StutterZero and StutterFormer: End-to-End Speech Conversion for Stuttering Transcription and Correction

QIANHENG XU[1]

Abstract—Over 70 million people worldwide experience stuttering, yet most automatic speech systems misinterpret disfluent utterances or fail to transcribe them accurately. Existing methods for stutter correction rely on handcrafted feature extraction or multi-stage automatic speech recognition (ASR) and text-to-speech (TTS) pipelines, which separate transcription from audio reconstruction and often amplify distortions. This work introduces StutterZero and StutterFormer, the first end-to-end waveform-to-waveform models that directly convert stuttered speech into fluent speech while jointly predicting its transcription. StutterZero employs a convolutional–bidirectional LSTM encoder–decoder with attention, whereas StutterFormer integrates a dual-stream Transformer with shared acoustic–linguistic representations. Both architectures are trained on paired stuttered–fluent data synthesized from the SEP-28K and LibriStutter corpora and evaluated on unseen speakers from the FluencyBank dataset. Across all benchmarks, StutterZero had a 24% decrease in Word Error Rate (WER) and a 31% improvement in semantic similarity (BERTScore) compared to the leading Whisper-Medium model. StutterFormer achieved better results, with a 28% decrease in WER and a 34% improvement in BERTScore. The results validate the feasibility of direct end-to-end stutter-to-fluent speech conversion, offering new opportunities for inclusive human–computer interaction, speech therapy, and accessibility-oriented AI systems.

Keywords—automatic speech recognition, deep learning, human-computer interaction, speech correction, speech processing, stuttering

I. Introduction

Stuttering is a prevalent speech disorder that affects more than 70 million individuals worldwide [1]. It manifests as interruptions in speech flow, including unintentional repetitions, prolongations, and pauses. Table I presents the five main phonological classifications of stuttering and examples of their symptoms [2] [3]. Such interruptions can significantly hinder effective communication, frequently causing social anxiety, depression, and a diminished quality of life [4]. In children, stuttering may lead to bullying and social isolation, exacerbating the emotional and psychological difficulties they encounter [5].

Fluent speech plays a critical role in daily communication, both in interpersonal situations and when engaging with intelligent voice-based technologies. Through a quality-of-life survey, people who stutter (PWS) showed a statistically significant decrease in emotional health and social function metrics [6]. Stuttering can also cause feelings of shame, fear, anxiety, and guilt [7]. These challenges are exacerbated by the growing dependence on intelligent voice assistants, such as Google Home, Amazon Echo, and Apple Siri, which are designed to process fluent speech. As a result, people who stutter frequently encounter recognition errors, limited functionality, and exclusion when interacting with voice-controlled technologies.

Classification	**Example 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 I: Categories of stuttering with representative phonological examples.

For more severe forms of stuttering, automatic speech recognition (ASR) models, which transcribe speech, may insert unintended words or even truncate the speech due to a blocking stutter. Lea et al. [8] evaluated the speech of individuals with PWS using the Apple Speech framework, a production-level automatic speech recognition (ASR) system designed for fluent speakers. They reported a Word Error Rate (WER) of 19.8%, indicating that 19.8% of words in the ASR-generated transcript did not match the reference ground truth. They also reported a truncation rate of 23.8%, where 23.8% of utterances from these PWS were prematurely cut off [8]. Addressing this challenge, this study develops deep learning models that convert stuttered speech into fluent audio, going beyond transcription to reconstruct corrected acoustic signals. By generating fluent speech that preserves semantic content and improves intelligibility, these models aim to enhance communication for PWS and increase accessibility in human-machine interactions.

Previous work in the field of stutter correction can be separated into three categories: (1) Digital signal processing (DSP) and rule-based classifiers, (2) ASR and text-to-speech (TTS) pipelines, and (3) deep learning (DL) approaches.

1) Conventional Digital Signal Processing Approaches: DSP approaches utilize audio feature extraction methods to obtain condensed numerical features from complex audio signals. Then, a ruleset or a set of predetermined filters is applied to the features to determine which timeframes contain a stutter. Finally, these timeframes are cut out of the audio, removing the stutter. Some frequently utilized feature sets include Mel-Frequency Cepstral Coefficients (MFCCs), Linear Predictive Coding (LPC), and Linear Predictive Cepstral

Manuscript received Month Day, Year; revised Month Day, Year. Corresponding author: Qianheng Xu (email: 26xuq@millburn.org).
[1]Millburn High School, Millburn, NJ 07041 USA.

Stuttering Type	Features Used	Total	Removed	Retained	Accuracy
Repetition	MFCC	60	54	6	90.0%
	LPC	60	52	8	86.7%
Prolongation	MFCC	70	67	3	95.7%
	LPC	70	65	5	92.9%

TABLE II: Performance of DSP-based stutter removal methods using MFCC and LPC features across two stuttering types [10].

Coefficients (LPCCs) [9]. For instance, MFCC features are generated by first applying the Fast Fourier Transform (FFT) to convert an audio sample from the amplitude domain to the frequency domain, generating the power spectrum. After that, the Mel filter bank is used to map the power spectrum to Mel frequencies, employing a set of nonlinear triangular filters. This aligns the intensity of frequencies to match the nonlinearity of human auditory perception. Finally, a Discrete Cosine Transform (DCT) is used to generate the cepstral coefficients through the decorrelation of features. This pipeline is applied to every window of audio, usually 20–50 ms long [9] [10].

K. N. et al. introduced a rule-based approach for stutter detection and removal by computing a "correlation factor" between adjacent audio windows using either MFCC or LPC features [10]. A high correlation value, empirically determined to be 0.92, was used to identify repeated or prolonged segments, prompting the deletion of redundant frames. The unusually high correlation factor was used to detect repeated audio patterns, such as recurring phonemes or extended silences. However, this approach was evaluated exclusively on repetition and prolongation stutters, without accounting for other common disfluencies such as blocks or interjections. Additionally, the experimental scope of the study was confined to only 60 repetition and 70 prolongation events obtained from a limited selection of audio files. As a result, its generalizability across diverse speakers, accents, or spontaneous conversational settings remains uncertain [10]. Table II summarizes the performance, demonstrating high within-sample accuracy.

In some cases, low-level features such as MFCC, LPC, and LPCC are extracted from the audio and used as input to neural networks or machine learning models, which then classify whether a stutter has occurred at each point in time.

For example, StutterNet, introduced by Sheikh et al., is 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. While StutterNet was only able to detect stutters, it is plausible that similar systems could be used to flag sections of stuttered audio for removal [11]. Table III summarizes representative DSP-based systems, their methodological choices, and reported performance.

2) ASR & TTS-based Approaches: The second approach to stutter correction involves fine-tuning or training an ASR model on stuttered speech such that the ASR model can generate transcripts of that speech. The ASR model can either explicitly transcribe the stutter into text or ignore the stuttered portions, producing a fluent transcript. That transcript can be filtered with rule-based systems and finally passed through a TTS model to produce fluent audio sequences. Figure 1 shows the general pipeline to achieve ASR & TTS-based stutter correction. This approach helps to omit artifacts caused by splicing audio in the DSP-based approaches, since TTS models are generating new audio sequences. However, DSP methods can preserve speaker prosody more naturally, as they simply edit the original speech. For a TTS model to mimic the tone and prosody of the original speaker, it would need more utterances to fine-tune on.

Mujtaba & Mahapatra describe fine-tuning Whisper-small, a state-of-the-art ASR model pretrained on fluent speech corpora. To allow for efficient training, low-rank adaptation (LoRA) was used as a form of parameter-efficient fine-tuning. Whisper-small was fine-tuned on ground truth transcripts provided by the FluencyBank and private HeardAI datasets. They reported that Whisper-Small achieved a WER of 33.88% without any training or fine-tuning. After fine-tuning Whisper-Small using the aforementioned LoRA methods, it achieved a WER of 9.39% [12].

Due to the adaptability of fine-tuned ASR models, there has been research into fine-tuning pre-trained ASR models for other speech impairments such as dysarthria. Wang et al. fine-tuned wav2vec2.0 and HuBERT ASR models on dysarthria datasets. They achieved the best WER of 16.53% by fine-tuning a pre-trained wav2vec2.0 ASR model on a dysarthria corpus augmented by a generative adversarial network. Even without any data augmentation, they obtained a WER of 22.25% on a fine-tuned wav2vec2.0 model and a WER of 21.10% on a fine-tuned HuBERT model [13].

