Name of the Participating Students: Qianheng Xu

High School: Millburn High School

Province: Millburn

Country/Region: New Jersey, United States of America

Name of Advisor(s): Christopher Cook

Affiliation of Advisor(s): Millburn High School

Title of Research Report: StutterZero and StutterFormer: End-to-end speech conversion to transcribe and correct stutters

StutterZero and StutterFormer: End-to-end speech conversion to transcribe and correct stutters

Qianheng (Hendry) Xu

Millburn High School, Millburn, New Jersey, USA

Abstract

Over 70 million people worldwide suffer from stuttering, a speech impairment characterized by involuntary disruptions during normal speech. This creates communication barriers and limits access to voice-enabled technologies, which often fail to recognize disfluent speech or mistake stutters for meaningful words. This study develops two deep learning solutions for correcting stuttered speech conversion, that is, generating a fluent speech signal from a stuttered speech signal. First, this study develops an ASR-based pipeline consisting of a fine-tuned Whisper-Small ASR model to produce accurate transcripts and a text-to-speech generator. More importantly, this study presents the first end-to-end multitask models for stutter correction: StutterZero and StutterFormer. StutterZero uses an encoder-decoder architecture with conventional convolution and LSTM layers, while StutterFormer is a purely Transformer-based model. Both process Log-Mel spectrograms to directly generate fluent speech from stuttered audio. The SEP-28K and LibriStutter datasets, containing natural and artificially generated stuttered audio trains, evaluate both models' performance. StutterZero demonstrated statistically significant enhancements in five-fold cross-validation, including a 24% decrease in Word Error Rate (WER), a 4% reduction in Character Error Rate (CER), and a 31% increase in semantic similarity (BERTScore) compared to the leading Whisper-Medium model.. StutterFormer achieves better results, with a 28% reduction in WER, 9% reduction in CER, and a 34% improvement in BERTScore. Both models show strong potential for real-time stutter correction in digital communications and human-machine interactions. In addition to technical contributions, this work releases two extended, publicly available datasets of paired stuttered and fluent speech, supporting future research in stuttered speech processing.

Keywords: stuttering, speech processing, automatic speech recognition, deep learning, Whisper, transformer, LSTM, spectrogram, speech correction, human-computer interaction

This work was supported by Mr. Cook from Millburn High School, Professor Chen Zeng from The George Washington University, Professor JoAnne Cascia and Dr. Iyad Ghanim from Kean University.

CONTENTS

I Introduction 4

I-A Problem Statement . . . 4

I-B Background and Related Work . . . 5

I-B1 Conventional Digital Signal Processing Approaches . . . 5

I-B2 ASR & TTS-based Approaches . . . 6

I-B3 Deep Learning & End-to-end Approaches . . . 7

II Research objectives 8

III Methodology 8

III-A Data . . . 8

III-B ASR-based approach . . . 8

III-C Whisper-Small Architecture . . . 9

III-D Whisper-Small Fine-tuning . . . 10

III-E StutterZero Model Architecture . . . 11

III-E1 StutterZero Encoder . . . 11

III-E2 StutterZero Spectrogram Decoder . . . 12

III-E3 StutterZero Transcript Decoder . . . 13

III-E4 StutterZero Training . . . 13

III-F StutterFormer Model Architecture . . . 14

III-F1 Multi-Head Attention . . . 14

III-F2 StutterFormer Encoder . . . 15

III-F3 StutterFormer Decoders . . . 15

III-F4 StutterFormer Training . . . 16

IV Results 17

IV-A Benchmarking . . . 17

IV-B Wilcoxon Signed-Rank Test . . . 18

IV-C Ablation Study . . . 18

V Discussion 19

V-A Limitations . . . 19

V-B Future Work . . . 20

VI Conclusion 20

VII Acknowledgment 28

I. Introduction

A. Problem Statement

Stuttering is a prevalent speech disorder, impacting more than 70 million individuals globally [2]. It manifests through 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 [7] [37]. 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 [3].

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 I: 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 phonetic patterns commonly used in speech therapy and pathology.

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 [32]. Stuttering can also cause feelings of shame, fear, anxiety, and guilt [35]. 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.

For more severe forms of stuttering, automatic speech recognition (ASR) models — models that transcribe speech — may insert unintended words or even truncate the speech due to a block. Lea et al. [17] 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 nearly 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. 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.

B. Background and Related Work

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 a complex audio signal. Then, a ruleset or 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 Coefficients (LPCCs). For instance, MFCC features are generated by first applying the Fast Fourier Transform (FFT) to convert an audio sample from the amplitude to frequency, 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. Ultimately, a Discrete Cosine Transform (DCT) is used to generate the cepstral coefficients through decorrelation of features. This pipeline is applied to every window of audio, usually 20-50 ms long [11] [9].

