis.2009.06.003.

4 L. Iverach, M. Jones, L. McLellan, H. Lyneham, R. Menzies, M. Onslow and R. Rapee, *Prevalence of anxiety disorders among children who stutter*, https://doi.org/10.1016/j.jfludis.2016.07.002.

5 S. Sheikh, M. Sahidullah, F. Hirsch and S. Ouni, *Machine learning for stuttering identification: Review, challenges and future directions*, https://doi.org/10.1016/j.neucom.2022.10.015.

6 O. Chia Ai, M. Hariharan, S. Yaacob and L. Sin Chee, *Classification of speech dysfluencies with MFCC and LPCC features*, https://doi.org/10.1016/j.eswa.2011.07.065.

7 C. Fook, H. Muthusamy, L. Chee, A. Adom and S. Yaacob, *Comparison of speech parameterization techniques for the classification of speech disfluencies*, https://doi.org/10.3906/elk-1112-84.

8 S. Khara, S. Singh and D. Vir, *A Comparative Study of the Techniques for Feature Extraction and Classification in Stuttering*, `https://doi.org/10.1109/ICICCT.2018.8473099`.

9 P. Mahesha and D. Vinod, *Quality, Reliability, Security and Robustness in Heterogeneous Networks*, vol. 115, p. 298–308.

10 K. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks*, `https://doi.org/10.48550/arXiv.1511.08458`, No. arXiv:1511.08458). arXiv.

11 S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, `https://doi.org/10.1162/neco.1997.9.8.1735`.

12 R. Staudemeyer and E. Morris, *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*, `https://arxiv.org/abs/1909.09586v1`, arXiv.Org.

13 D. Bahdanau, K. Cho and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, `https://doi.org/10.48550/arXiv.1409.0473`, No. arXiv:1409.0473). arXiv.

14 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, `https://arxiv.org/abs/1706.03762v7`, Attention Is All You Need. arXiv.Org.

15 S. Sheikh, M. Sahidullah, F. Hirsch and S. Ouni, *StutterNet: Stuttering Detection Using Time Delay Neural Network*, `https://doi.org/10.23919/EUSIPCO54536.2021.9616063`.

16 K. Basak, N. Mishra and H.-T. Chang, *TranStutter: A Convolution-Free Transformer-Based Deep Learning Method to Classify Stuttered Speech Using 2D Mel-Spectrogram Visualization and Attention-Based Feature Representation*, `https://doi.org/10.3390/s23198033`.

17 C. Lea, V. Mitra, A. Joshi, S. Kajarekar and J. Bigham, *SEP-28k: A Dataset for Stuttering Event Detection From Podcasts With People Who Stutter*, `https://doi.org/10.48550/arXiv.2102.12394`, No. arXiv:2102.12394). arXiv.

18 A. Al-Banna, E. Edirisinghe, H. Fang and W. Hadi, *Stuttering disfluency detection using machine learning approaches*, `https://doi.org/10.1142/S0219649222500204`.

19 R. Lickley, *Fattori sociali e biologici nella variazione fonetica*, vol. 3, p. 373–387.

20 S. Stevens, J. Volkmann and E. Newman, *A Scale for the Measurement of the Psychological Magnitude Pitch*, `https://doi.org/10.1121/1.1915893`.

21 H. Meng, T. Yan, F. Yuan and H. Wei, *Speech Emotion Recognition From 3D Log-Mel Spectrograms With Deep Learning Network*, `https://doi.org/10.1109/ACCESS.2019.2938007`.

22 D. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. Cubuk and Q. Le, *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*, `https://doi.org/10.21437/Interspeech.2019-2680`.

23 M. Schuster and K. Paliwal, *Bidirectional recurrent neural networks*, `https://doi.org/10.1109/78.650093`.

24 V. Zayats, M. Ostendorf and H. Hajishirzi, *Disfluency Detection Using a Bidirectional LSTM*, `https://doi.org/10.21437/Interspeech.2016-1247`.

25 C. Cortes, M. Mohri and A. Rostamizadeh, `https://doi.org/10.48550/arXiv.1205.2653`, L2 Regularization for Learning Kernels (No. arXiv:1205.2653). arXiv.