3) Deep Learning & End-to-end Approaches: Deep learning approaches for speech recognition and speech conversion have been the subject of extensive investigation. Speech conversion focuses on modifying or transforming a speech signal, such as changing the speaker's voice or style, while preserving the original linguistic content. In contrast, speech recognition involves accurately transcribing spoken language into text [14] [15] [16]. Deep learning approaches for speech processing typically involve training neural networks or other computational models, eliminating the need for manual feature engineering. Recent advancements in deep learning have popularized end-to-end models, systems that learn to perform the entire correction process directly from input data without requiring handcrafted features or intermediate steps.

While there is considerable work on end-to-end speech conversion of fluent speech and other speech impairments such as dysarthria, there is little research on deep learning or end-to-end stutter correction. This paper first reviews deep learning and end-to-end methods for fluent speech recognition, conversion, and the correction of dysarthric speech.

The popularity of end-to-end models stems from their ability to consolidate the entire training and inference pipeline into a single model optimized with one objective function that directly reflects the training goal. In contrast, a traditional DSP-based pipeline might first extract MFCC features from audio and then train a neural network to classify whether each frame contains a stutter. The neural network in this case is trained for frame-level accuracy in detecting stutters, which

Author(s)	Dataset	Feature Extraction	Detection Method	Stutter Type(s)	Best Accuracy
Mishra et al., 2021	UCLASS	MFCC, RMSE	Deep neural network	Repetition, Prolongation, Block	86.67%
Dash et al., 2018	Private human speech samples	Amplitude	Deep neural network	Prolongation	86%
Shonibare et al., 2022	Private human speech samples	log-Mel Spectrograms	CNN	Repetition, Block, Prolongation	71.24% reduction in WER

TABLE III: Summary of studies on stutter detection, including datasets, feature extraction methods, detection approaches, stutter types analyzed, and best reported accuracies [10].

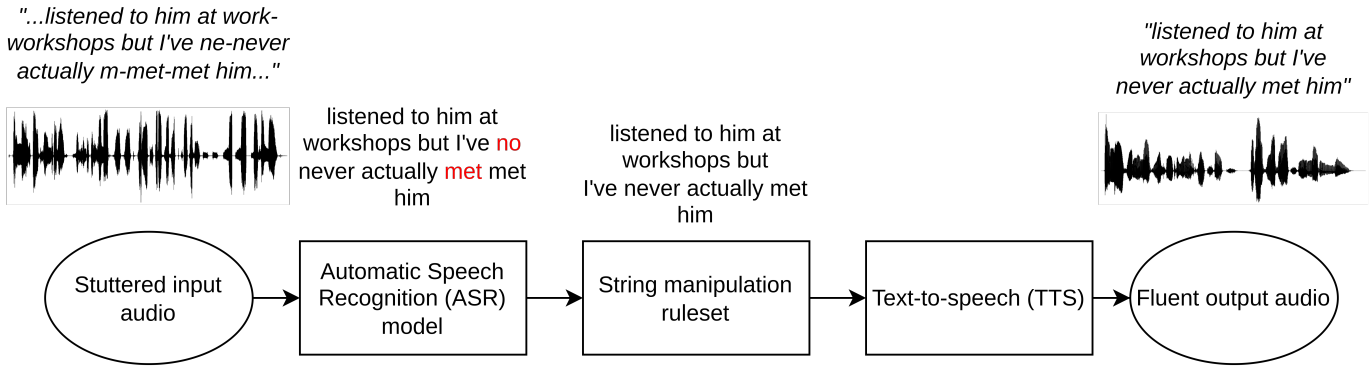


Fig. 1: General pipeline of an ASR-TTS approach.

does not align with removing stutters at the speech sequence level [17] [18]. The lack of manual feature engineering in end-to-end models also allows for greater generalizability and adaptability to similar problems [19]. Finally, end-to-end models such as encoder-decoder and RNN-Transducer models are suitable for sequence-to-sequence tasks such as speech conversion, as they do not require alignment of acoustic sequences to ground truth transcripts and are very flexible with input and output sequence lengths [16].

For example, Toshniwal et al. developed an attention-based encoder-decoder model inspired by recurrent neural networks [20]. A speech encoder is built on a deep bidirectional long short-term memory (BiLSTM) network, which produces a sequence of abstract hidden representations. These hidden representations are passed into the character decoder, which is a single-layer LSTM that predicts the most likely letter uttered [20].

Wang [21] introduced an end-to-end encoder-decoder for dysarthric speech correction. Three multitask encoders were used: a content encoder to learn underlying semantic meaning, a prosody encoder to learn and correct audio features salient to dysarthria, and a speaker encoder to capture prosody and recreate the tone and voice of the original speaker. A decoder aggregates hidden representations from all three layers and generates acoustic features from which audio can be reconstructed using a vocoder [21].

Despite these advances, work on stuttering remains limited. Most deep learning research to date has focused on event-level detection and classification of disfluencies rather than the synthesis of corrected audio. Approaches that combine automatic speech recognition with text-to-speech generation can yield fluent renditions, but they are constrained by a lack of transcription accuracy or generalization across all categories of stuttering. Direct acoustic-to-acoustic correction of stuttered speech into fluent speech remains an underexplored problem. This gap motivates the present study, which introduces two end-to-end models, *StutterZero* and *StutterFormer*, that jointly address disfluency detection, transcription, and fluency restoration. By explicitly targeting corrected audio generation while preserving semantic content, these models extend prior deep learning approaches toward real-time stutter correction and inclusive speech technology.

II. Methodology

This section presents the methodological framework of the study, which is organized into four main parts. First, the datasets used for training and evaluation are described, along with the corresponding data preprocessing and cleaning procedures. Second, the ASR–TTS baseline pipeline is outlined, in which a pretrained ASR system is adapted to stuttered speech and fluent output is resynthesized using a TTS model. Third, two proposed end-to-end models, StutterZero and StutterFormer, are introduced; these models directly transform disfluent speech into fluent speech through multitask encoder–decoder architectures. Finally, the training configuration, optimization procedures, and cross-validation strategy employed to ensure robustness and reproducibility are detailed.

A. Data

This research uses two datasets for training: Stuttering Events in Podcasts (SEP-28K) and LibriStutter. The SEP-28K dataset contains 23 hours of naturally occurring stuttering events, divided into 28,000 clips, each lasting 3 seconds [8]. While this dataset does not contain any ground truth for what the fluent speech should be, it includes labels classifying every stutter into the categories shown in Table I. All audio

recordings were taken from public podcasts with people who stutter at a standard sampling rate of 16 kHz. After all cleaning and processing, about 14 hours of raw audio data were left.

About 20 hours of artificially produced stuttered audio are included in the LibriStutter dataset. This corpus was created by splicing, cutting, duplicating, and performing other manipulations on the fluent LibriSpeech corpus to mimic the prosodic characteristics of stuttering [2]. SEP-28K does not include fluent reference transcripts, whereas LibriStutter contains paired fluent transcriptions that facilitate supervised training.

1) Data Cleaning and Preprocessing: To ensure that all audio contained speech, audio files labeled "Music" or "NoSpeech" were removed from the SEP-28K dataset. To account for bias or skew in the frequency of each type of stutter, data resampling was conducted on SEP-28K to balance out stutter categories that may have been less common.

After cleaning, approximately 34 hours of stuttered audio were retained across the SEP-28K and LibriStutter datasets, with no overlap in speakers or transcripts between the two corpora.

The University College London Archive of Stuttered Speech (UCLASS) and the FluencyBank dataset are the two other well-known stuttering datasets [22] [23]. However, both datasets have some limitations. The UCLASS dataset does not provide fluent "ground truth" transcripts for the stuttered speech, so there are no reference transcripts to fine-tune Whisper-Small on.

The FluencyBank dataset contains the Voices-AdultsWhoStutter (Voices-AWS) corpus, which was de-identified and made publicly available to researchers who created a free account. StutterZero and StutterFormer were later tested on the Voices-AWS subset of the FluencyBank dataset for validation purposes.