K N et al. introduce a rule-based approach for stutter detection and removal by computing a "correlation factor" between adjacent audio windows using either MFCC or LPC features. A high correlation value — empirically determined to be 0.92 — was used to identify repeated or prolonged segments, prompting deletion of redundant frames. The unusually high correlation factor was used to detect repeated audio patterns, such as recurring phonemes or extended silences. However, the approach was evaluated exclusively on repetition and prolongation types, 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 [9]. Table II summarizes the performance, demonstrating high within-sample accuracy.

Stuttering Type	Feature 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[9].

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, 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 can be used to flag sections of stuttered audio for removal [6]. Table III summarizes representative DSP-based systems, their methodological choices, and reported performance.

Author(s)	**Dataset**	**Feature Extraction Method(s)**	**Method of Detecting Stutters**	**Type(s) of Stutter**	**Best Accuracy**
Mishra et al., 2021	UCLASS	MFCC, Root Mean Square Error	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	Convolutional neural network	Repetition, Block, Prolongation	Reduction of 71.24% in WER

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

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.

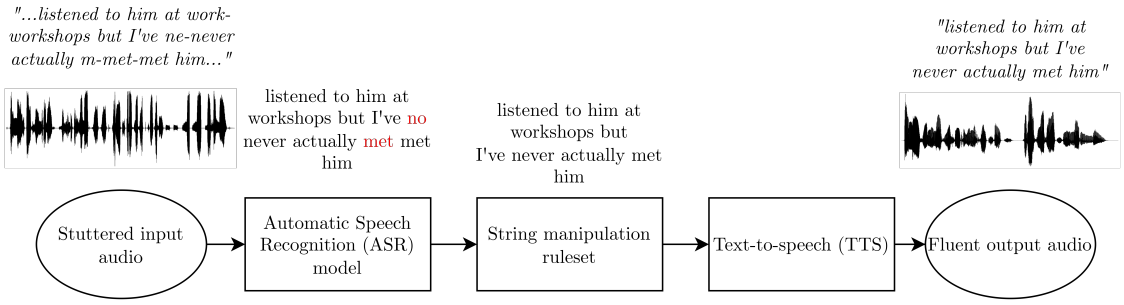

Fig. 1: General pipeline of a ASR-TTS approach

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% [16].

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 [33].

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 [38] [39] [42]. 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 in deep learning or end-to-end stutter correction. This paper will first review deep learning and end-to-end methods of fluent speech recognition, conversion, and the correction of dysarthic speech.

The popularity of end-to-end models stems from their ability to consolidate the entire 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 does not align to remove stutters at a sequence level [40] [46]. The lack of manual feature engineering for end-to-end models also allows for greater generalizability and adaptability to similar problems [43]. 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/output sequence lengths [42].

For example, Toshniwal et al. [29] developed an attention-based encoder-decoder model inspired by recurrent neural networks. 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. [29].

Wang [44] introduces an end-to-end encoder-decoder for dysarthic 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 dysarthia, 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 [44].

II. Research objectives

This study presents three pipelines for stutter correction. First, this research introduces and validates an ASR-based pipeline by fine-tuning Whisper-Small and applying a TTS model to generate fluent speech sequences.

This research also introduces two end-to-end models for stutter correction. Both are multitask, encoder-decoder models that do not rely on any specific feature extraction processes. StutterZero uses more conventional layers, such as Long Short-Term Memory (LSTM) and convolutional blocks, while StutterFormer relies purely on the Transformer architecture. Both end-to-end models are trained with the same dataset across a five-fold cross-validation.

III. Methodology

A. Data

This research utilizes two datasets: 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 [17]. All audio recordings were taken from public podcasts with people who stutter at a standard sampling rate of 16 kHz. 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 [7]. Because the LibriStutter dataset was created from fluent speech, it also contains fluent transcriptions of the stuttered speech. Unfortunately, the SEP-28K dataset does not have fluent transcriptions due to the time and energy required to manually label every speech clip.

The University College London Archive of Stuttered Speech (UCLASS) and the Fluencybank dataset are the two other more well-known stuttering datasets [34] [62]. 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 is restricted to a limited set of recordings available only through password-protected channels, making it unsuitable for large-scale training.

B. ASR-based approach

In order to have fluent speech data to train StutterZero as an end-to-end model, this research also developed an auxiliary pipeline to first generate fluent speech data, thereby “completing” both datasets. To reliably generate fluent audio sequences, this research first developed the ASR-TTS pipeline as shown in Figure 2:

1) Stuttered audio and fluent transcripts from LibriStutter are used to fine-tune a Whisper-Small ASR model that was originally trained on fluent speech only.
2) The fine-tuned Whisper-Small model was applied to the SEP-28K dataset, generating fluent transcripts for all SEP-28K audio clips.
3) A pretrained MeloTTS text-to-speech model was applied to all fluent transcripts from both datasets, generating reliable and clear flu ent audio sample counterparts.

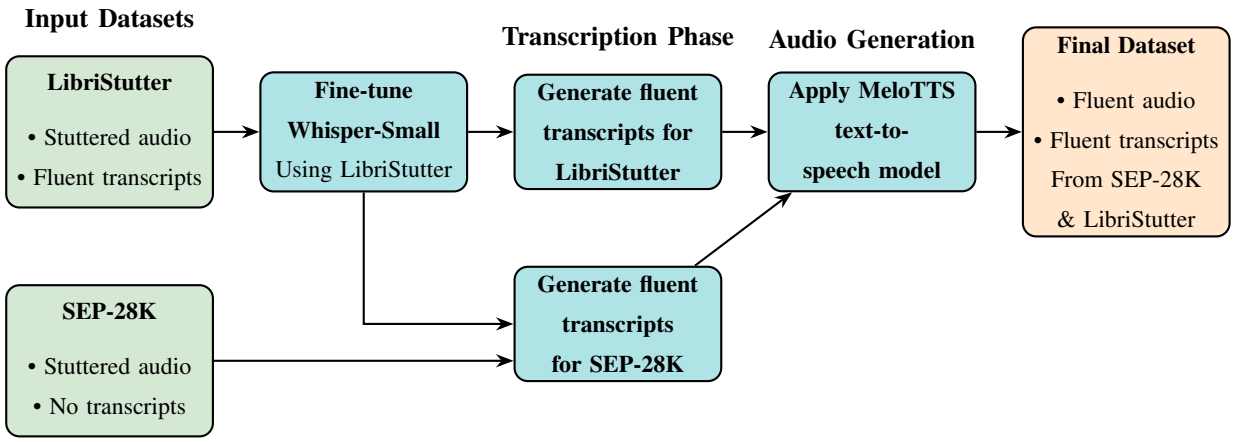

Fig. 2: Overview of this research's ASR-based pipeline. The LibriStutter dataset, containing stuttered audio and fluent transcripts, is used to fine-tune a Whisper model. This model is then used to generate fluent transcripts for both LibriStutter and SEP-28K (which lacks transcripts). The MeloTTS model is applied to the fluent transcripts to generate fluent audio, resulting in a final dataset with aligned fluent audio and transcripts derived from both source datasets.

A

C. Whisper-Small Architecture

This study fine-tunes Whisper-Small, a lightweight derivative from the Whisper family of ASR models. Audio data from both aforementioned datasets are in the amplitude domain, though Whisper-Small accepts a spectrogram in the frequency domain as input [22].

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 [8]. Figure 3 displays visualized spectrograms of the conversion process to obtain a Log-Mel Spectrogram.

Whisper-Small computes Log-Mel Spectrograms with 25 millisecond-long windows that move forward by 10 milliseconds every timestep.

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 a fully connected hidden layers. The encoder generatesa context vector, which is passed via cross attention to each of the 12 decoder attention blocks. Each decoder layer

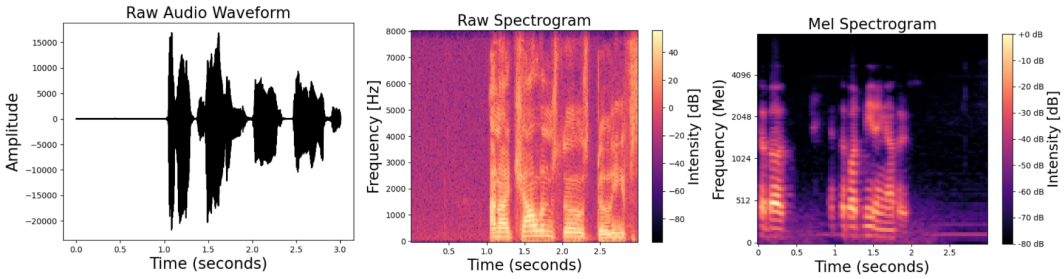

Fig. 3: Visualization of conversion to a Log-Mel Spectrogram. The raw audio waveform (left) presents the change in amplitude over time (amplitude domain). After the Short-Time Fourier Transform, the amplitude domain is converted to the frequency domain as a spectrogram (middle). After applying the Mel filterbank, the Log-Mel spectrogram is obtained (right)

contains a self-attention, cross-attention, and fully connected network [22]. At every timestep, the autoregressive decoder is fed the aggregated output tokens from all previous timesteps so 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, we split lengthy audio samples to the 30-second mark.

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) [47].

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

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).