2) Data Splitting and Cross Validation for Training: To train and validate all three approaches (ASR-TTS, StutterZero, and StutterFormer) introduced in this paper, the combined dataset of audio-transcript pairs from SEP-28K and LibriStutter was randomly sampled and split into training (80%), test (10%), and validation (10%) sets. The training and testing splits were used in a five-fold cross-validation setup, with the validation set being used for a fair comparison between all three approaches after training. All training for this study was conducted on an NVIDIA RTX 3080 with 10 gigabytes of VRAM.

B. ASR-based approach

To obtain fluent speech data to train StutterZero and StutterFormer as end-to-end models, this research also developed an auxiliary pipeline as described below to first generate fluent speech data, effectively "completing" both datasets:

1) Stuttered audio and fluent transcripts from LibriStutter were used to fine-tune a Whisper-Small ASR model that was originally trained on fluent speech.
2) The fine-tuned Whisper-Small model was applied to the SEP-28K dataset, generating fluent transcripts for all SEP-28K audio clips.

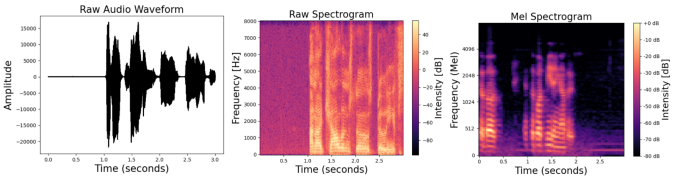


Fig. 2: Conversion from waveform to log-Mel spectrogram. The waveform (left) shows amplitude over time. After STFT, the spectrogram (middle) represents frequency over time. Applying the Mel filterbank produces a log-Mel spectrogram (right), aligned with human auditory perception.

3) A pretrained MeloTTS text-to-speech model was applied to all fluent transcripts from both datasets, generating accurate and clear fluent audio sample counterparts.

C. Whisper-Small Architecture

This study fine-tuned Whisper-Small, a lightweight derivative of the Whisper family of ASR models. Audio data from both datasets are in the amplitude domain, though Whisper-Small accepts a spectrogram in the frequency domain as input [24].

Whisper-Small's WhisperFeatureExtractor uses a Short-Time Fourier Transform (STFT), computing the Fast Fourier Transform (FFT) on overlapping segments, or "windows," of the audio as it slides across the entire signal. Then, 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 does not perceive pitch in a linear manner; humans are more attuned to changes in lower pitches than in higher pitches. 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. Once a spectrogram is transformed into a log-Mel spectrogram, equal intervals on the Mel scale reflect equal perceived differences in pitch. The practical effect is that Mel spectrograms highlight the frequencies of human speech while minimizing the intensity of background noise, allowing the model to concentrate solely on the speech signals [25]. Figure 2 displays visualized spectrograms of the conversion process used to obtain a log-Mel spectrogram.

Whisper-Small computes log-Mel spectrograms with 25-ms-long windows that move forward by 10 milliseconds at every time step.

Whisper-Small is a Transformer encoder-decoder model featuring 12 layers in both the encoder and decoder, a hidden width of 768 units, and 12 attention heads, amounting to around 244 million parameters in total. The encoder layers consist of a self-attention mechanism and fully connected hidden layers. The encoder generates a context vector, which is passed via cross-attention to each of the 12 decoder attention blocks. Each decoder layer contains self-attention, cross-attention, and a fully connected network [24]. At every time step, the autoregressive decoder is fed the aggregated output tokens from all previous time steps so that it can predict the most likely following token.

Whisper's built-in byte-pair encoding (BPE) tokenizer converts the numeric predictions of Whisper, called tokens, back into readable text.

D. Whisper-Small Fine-tuning

To fine-tune Whisper-Small, this study aggregated all stuttered audio samples and normalized the sampling rate to 16 kHz. For LibriStutter audio clips, longer audio samples are truncated after 30 seconds.

Fine-tuning begins with Whisper-Small's pretrained weights to take advantage of the accurate general transcription properties that Whisper is already trained for. The evaluation metric is Word Error Rate (WER), which is formally defined in Equation (1) [26].

$$\text{WER} = \frac{\text{substitutions} + \text{deletions} + \text{insertions}}{\text{number of words in the reference}} \quad (1)$$

S is the number of substitutions (words in the reference that are replaced with incorrect words in the hypothesis), D is the number of deletions (words in the ground truth that are missing from the predicted speech), and I is the number of insertions (extra words in the predicted speech that are not in the ground truth).

Training runs for a maximum of 10,000 steps with a learning rate of 1e-5 and a batch size of 8. Gradient accumulation over two steps is used to simulate a larger batch size. Gradient checkpointing and mixed precision are enabled to save memory and speed up training. Evaluation occurs every 1,000 steps, and the model weights with the lowest WER are saved.

E. StutterZero Model Architecture

End-to-end models have seen significant usage in other speech-based tasks such as translation and transcription. These models are characterized by directly converting an input signal to an output signal without any intermediate feature engineering or representation.

This study introduces an autoregressive, end-to-end, multitask model named StutterZero. Inputs consist of 80-channel log-Mel spectrograms computed with a 50 ms window length and a 12.5 ms frame shift. The encoder converts the input log-Mel spectrogram into a higher-level representation called a context vector. The context vector is a numerical representation of features that the trained encoder determines to be relevant. While typical encoder-decoder models use one encoder and one decoder, this study proposes a multitask decoder in which two decoders are forced to predict different data types. During training, a multitask model minimizes a joint loss function, forcing the decoders to learn more generalized patterns that benefit all constituent losses [27]. The spectrogram decoder predicts the fluent spectrogram signal, while the transcript decoder predicts the grapheme being uttered.

The spectrogram decoder generates the spectrogram one frame at a time, using the context vector along with the previously predicted frames as contextual input to append each new predicted frame to the output spectrogram. Similarly, the transcript decoder uses previously predicted tokens to predict the following grapheme tokens. Figure 3 displays an overview of the model architecture.

1) StutterZero Encoder: The encoder consists of two convolutional blocks, each including a two-dimensional convolution layer with a 3×3 kernel, a 2×2 stride, and batch normalization. To effectively capture both spatial and temporal features of the input sequence, a convolution-augmented bidirectional long short-term memory network (ConvBiLSTM) is employed. Log-Mel spectrograms provide information about both the frequency content of a signal and the timing of its occurrence. Unlike traditional LSTMs, the ConvBiLSTM replaces standard matrix multiplication with a convolution operation, which promotes the learning of local spatial patterns in two-dimensional data. The network includes mechanisms analogous to the standard LSTM gates—input, forget, and output gates—as well as candidate cell states and hidden states. Each gate and state use convolutional kernels and trainable biases to process the current input in combination with the hidden state from the previous time step, enabling the model to capture complex spatiotemporal dependencies [28].

The encoder generates a 512-dimensional context vector, which is used as input for both decoders.

2) StutterZero Spectrogram Decoder: The spectrogram decoder uses the previously predicted spectrogram and the current context vector as inputs. Two fully connected layers form a pre-net that compress and transform the previous spectrogram frames into a lower-dimensional representation. A well-trained pre-net simplifies the input and extracts the most salient features. If the raw spectrogram frames were used, the decoder might "shortcut" the learning process by copying the previous frame or minimally modifying it [29] [30].

StutterZero employs a location-based attention mechanism, which is an extension of classical content-based attention. In content-based attention, the model computes an attention score between a query (representing the decoder's current state) and a key (representing the encoder's output at any time step). The score measures how relevant each encoder state is to the current decoding step. The attention weights are then obtained by normalizing these scores using a softmax function. These weights form a probability distribution that determines how much "attention" the decoder should pay to each encoder state when predicting the next output. Finally, a weighted context vector is used for output prediction [31].

Because speech data are only read in one direction (monotonically), classical content-based attention may "jump around" erratically without respecting the flow of time. Location-based attention also computes a set of features using concatenated previously calculated attention weights via a one-dimensional convolution.

The location-based features are included in the attention score along with another set of trainable weights. In this way, the decoder is informed by past attention behavior and adjusts its focus accordingly to maintain a monotonic flow of data.

Once the spectrogram decoder predicts a fluent spectrogram signal, the Griffin–Lim algorithm is used to reconstruct the phase data and generate an audio signal in the amplitude domain [32].