Fine-tuning was done with 80% of all audio-transcript pairs used for training, 10% used for testing, and 10% used for validation. All training for this study was done on an NVIDIA RTX 3080 with 10 gigabytes of Video RAM. Training runs for a maximum of 10,000 steps with a small learning rate of 1e-5 and uses a batch size of 8 per device, with gradient accumulation over 2 steps 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 lowest WER model weights 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. The encoder converts the input sequence to 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 usual encoder-decoder models use one encoder and one decoder, this study proposes a multitask decoder where two decoders, given the same context vector, are forced to predict different data types. The spectrogram decoder autoregressively predicts the fluent spectrogram signal, while the transcript decoder autoregressively predicts the grapheme being uttered.

The spectrogram decoder generates the spectrogram one frame at a time, using the context vector along with only 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 4 displays an overview of the model architecture.

1) StutterZero Encoder: This encoder is composed of two convolutional blocks, each featuring a strided 2-dimensional convolution layer with a 3x3 kernel size, a 2x2 stride, and batch normalization. To capture the spatiotemporal characteristics of the input sequence, a convolution-augmented bidirectional long short-term memory network (ConvBiLTSM) was used. Log-Mel Spectrograms encode both spatial information about the frequency of a signal and temporal information about when a specific signal was uttered. The ConvBiLTSM replaces the traditional matrix multiplication operation of a set of weights on the entire input or hidden state with a convolution, encouraging learning local spatial patterns in two-dimensional data. For example, Equation (2) convolves the input gate kernel W_{xi} with the input tensor X_t with the $*$ operator instead of multiplying the matrices ($W_{xi}X_t$). Equations (2) to (7) define functions for input gate I_t, forget gate F_t, candidate cell state $\tilde{C}_t$, cell state C_t, output gate O_t and hidden state H_t respectively. W_{any} defines the weights to applied to any specific datapoint. H_{t-1} defines the hidden state of the previous timestep, and b_{any} defines trainable biases [25].

$$I_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + b_i) \tag{2}$$

$$F_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + b_f) \tag{3}$$

$$\tilde{C}_t = \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{4}$$

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \tag{5}$$

$$O_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + b_o) \tag{6}$$

$$H_t = O_t \odot \tanh C_t \tag{7}$$

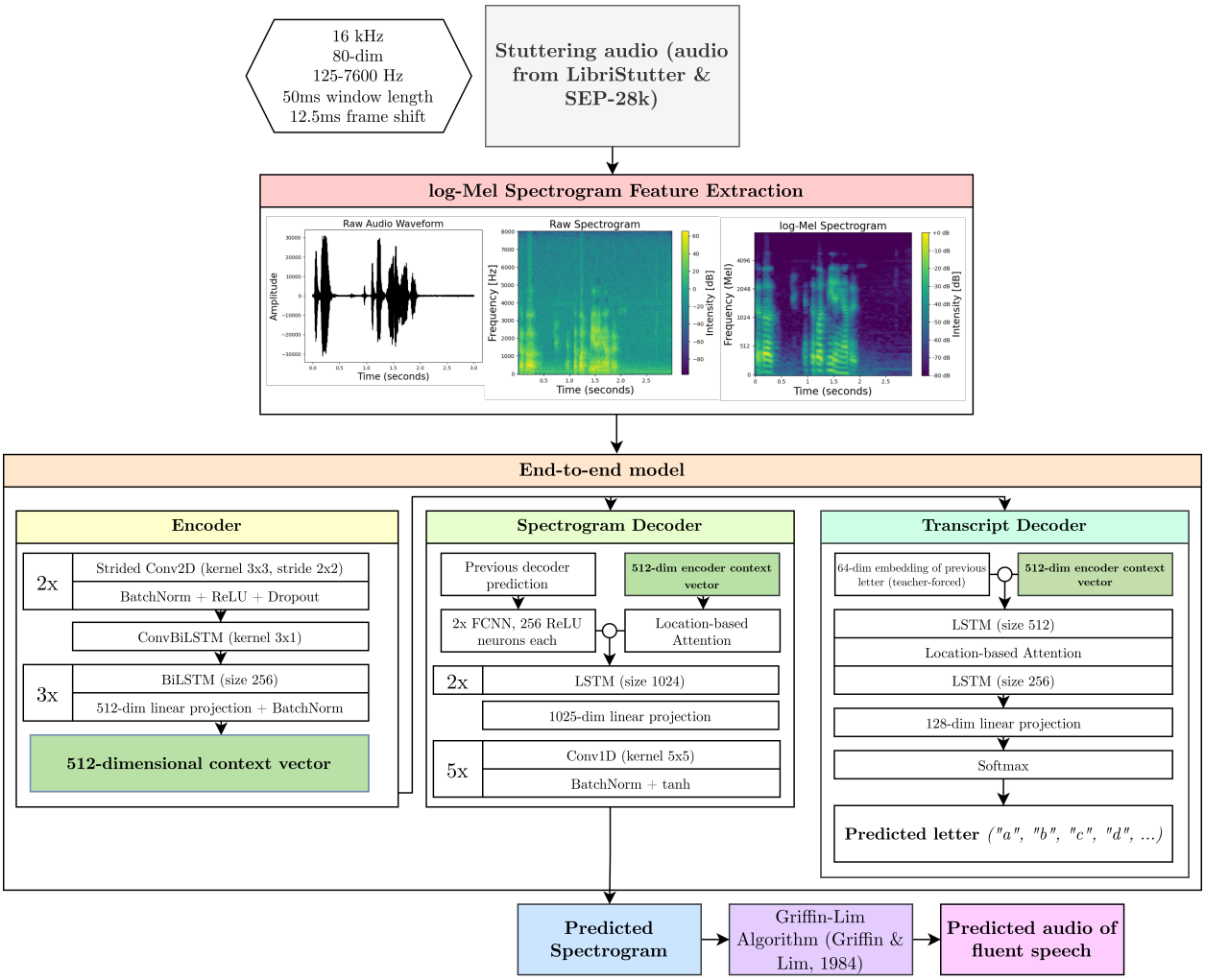

Fig. 4: Full network architecture of StutterZero: End-to-end model consists of an encoder and multitask decoders which fluents the output spectrogram and transcript.

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 compresses and transforms the previous spectrogram frames into a lower-dimension representation. A well-trained pre-net will simplify the input and extract the most salient features. If the raw spectrogram frames were used, the decoder may ”shortcut” the learning process by copying the previous frame or minimally modifying it [27] [23].

Instead of classical content-based attention, StutterZero used Location-based attention. Classical content-based attention computes an attention score e_t between query Q and key K as shown in Equation (8). e_t is a measure of how well Q and K_t match or are contextually relevant to each other. Attention weights w_t form a probability distribution that dictates how much the model should pay attention to the encoder state at t when predicting outputs. Finally, a weighted context vector c is used for output prediction [28].

$$e_t = \mathbf{v}^T \tanh(W_q Q + W_k K_t) \tag{8}$$

$$w_t = \text{softmax}(e_t) \tag{9}$$

$$c = \sum_{t=1}^{t_{\max}} w_t V_t \tag{10}$$

Because speech data is 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 f_t using concatenated previously calculated attention weights via a 1-dimensional convolution as shown in Equation 11

$$f_t = \text{Conv1D}([\sum_{t}^{t_{\max}} w_t]) \tag{11}$$

The location-based features are included in the attention score e_t as shown in Equation 12. This way, the decoder is informed by past attention behavior and adjusts its focus accordingly to maintain a monotonic flow of data.

$$e_t = \mathbf{v}^T \tanh(W_q Q + W_k K_t + W_f f_t) \tag{12}$$

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.

3) StutterZero Transcript Decoder: Previous work in text-to-speech and fluent speech conversion models has demonstrated the efficacy of multitask decoder architectures. Multitask 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 [22] [29].

In this case of the transcript decoder, adding an objective function at the grapheme level may force StutterZero to learn more intricate orthographical 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. 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, where wrong predictions may accumulate.

4) StutterZero Training: The spectrogram decoder uses mean-squared error (MSE) loss is defined in Equation 13 where S_{true} is the ground truth spectrogram, S_{pred} is the predicted fluent spectrogram, and f are the discretized frequency bins at time t [49] [48].

$$L_{\text{MSE}} = \frac{1}{F \times T} \sum_{f=1}^{F} \sum_{t=1}^{T} \left(S_{\text{true}}(f,t) - S_{\text{pred}}(f,t)\right)^2 \tag{13}$$

Because grapheme prediction is a categorical, Cross-Entropy loss is used for the transcript decoder. This is defined in Equation 14 where y is the ground truth grapheme, and $\mathbf{x}$ is the input Log-Mel Spectrogram. $\log P(y_t|y_{<t}, \mathbf{x})$ is the probabiity of predicting the correct token y_t given all previous tokens $y_{<t}$ and input features $\mathbf{x}$.

$$L_{\text{CE}} = -\sum_{t=1}^{T} \log P(y_t|y_{<t}, \mathbf{x}) \tag{14}$$

To combine the loss functions for both decoders, a bespoke loss function 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.

StutterZero utilized the Adam optimizer for training, starting with an initial 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. Like the ASR-based approach, 80% of all audio-transcript pairs were used for training, 10% used for testing, and 10% used for validation. It was trained for 1000 epochs on an NVIDIA RTX 3080 with 10 GB VRAM across five-fold cross-validation (90% training, 10% evaluation).