3) StutterZero Transcript Decoder: Previous work in text-to-speech and fluent speech conversion models has demonstrated the efficacy of multitask decoder architectures. Mul-

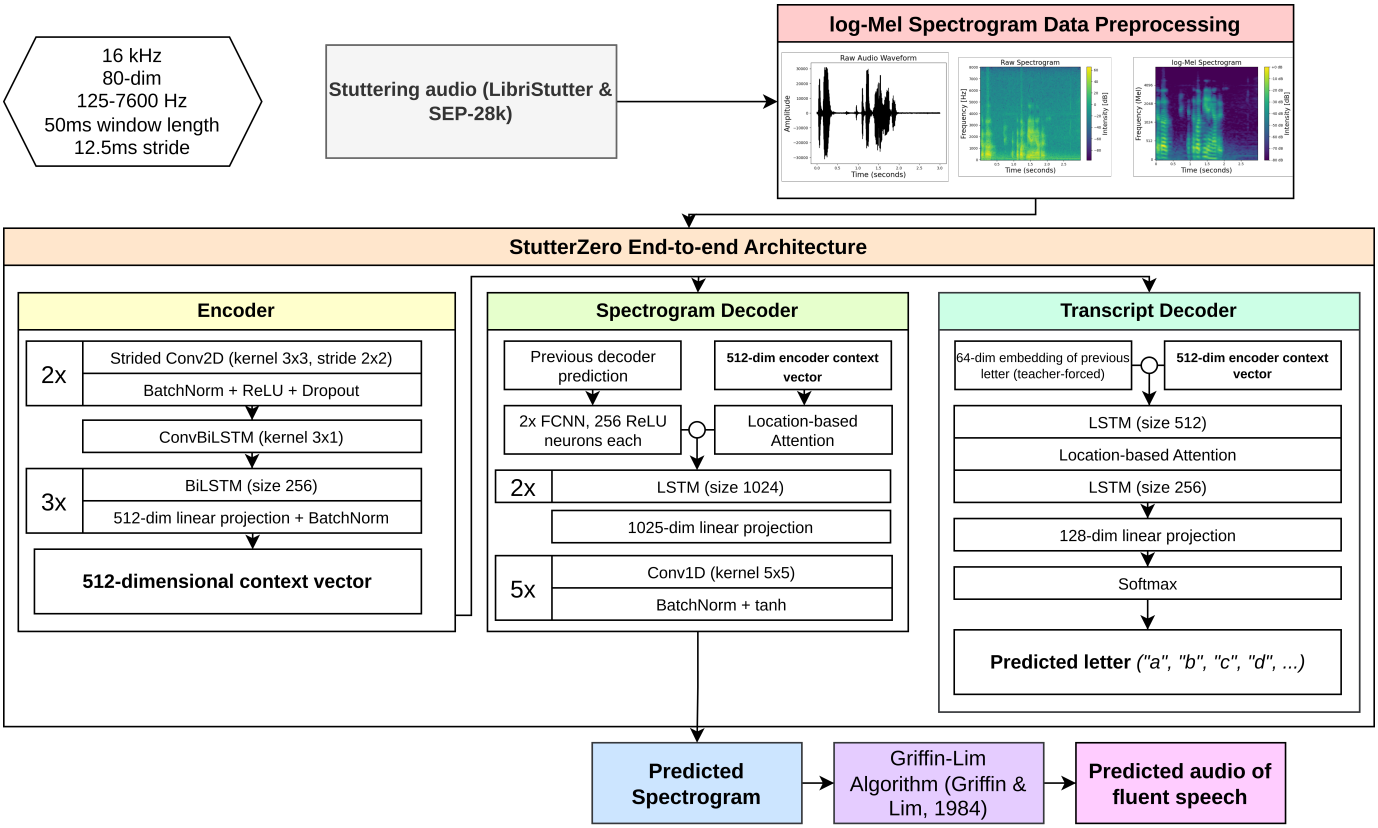


Fig. 3: Overview of the StutterZero end-to-end architecture. A convolutional BiLSTM encoder compresses these features into a 512-dimensional context vector, which is shared between two multitask decoders. The spectrogram decoder predicts fluent spectrogram frames that are reconstructed into audio using the Griffin–Lim algorithm, while the transcript decoder predicts grapheme tokens.

titask training sums the loss for both the spectrogram and transcript decoders, creating a joint loss function that forces the training process to optimize the loss on both decoders [24] [20] [27].

In the case of the transcript decoder, adding an objective function at the grapheme level may allow StutterZero to learn more intricate orthographic features of a word to correctly distinguish between allophones and homophones.

The transcript decoder also uses a teacher-forced embedding of the previous text as its input during training. Instead of using its previous prediction as input for generating the next token, the true ground-truth token from the training data is fed as input to the next step. This stabilizes and speeds up training because the model always conditions on the correct previous tokens. Additionally, it avoids extreme divergence and error in the early stages of training, when incorrect predictions may accumulate.

4) StutterZero Training: The spectrogram decoder uses mean squared error (MSE) loss, where the model minimizes the difference between the ground-truth spectrogram and the predicted fluent spectrogram across all frequency bins and time steps [33], [34].

Cross-entropy loss is used for the transcript decoder because grapheme prediction is a categorical task. This loss measures how well the model predicts the correct token given all previous tokens and the input log-Mel spectrogram.

To combine the loss functions for both decoders, a bespoke loss function (described in Equation 2) is defined by summing two components: the mean-squared error (MSE) loss between the fluent and stuttered spectrogram frequency bins, and the cross-entropy loss between the predicted token probability distribution and the ground-truth tokens.

$$L_{\text{total}} = L_{\text{MSE}} + L_{\text{CE}} \tag{2}$$

StutterZero used standard stochastic gradient descent with the Adam optimizer for training, starting with a learning rate of $1e^{-4}$ and a weight decay of $1e^{-6}$. The weight decay discourages large weights by incorporating the L2 norm of the weights into the loss function. Table IV shows all the hyperparameters used in the Adam optimizer.

Parameter	Value
Learning Rate	$1e^{-4}$
Weight Decay	$1e^{-6}$
Batch Size	3
Betas (Momentum parameters)	$(0.9, 0.999)$
Epsilon	$1e^{-6}$

TABLE IV: Training hyperparameters used for the StutterZero Adam optimizer.

F. StutterFormer Model Architecture

StutterFormer maintains the same multitask architecture as StutterZero, but switches out internal layers for the Attention mechanism found in Transformers as described in [35]. Figure 4 displays a high-level summary of StutterFormer's architecture.

1) Multi-Head Attention: Multi-head attention serves as the fundamental building blocks of StutterFormer. Compared to single-head attention, which computes attention in a single attention distribution, multi-head attention projects queries, keys, and values into multiple subspaces, applies attention in parallel, and then recombines the results. This makes it possible for the model to take a joint attention function to data from several learned subspaces. For instance, one head may focus on short-range syntactic dependencies only while another head tracks long-range semantic connections. This increase in flexibility allows multi-head attention to learn a greater breadth of information while keeping the parameter count relatively equal to a larger single-head attention module [35].

2) StutterFormer Encoder: The encoder input remains a log-Mel spectrogram. Because a Transformer processes all input frames in parallel, it does not have a sense of order by default. Sinusoidal positional encoding gives each time step a deterministic vector (based on the position index) for the embedding at each time step. Residual and layer normalization layers are used between the multi-head attention and feed-forward layers to mitigate vanishing or exploding gradients [36] [37]. Three multi-head attention units are utilized in the encoder, each consisting of four heads. Similarly to StutterZero, the encoder outputs a context vector – a learned hidden representation of the input.

3) StutterFormer Decoders: The spectrogram decoder uses the context vector from the input and uses a similar pre-net architecture as StutterZero to learn a lower-dimension representation. Masked multi-head self-attention is used to prevent the decoder from "looking ahead" at future frames that have not been predicted yet. After applying the mask, the decoder can only attend to itself and past frames. Cross attention utilizes queries from the preceding decoder layer, while the keys and values are derived from the original encoder context vector. This allows the decoder to observe the entire encoder context when deciding the next frame.

Finally, a post-net consisting of five 1-dimensional convolutions refines the predicted mel spectrogram to denoise and sharpen the final audio signal [29] [38].