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 [57]. Figure 6 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 [57].

For every head i, global Q/K/V values are first projected into individual learned subspaces with linear projection matrices using matrices W_i^Q, W_i^K, and W_i^V.

$$Q_i = QW_i^Q, \quad K_i = KW_i^K, \quad V_i = VW_i^V \tag{15}$$

Each head then computes Attention individually:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) \tag{16}$$

Finally, all attention heads are reconcatenated:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_n) \tag{17}$$

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 [60] [59]. 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.

Interestingly, StutterFormer’s spectrogram decoder predicts both a ”coarse” and ”fine” spectrogram. The coarse spectrogram is a high-level prediction of the fluent signal — capturing global structures such as broad formats and the general distribution of energy. A fine spectrogram is used to supplement the final prediction with more subtle, detailed signals such as fricatives. Both spectrograms are merged with an element-wise weighted sum. The fine spectrogram is weighted at 30% of the strength of the coarse spectrogram. This ensures the fine spectrogram only augments the coarse spectrogram and does not overpower it. Figure 5 shows the coarse and fine spectrograms in the bottom left and right.

Finally, a post-net consisting of five 1-dimensional convolutions refines the predicted mel spectrogram to denoise and sharpen the final audio signal [27] [61].

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

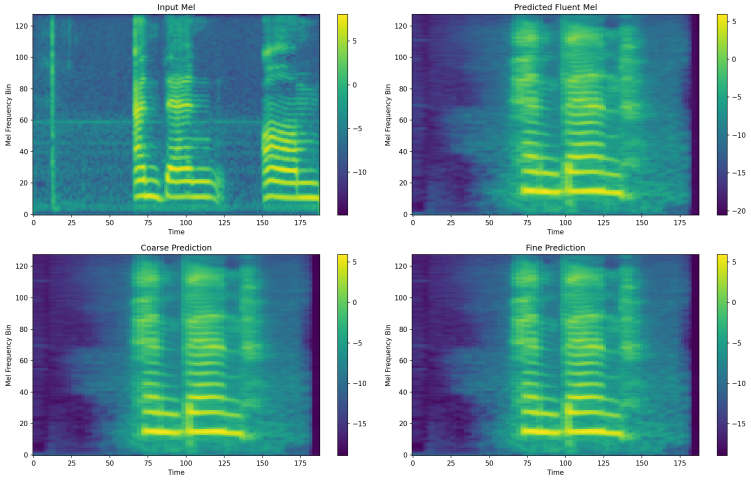

Fig. 5: Comparison of input, output, coarse, and fine spectrograms. Top Left: Input (stuttered) spectrogram, Top Right: Final predicted fluent spectrogram, Bottom Left: Coarse predicted spectrogram, Bottom Right: Fine predicted spectrogram.