The transcript decoder uses a very similar architecture to the spectrogram decoder and employs the same Transformer decoder architecture.

4) StutterFormer Training: Like StutterZero, StutterFormer employs a hybrid loss function that combines multiple weighted losses. The spectrogram decoder also uses MSE loss between the predicted and ground truth spectrograms. In addition to MSE loss, the spectrogram decoder also computes a mean absolute error (MAE) loss. MAE loss measures the absolute difference of the predicted values and target values without squaring. When tested with spectrogram applications, MAE loss promotes sharper spectrograms that prevent the predicted fluent speech from sounding slurred [39]. The transcript decoder also uses cross-entropy loss.

StutterFormer is trained on a cosine annealing with warm restarts scheduler. This approach periodically "restarts" the learning rate to help escape local minima and explore the loss landscape more effectively [40] [41] [42]. Like StutterZero, StutterFormer was trained for 1000 epochs.

Parameter	**Value**
Learning Rate	$1e^{-4}$
Weight Decay	$1e^{-5}$
Batch Size	3
Betas (Momentum parameters)	$(0.9, 0.98)$
Epsilon	$1e^{-6}$
T_0 (Initial restart epochs)	50
T_{mult} (Period multiplication factor)	2

TABLE V: Training hyperparameters used for the cosine annealing scheduler and optimizer.

III. Results

A. Benchmarking

In addition to the ASR-based and end-to-end approaches used in this study, state-of-the-art fluent-speech ASR models were also evaluated to establish a baseline for how accurately current speech recognition systems transcribe stuttered speech. This study used the Whisper-Tiny, Whisper-Small, and Whisper-Medium pretrained fluent ASR models as baselines [24]. Any larger models such as Whisper-Large could not be tested due to memory and hardware limitations. The 10% validation data split was used to assess each of the six models.

Three metrics were used to evaluate all models: Word Error Rate (WER), Character Error Rate (CER), and BERTScore. CER is defined in Equation (3) as the proportion of incorrectly predicted characters compared to the ground truth string. To assess the semantic similarity between the ground truth and predicted utterances, a BERTScore is calculated using pre-trained contextual embeddings from the Bidirectional Encoder Representations from Transformers (BERT) language model [43]. Cosine similarity is used as a metric to quantify similarity between high-dimensional embedding vectors. The BERTScore defines how similar the two strings are semantically, meaning it is more forgiving towards minor transcription mistakes that still preserve the overall meaning of the utterance [43].

$$\text{CER} = \frac{\text{char. substitutions} + \text{char. deletions} + \text{char. insertions}}{\text{number of characters in the reference}} \tag{3}$$

Because both approaches in this research produce audio signals, the audio signals must be converted back to text before any word or character-level evaluation. This study used an unmodified, pretrained copy of the Whisper-Small ASR model to transcribe the predicted fluent sequences from both the ASR and end-to-end approaches, as shown in Figure 5. This was an attempt to simulate what an untrained, average English-speaking person would interpret the fluent speech to be. The

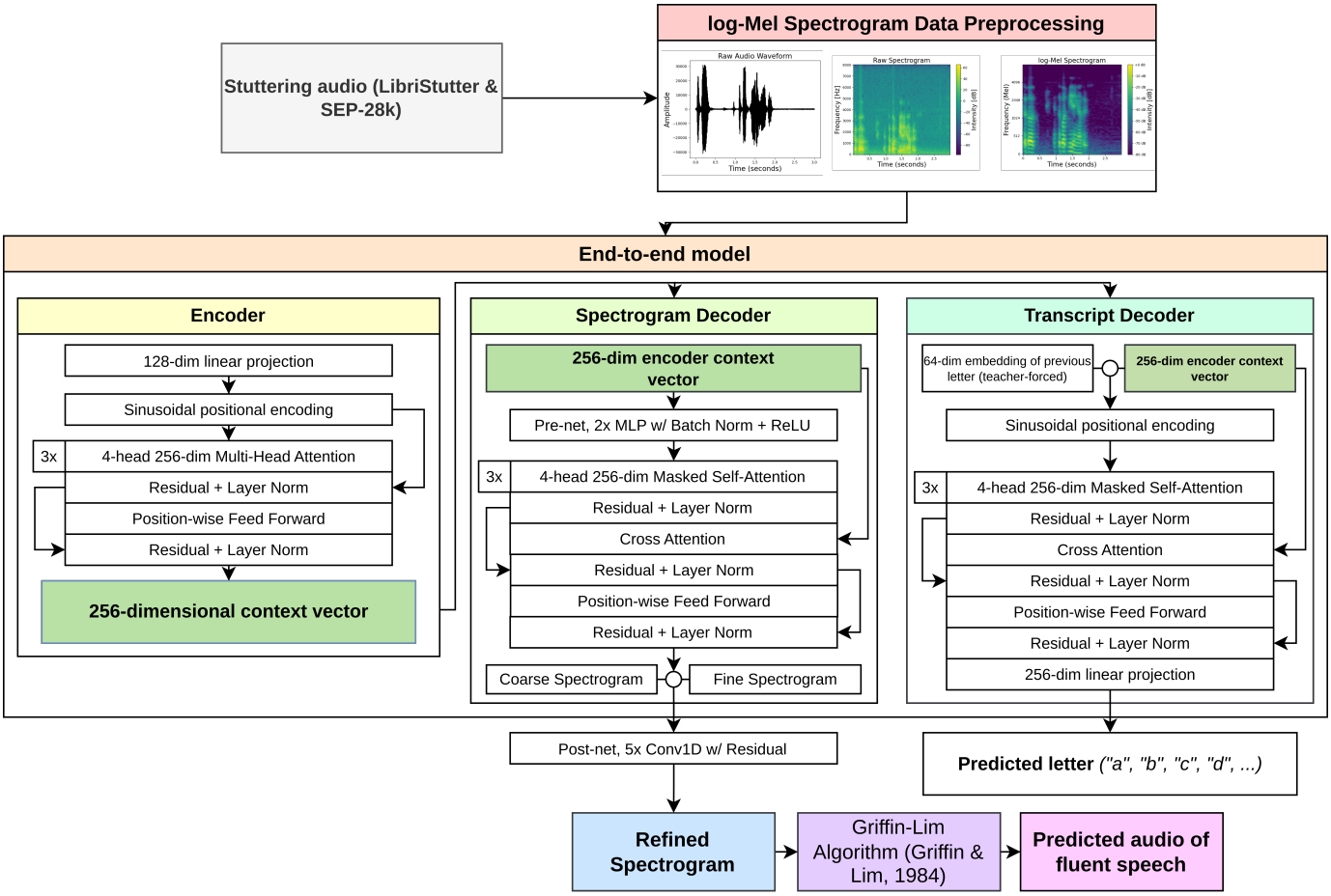


Fig. 4: Overview of the StutterFormer end-to-end model. The encoder projects features into a 128-dimensional space, applies sinusoidal positional encoding, and processes them through stacked multi-head attention and feed-forward layers to produce a 256-dimensional context vector. This vector is shared between two multitask decoders. The spectrogram decoder refines fluent spectrograms via masked self-attention, cross attention, and convolutional post-nets, while the transcript decoder predicts grapheme tokens using masked self-attention and cross attention layers. The Griffin–Lim algorithm reconstructs waveforms from refined spectrograms, jointly optimizing fluency restoration and transcription.

predicted transcripts of the fluent speech were compared with ground truth transcripts to calculate the final metrics. Table VI displays the mean WER, CER, BERTScore Precision, and their standard deviations in the validation data split.

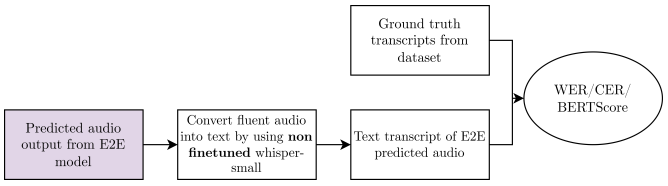


Fig. 5: The validation pipeline. An untouched Whisper-Small ASR model acted as ”judge” to evaluate speech clarity and intelligibility.

Table VI shows that all three proposed models—ASR-based, StutterZero, and StutterFormer—outperform the best Whisper baseline (Whisper-Medium) across both WER and CER metrics. The Whisper-Medium model achieves a WER of 0.361 and a CER of 0.162, while the ASR-based approach dramatically reduces these errors to 0.04 and 0.02, respectively. This indicates a substantial improvement in transcription accuracy and character-level precision.

B. Statistical Significance

The non-parametric, two-sided Wilcoxon Signed-Rank Test is used to assess the statistical significance of the improvements achieved by this research compared to the baseline Whisper performance. This test is chosen due to its widespread use and established precedent in evaluating significant differences between models [44]. Comparing the performance of the ASR-based approach with Whisper-Medium (the best performing Whisper model), the Wilcoxson Test returns a test statistic of 77631.0, a p-value of $< 1e^{-100}$, significant at $\alpha = 0.05$. Running the test comparing the performance of the end-to-end StutterZero and StutterFormer models against Whisper-Medium also returns a p-value of $< 1e^{-100}$, significant at $\alpha = 0.05$. This shows that both StutterZero and StutterFormer demonstrate a significant improvement over state-of-the-art fluent speech models.

Model	Mean WER	Mean CER	Mean BERTScore Precision
Whisper-Tiny	0.415 ± 0.049	0.227 ± 0.033	0.5768 ± 0.027
Whisper-Small	0.370 ± 0.037	0.171 ± 0.027	0.5956 ± 0.017
Whisper-Medium	0.361 ± 0.032	0.162 ± 0.022	0.601 ± 0.017
ASR-based	0.04 ± 0.01	0.02 ± 0.01	0.9516 ± 0.04
StutterZero (end-to-end)	0.116 ± 0.013	0.110 ± 0.052	0.9174 ± 0.034
StutterFormer (end-to-end)	0.08 ± 0.03	0.07 ± 0.011	0.9411 ± 0.012

TABLE VI: Combined WER, CER, and Mean BERTScore Precision metrics for all models.

C. Ablation Study

An ablation study was conducted to determine the impact of the multitask architecture and transcript decoder. StutterZero and StutterFormer were re-trained across a five-fold cross-validation with all hyperparameters, data splits, and other tunable values kept constant. However, the transcript decoder was removed from both models during the ablation. This significant difference in every metric after the ablation in both models demonstrates the importance of the transcript decoder. Observing StutterZero, the WER increased by 22.3% whilst the CER only increased by 15%. This shows the ablated StutterZero predicted most of the characters in a word correctly, but perhaps missed a few characters in some more words. The same phenomenon is seen in StutterFormer ablation, but to a lesser degree. This aligns with the functionality of the transcript decoder – to help both end-to-end models capture more detailed orthographical features in words. A greater increase in WER than in CER could mean StutterZero/StutterFormer incorrectly predicted more allophones and homophones, which have similar character-level spellings but are different words.

D. FluencyBank Validation Test

FluencyBank is a subset of the TalkBank corpus, an open repository for spoken language data. FluencyBank specifically focuses on disfluencies, including stuttering. Because the FluencyBank dataset is password-protected and it is not possible to automate the scraping of data from the website, this research manually downloaded audio clips from a randomly selected recording along with their transcripts from the VoiceAdultsWhoStutter (Voices-AWS) subset. After downloading and splitting each audio file to be 30 seconds or less, there were 800 stuttered audio samples.

Because FluencyBank is an entirely new dataset with unique audio characteristics, StutterZero and StutterFormer are tested on data they have never encountered before. This ensures that no bias from the training data influences the results. Table VIII shows the WER and BERTScore metrics after testing StutterZero and StuttFormer on samples of the FluencyBank dataset.

Audio ID	SZ WER	SF WER	SZ BERT	SF BERT
Mean	0.161 ± 0.034	0.120 ± 0.013	0.915 ± 0.015	0.937 ± 0.023

TABLE VIII: Performance comparison of StutterZero (SZ) and StutterFormer (SF) on the FluencyBank dataset using WER and BERTScore Precision metrics.

Table IX presents a qualitative comparison between the stuttered and fluent spectrograms for three speech samples from the FluencyBank dataset. The regions enclosed by green rectangles highlight sections containing a stutter in the original audio or the corresponding fluent segments after correction.

The effects of stutter correction are most apparent in the first row (audio 29mb_1.wav). In the stuttered spectrogram, the doubled upward-sloping signal represents a word repetition ("can you–can you"). Both StutterZero and StutterFormer successfully remove this repetition in their predicted fluent spectrograms, producing a single, continuous signal in place of the duplicated pattern.

IV. Discussion

This work introduces three stutter correction systems, including the first two end-to-end encoder–decoder models, and provides evidence that direct conversion of stuttered to fluent audio is both feasible and effective.

StutterZero and StutterFormer significantly outperformed state-of-the-art fluent speech recognition models in WER, CER, and BERTScore. The Wilcoxon Signed-Rank Test demonstrated the improvements ($p < 0.05$) of StutterZero and StutterFormer over the next best model, Whisper-Medium. This demonstrates the effectiveness of multitask, end-to-end models for stuttered speech recognition.

Unlike existing stutter correction approaches, which struggle to address all five stuttering types, StutterFormer achieves an average transcription accuracy of 90% on the combined SEP-28K and LibriStutter validation set. It maintains robust performance on entirely new data with different speakers and recording conditions, achieving 88% accuracy. These results highlight StutterFormer's resilience to variations in audio quality and speaker prosody, enabling consistently high performance across diverse datasets.

StutterZero is not far behind, with an 88% transcription accuracy on the combined validation set and 84% on the FluencyBank subset. The slight decrease in accuracy may be attributed to the Transformer architecture's efficiency in modeling speech and audio sequences. Indeed, correcting stuttered speech is a sequence-to-sequence task, similar to machine translation and other language tasks where Transformers have shown exceptional performance [31].

The observed performance improvements resonate with trends in related domains, such as dysarthric and accented speech recognition, where multitask and end-to-end approaches have consistently demonstrated robustness to atypical input [12], [21]. By showing that stutter correction benefits

Model	WER	CER	Mean BERTScore Precision
StutterZero (with transcript decoder)	0.116 ± 0.013	0.110 ± 0.052	0.9174 ± 0.034
Ablated StutterZero (without transcript decoder)	0.339 ± 0.011	0.260 ± 0.012	0.437 ± 0.010
StutterFormer (with transcript decoder)	0.08 ± 0.03	0.07 ± 0.011	0.9411 ± 0.012
Ablated StutterFormer (without transcript decoder)	0.221 ± 0.016	0.194 ± 0.019	0.552 ± 0.015

TABLE VII: Ablation Study Results on Validation Data.

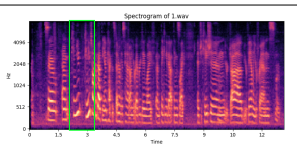
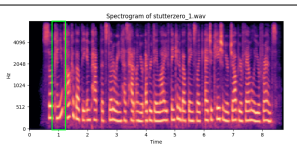
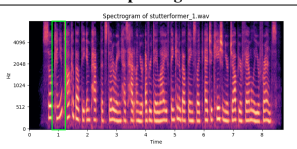
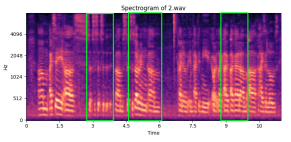
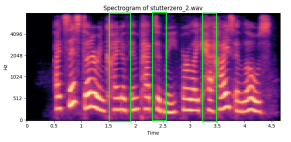
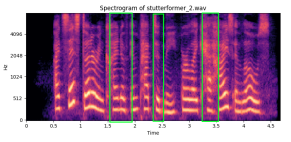
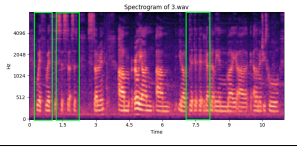
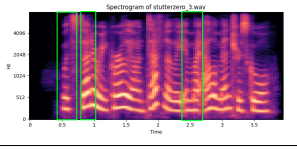
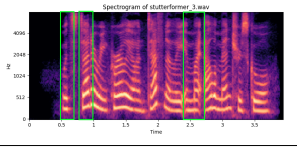
ID	Transcript	Stuttered Spectrogram	StutterZero Predicted Fluent Spectrogram	StutterFormer Predicted Fluent Spectrogram
29mb_1.wav	so can you can you talk about the impact that stuttering has had on your everyday life thinking about things like education your job your family your friends			
29mb_2.wav	i'd say s-s-s uhm s-s-s-stuttering kind of impacted me or like I-I-I-I had highs and lows			
29mb_3.wav	with with s-s-stuttering early on d-d-d-definitely in my elementary school			

TABLE IX: Spectrograms of stuttered audio, StutterZero-predicted fluent audio, and StutterFormer-predicted fluent audio. Areas enclosed by green rectangles indicate either a stutter or the corresponding fluent region after correction.

strongly from the integration of textual and acoustic objectives, our findings suggest that linguistic supervision is not merely auxiliary but foundational. This echoes psycholinguistic accounts that disfluency cannot be treated as random noise but reflects systematic deviations in speech planning and motor execution. In this sense, our work challenges the assumption – common in early DSP and ASR pipelines – that acoustic correction can be divorced from lexical anchoring.