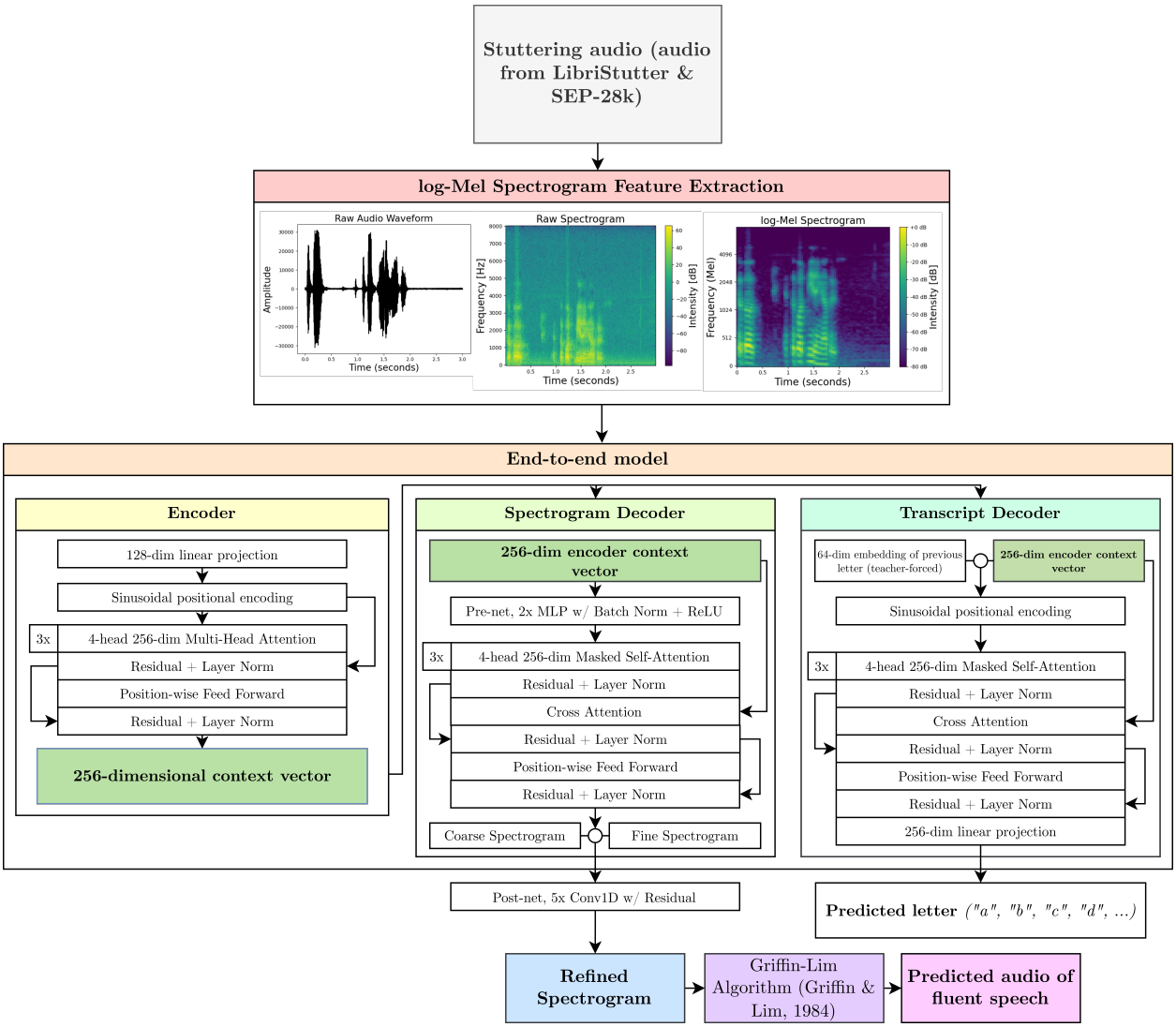

Fig. 6: Full network architecture of StutterFormer: Transformer-based end-to-end network for stutter correction.

4) StutterFormer Training: Like StutterZero, StutterFormer employs a hybrid loss function that combines multiple weighted losses. The spectrogram decoder also uses mean-squared error loss between the predicted and ground truth spectrograms. MSE loss is only used to predict the coarse fluent spectrograms. In addition to MSE loss, the spectrogram decoder also computes a mean absolute error (MAE) loss for predicting fine spectrograms only. 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 [63]. The transcript decoder also uses cross-entropy loss.

Like StutterZero, StutterFormer was trained for 1000 epochs on an NVIDIA RTX 3080 with 10 GB VRAM across five-fold cross-validation (90% training, 10% evaluation).

IV. 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. 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 (18) 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 [30]. 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 [30].

$$\text{CER} = \frac{\text{char. substitutions} + \text{char. deletions} + \text{char. insertions}}{\text{number of characters in the reference}} \tag{18}$$

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 7. This was an attempt to simulate what an untrained, average English-speaking person would interpret the fluent speech to be. The predicted transcripts of the fluent speech were compared with ground truth transcripts to calculate the final metrics. Table IV displays the mean WER and CER. The mean BERT scores for precision, recall, and F1 are shown in Table V.

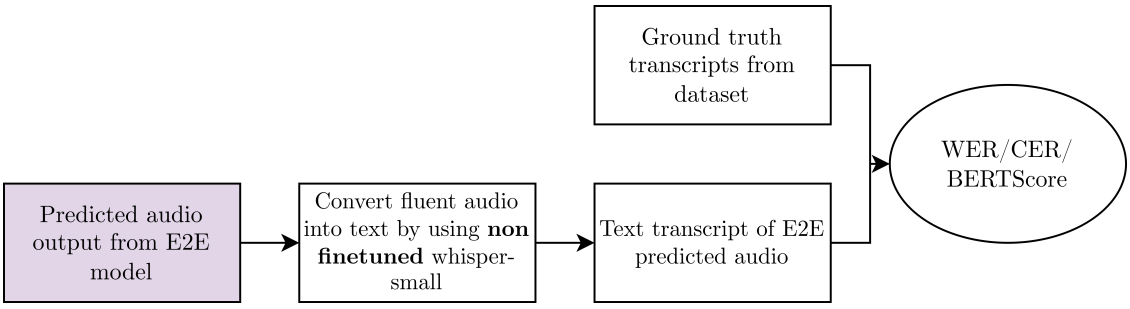

Fig. 7: The validation pipeline. An untouched Whisper-Small ASR model acted as "judge" to evaluate speech clarity and intelligibility