The superior performance of StutterFormer over StutterZero cannot be attributed solely to increased model capacity. Rather, Transformer attention provides specific advantages for disfluency correction. Multi-head self-attention captures long-range dependencies, allowing the model to flexibly relate repeated or prolonged segments of speech to their fluent counterparts. This mechanism facilitates alignment across syllables and words, which is essential when disfluencies span multiple phonemic units. In addition, attention layers can simultaneously model both global and local phonetic details, helping to preserve intonation and rhythm while removing interruptions. These properties explain why attention-based architectures are particularly well-suited for mapping stuttered to fluent speech [45].

A. Limitations

Several limitations should be noted. There remain potential areas for improvement that can be explored in future research.

1) Reliance on TTS-Generated Data: A substantial portion of training and evaluation relied on fluent audio generated by TTS systems. While this strategy enabled efficient creation of large-scale paired datasets, it introduced a prosodic mismatch between synthetic and natural speech. As a result, models may have partially learned to adapt to synthetic rhythms and intonation rather than capturing the full variability of natural stuttering. This could limit robustness when applied to spontaneous, emotionally nuanced speech. Future research should mitigate this gap by curating larger collections of natural stutter–fluent pairs, leveraging prosody encoders to capture expressive detail, and employing domain adaptation techniques to reduce distributional bias.

2) Need for Increased Dataset Diversity: The diversity of the training corpus was constrained by limited access to datasets such as UCLASS or FluencyBank. The absence of these datasets restricted coverage of various speaker demographics, accents, and tones, which in turn limits generalizability. There may be risks of overfitting and undergeneralization when training on only the SEP-28K data from the LibriStutter datasets. Even though five-fold cross-validation and testing on a completely new dataset was used to produce a candid estimate of the true performance both approaches, future steps should focus on data augmentation and training on larger datasets. Additional experiments using diverse corpora such as UCLASS or AS-70 [46] are needed.

3) Hardware and Compute Limits: The training process was constrained by significant hardware limitations. Since the model was trained on a single GPU with only 10 GB of VRAM, both the batch size and model complexity had to be reduced to make training feasible. This, however, resulted in slower training and unstable convergence of the training loss. With access to more powerful hardware, it would be possible to incorporate more advanced architectural choices – such as increasing the number of heads in the multi-head attention mechanism – potentially resulting in improved model

accuracy.

B. Future Work

While this research achieves impressive preliminary results, it also acts as a proof-of-concept, opening the doors for more advanced end-to-end models in stutter correction.

1) Prosody-aware Fine-tuning: To alleviate the prosodic loss due to training on TTS-generated fluent speech samples, using multiple encoders to capture the tonal and prosodic content of the stuttered speech could be explored. Multi-encoder models have been explored in dysarthic speech conversion, specifically using a prosody encoder to extract prosodic features [21]. This would enable fluent outputs that retain the speaker's identity, pitch, and expressive nuance. Additionally, expanding to multilingual and low-resource languages through cross-lingual pretraining and transfer learning would extend accessibility to a wider global population [46].

2) Dataset Expansion and Multilinguality: The current study is limited by access to a small number of corpora. Expanding to diverse datasets such as UCLASS, FluencyBank, and AS-70 [46], as well as developing multilingual training resources, would enhance model generalizability. Cross-lingual pretraining and transfer learning offer promising strategies to extend accessibility to low-resource and global language communities, ensuring that stutter correction technology benefits a wider population.

3) Integration with Clinical Practices: StutterZero and StutterFormer have great potential in automating and assisting with delayed auditory feedback (DAF), a common technique used in speech-language therapy. It involves recording and playing back a PWS's speech after a brief delay (usually a few milliseconds to fractions of a second) [47] [48]. Playback of the fluent speech can demonstrate how fluent speech is supposed to sound, thereby reinforcing self-monitoring and reducing disfluencies. Instead of the individual hearing their disfluent utterances delayed, the model could provide them with a fluent version of what they intended to say. This would supply the brain with consistent, fluent auditory feedback, potentially reducing the reinforcement of stuttered patterns while strengthening neural pathways associated with fluent production. Indeed, speaking in unison with a fluent signal (an external "fluent version" of one's speech content) reliably induces near-instant fluency in most PWS [49].

4) Real-Time Communication Applications: Beyond therapy, optimized versions of StutterZero and StutterFormer could be deployed in real-time communication settings. With techniques such as model quantization, pruning, and knowledge distillation, lightweight implementations may run on mobile or embedded devices. Potential applications include live correction during phone calls, video conferences, and online meetings, where disfluent speech is automatically converted to fluent audio for listeners. Similar methods could also be applied to voice recording, broadcasting, and sound engineering, eliminating the need for re-recording when disfluencies occur.

V. Conclusion

The research presents several contributions to the field of stutter correction and speech conversion. First, an ASR-based pipeline utilizing fine-tuned Whisper-Small that achieved significant performance improvements, reducing Word Error Rate to 4% compared to 36.1% for the baseline Whisper-Medium model. Second, and more importantly, this research introduced StutterZero and StutterFormer, the first two end-to-end stutter correction models that directly convert stuttered speech to fluent audio without intermediate transcription steps. StutterZero was a multitask encoder-decoder architecture using conventional convolution and LSTM layers, reducing Word Error Rate to 11%. StutterFormer was based on a modern Transformer architecture and reduced Word Error Rate even further to 8%. Being the first end-to-end models for stutter correction, both StutterZero and StutterFormer pave the way for future development of larger and more accurate end-to-end models. Specifically, the encoder-decoder architecture allows for great flexibility in the choice of encoder and decoder, allowing several multitask encoders and decoders to work in unison to learn different representations of the audio. In industry, this research may pave the way for more accessible human-machine interaction and communication.

References

[1] B. Ghai and K. Mueller, "Fluent: An AI Augmented Writing Tool for People who Stutter," in *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 1–8.

[2] T. Kourkounakis, A. Hajavi, and A. Etemad, "FluentNet: End-to-End Detection of Speech Disfluency with Deep Learning," Sep. 2020.

[3] 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," *Sensors*, vol. 23, no. 19, p. 8033, Jan. 2023.

[4] L. Iverach, S. O'Brian, M. Jones, S. Block, M. Lincoln, E. Harrison, S. Hewat, R. G. Menzies, A. Packman, and M. Onslow, "Prevalence of anxiety disorders among adults seeking speech therapy for stuttering," *Journal of Anxiety Disorders*, vol. 23, no. 7, pp. 928–934, Oct. 2009.

[5] L. Iverach, M. Jones, L. F. McLellan, H. J. Lyneham, R. G. Menzies, M. Onslow, and R. M. Rapee, "Prevalence of anxiety disorders among children who stutter," *Journal of Fluency Disorders*, vol. 49, pp. 13–28, Sep. 2016.

[6] F. Kasbi, M. Mokhlesin, M. Maddah, R. Noruzi, L. Monshizadeh, and M. Mir Mohammad Khani, "Effects of Stuttering on Quality of Life in Adults Who Stutter," *Middle East Journal of Rehabilitation and Health*, vol. 2, no. 1, Jan. 2015.

[7] O. Bloodstein, N. B. Ratner, and S. B. Brundage, *A Handbook on Stuttering*. San Diego, CA: Plural Publishing, Inc, 2021.

[8] C. Lea, Z. Huang, L. Tooley, J. Narain, D. Yee, P. Georgiou, T. D. Tran, J. P. Bigham, and L. Findlater, "From User Perceptions to Technical Improvement: Enabling People Who Stutter to Better Use Speech Recognition," Feb. 2023.

[9] 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, Dec. 2022.

[10] A. K N, K. S, K. D, P. Chanda, and S. Tripathi, "Automatic Correction of Stutter in Disfluent Speech," *Procedia Computer Science*, vol. 171, pp. 1363–1370, 2020.

[11] S. A. Sheikh, M. Sahidullah, F. Hirsch, and S. Ouni, "StutterNet: Stuttering Detection Using Time Delay Neural Network," in *2021 29th European Signal Processing Conference (EUSIPCO)*, Aug. 2021, pp. 426–430.

[12] D. Mujtaba and N. Mahapatra, "Fine-Tuning ASR for Stuttered Speech: Personalized vs. Generalized Approaches," Jun. 2025.

[13] H. Wang, Z. Jin, M. Geng, S. Hu, G. Li, T. Wang, H. Xu, and X. Liu, "Enhancing Pre-trained ASR System Fine-tuning for Dysarthric Speech Recognition using Adversarial Data Augmentation," Jan. 2024.

[14] A. Triantafyllopoulos, B. W. Schuller, G. İymen, M. Sezgin, X. He, Z. Yang, P. Tzirakis, S. Liu, S. Mertes, E. André, R. Fu, and J. Tao, "An Overview of Affective Speech Synthesis and Conversion in the Deep Learning Era," *Proceedings of the IEEE*, vol. 111, no. 10, pp. 1355–1381, Oct. 2023.

[15] T. Walczyna and Z. Piotrowski, "Overview of Voice Conversion Methods Based on Deep Learning," *Applied Sciences*, vol. 13, no. 5, p. 3100, Jan. 2023.

[16] D. Wang, X. Wang, and S. Lv, "An Overview of End-to-End Automatic Speech Recognition," *Symmetry*, vol. 11, no. 8, p. 1018, Aug. 2019.

[17] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai: IEEE, Mar. 2016, pp. 4960–4964.

[18] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," in *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Jun. 2014, pp. 1764–1772.

[19] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," Dec. 2014.

[20] S. Toshniwal, H. Tang, L. Lu, and K. Livescu, "Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition," Apr. 2017.

[21] D. Wang, "A Scalable Encoder-Decoder Framework for Disordered Speech Reconstruction," Ph.D. dissertation, The Chinese University of Hong Kong (Hong Kong), Hong Kong, 2022.

[22] "The University College London Archive of Stuttered Speech (UCLASS) | Journal of Speech, Language, and Hearing Research," https://pubs.asha.org/doi/abs/10.1044/1092-4388%282009/07-0129%29.

[23] N. Bernstein Ratner and B. MacWhinney, "Fluency Bank: A new resource for fluency research and practice," *Journal of Fluency Disorders*, vol. 56, pp. 69–80, Jun. 2018.

[24] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," Dec. 2022.

[25] S. S. Stevens, J. Volkmann, and E. B. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, Jan. 1937.

[26] M. J. Hunt, "Figures of merit for assessing connected-word recognisers," *Speech Communication*, vol. 9, no. 4, pp. 329–336, Aug. 1990.

[27] Y. Zhang and Q. Yang, "A Survey on Multi-Task Learning," Mar. 2021.

[28] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," Sep. 2015.

[29] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions," Feb. 2018.

[30] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," Apr. 2017.

[31] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," Jun. 2015.

[32] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," 1984.

[33] B. Rafaely, S. Weinzierl, O. Berebi, and F. Brinkmann, "Loss functions incorporating auditory spatial perception in deep learning – a review," Jun. 2025.

[34] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Aug. 2023.

[36] L. Borawar and R. Kaur, "ResNet: Solving Vanishing Gradient in Deep Networks," in *Proceedings of International Conference on Recent Trends in Computing*, R. P. Mahapatra, S. K. Peddoju, S. Roy, and P. Parwekar, Eds. Singapore: Springer Nature, 2023, pp. 235–247.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[38] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "FastSpeech: Fast, Robust and Controllable Text to Speech," Nov. 2019.

[39] E. Gusó, J. Pons, S. Pascual, and J. Serrà, "On loss functions and evaluation metrics for music source separation," Feb. 2022.

[40] Z. Liu, "Super Convergence Cosine Annealing with Warm-Up Learning Rate," in *CAIBDA 2022; 2nd International Conference on Artificial Intelligence, Big Data and Algorithms*, Jun. 2022, pp. 1–7.

[41] R. Jiang, G. Du, S. Yu, Y. Guo, S. K. Goh, and H.-K. Tang, "CADE: Cosine Annealing Differential Evolution for Spiking Neural Network," Jun. 2024.

[42] T. Cazenave, J. Sentuc, and M. Videau, "Cosine Annealing, Mixnet and Swish Activation for Computer Go," in *Advances in Computer Games*, C. Browne, A. Kishimoto, and J. Schaeffer, Eds. Cham: Springer International Publishing, 2022, vol. 13262, pp. 53–60.

[43] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," Feb. 2020.

[44] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, no. 1, p. 6086, Mar. 2024.

[45] A. Zeyer, P. Bahar, K. Irie, R. Schluter, and H. Ney, "A Comparison of Transformer and LSTM Encoder Decoder Models for ASR," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. SG, Singapore: IEEE, Dec. 2019, pp. 8–15.

[46] R. Gong, H. Xue, L. Wang, X. Xu, Q. Li, L. Xie, H. Bu, S. Wu, J. Zhou, Y. Qin, B. Zhang, J. Du, J. Bin, and M. Li, "AS-70: A Mandarin stuttered speech dataset for automatic speech recognition and stuttering event detection," Jun. 2024.

[47] M. Ozker and P. Hagoort, "Susceptibility to auditory feedback manipulations and individual variability," *PLOS One*, vol. 20, no. 5, p. e0323201, May 2025.

[48] P. B. M. D. M. Buzzeti and C. M. C. D. Oliveira, "Immediate effect of delayed auditory feedback on stuttering-like disfluencies," *Revista CEFAC*, vol. 20, no. 3, pp. 281–290, May 2018.

[49] J. Kalinowski and T. Saltuklaroglu, "Choral speech: The amelioration of stuttering via imitation and the mirror neuronal system," *Neuroscience & Biobehavioral Reviews*, vol. 27, no. 4, pp. 339–347, Jun. 2003.

ACKNOWLEDGMENT

This research originated from personal initiative, as the student researcher has a stutter and sought to address this challenge through computational methods. The author expresses sincere gratitude to Mr. Cook, teacher of the Millburn High School Science Research Program, for his invaluable mentorship. Mr. Cook provided detailed guidance on the scientific process — from formulating research questions and establishing goals to conducting literature reviews and writing in accordance with academic standards. Equally important, he encouraged students to reach out to professors and domain experts, a practice that significantly shaped the independent yet rigorous nature of this work.

The author also thanks Professor Chen Zeng of The George Washington University. Professor Zeng generously offered his time in biweekly virtual meetings during the summer, providing feedback on datasets, reviewing model architectures, and suggesting future research directions. His role was purely advisory and uncompensated, and he neither contributed code nor writing. His mentorship strengthened the soundness, depth, and rigor of this work.

Special thanks are also extended to Professor JoAnne Cascia and Dr. Iyad Ghanim of Kean University, specialists in speech-language pathology, who provided critical insights into the clinical relevance and potential applications of this research. Their perspectives grounded the project in real-world practice, highlighting how computational models may complement therapeutic approaches and better serve individuals who stutter. Their contributions were advisory only, without involvement in coding or writing, and were provided entirely without compensation.

The guidance of Mr. Cook, Professor Zeng, Professor Cascia, and Dr. Ghanim reflects generosity and a genuine commitment to supporting student research. The author is deeply grateful for their time, patience, and willingness to share expertise.

Finally, the author acknowledges the broader research community that made this project possible. The datasets employed — including SEP-28k and LibriStutter — are publicly available and de-identified, developed by teams committed to advancing accessibility in speech technology. Without these resources, this project would not have been feasible.