Model	WER	CER
Whisper-Tiny	0.415 ± 0.485	0.227 ± 0.326
Whisper-Small	0.370 ± 0.364	0.171 ± 0.293
Whisper-Medium	0.361 ± 0.302	0.162 ± 217
ASR-based	0.04 ± 0.032	0.02 ± 0.15
StutterZero (end-to-end)	0.116 ± 0.101	0.110 ± 0.096
StutterFormer (end-to-end)	0.08 ± 0.089	0.07 ± 0.109

TABLE IV: WER and CER Metrics for various models

Model	Mean BERTScore Precision	Mean BERTScore Recall	Mean BERTScore F1
Whisper-Tiny	0.5768 ± 0.157	0.6046 ± 0.1542	0.5915 ± 0.151
Whisper-Small	0.5956 ± 0.1581	0.6182 ± 0.1553	0.6076 ± 0.1531
Whisper-Medium	0.601 ± 0.1571	0.6195 ± 0.1579	0.6107 ± 0.1545
ASR-based	0.9516 ± 0.04	0.9517 ± 0.03	0.9517 ± 0.04
StutterZero (end-to-end)	0.9174 ± 0.034	0.9102 ± 0.025	0.9034 ± 0.046
StutterFormer (end-to-end)	0.9411 ± 0.012	0.9102 ± 0.011	0.9034 ± 0.013

TABLE V: Model performance comparison using BERTScore metrics

B. Wilcoxon Signed-Rank Test

The non-parametric, two-sided Wilcoxon Signed-Rank Test is used to confirm the statistical significance of the improvement made by this research's findings and baseline Whisper performance. 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 $< 1\mathrm{e}^{-100}$, 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 $< 1\mathrm{e}^{-100}$, significant at $\alpha = 0.05$. This shows that both StutterZero and StutterFormer demonstrate a significant improvement over state-of-the-art fluent speech models.

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.

Model	WER	CER	Mean BERTScore Precision
StutterZero (with transcript decoder)	0.116 ± 0.101	0.110 ± 0.096	0.9174 ± 0.034
Ablated StutterZero (without transcript decoder)	0.339 ± 0.176	0.260 ± 0.166	0.437 ± 0.1658
StutterFormer (with transcript decoder)	0.08 ± 0.089	0.07 ± 0.109	0.9411 ± 0.012
Ablated StutterFormer (without transcript decoder)	0.221 ± 0.213	0.194 ± 0.286	0.552 ± 0.2096

TABLE VI: Ablation comparison using BERTScore metrics

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.

V. Discussion

This study introduces three pipelines for stuttered speech conversion and correction: a fine-tuned automatic speech recogntion-based system and two multitask end-to-end models.

All three approaches introduced by this study significantly outperformed state-of-the-art fluent speech recognition models. This research first validated the performance of a fine-tuned Whisper-Small ASR model and built the entire speech conversion pipeline by applying a TTS model to the fluent transcripts.

More importantly, this research presents the first end-to-end models for stutter correction. This proves the viability of end-to-end and encoder-decoder models in this application and opens the door for future research with more powerful and diverse encoder-decoder models. StutterZero demonstrates a significant improvement fluent speech models by using conventional LSTM and convolutional layers. By taking advantage of the nonlinear dependencies in human speech, the Transformer-based StutterFormer is able to perform slightly better than StutterZero by only using the Attention mechanism.

Finally, as per the ablation study, the multitask decoder architecture is crucial to model performance.

A. Limitations

There remain potential areas for improvement that can be explored in future research. First, StutterZero's reliance on TTS-generated fluent speech data for a substantial portion of the training and evaluation data introduces a prosodic mismatch between synthetic and naturally spoken audio. While this TTS-generation approach allowed for efficient generation of semantically accurate large-scale audio data, it limits StutterZero from fully capturing tone or emotion in the speaker's voice.

Second, 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 was used to produce a more candid estimate of the true performance both approaches, future steps should focus on data augmentation and training on larger datasets.

Third, 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.

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 [44]. 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 [50].

StutterZero/StutterFormer has 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) [52] [51]. 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 [56].

This research also has uses for real-time stuttering correction applications, especially over digital communication. StutterZero/StutterFormer could be further optimized through methods such as quantization or distillation to run more efficiently on lower-powered devices. Then, StutterZero/StutterFormer can work in real-time to correct the stutters of speakers. StutterZero/StutterFormer could be used in phone calls, video conferences, or online meetings. If a person stutters, their voice can be passed through StutterZero/StutterFormer before reaching everyone else in the meeting so that StutterZero/StutterFormer else hears only the fluent speech. In the same vein, StutterZero/StutterFormer has applications in voice recording and sound engineering, where a speaker's accidental stutter can be corrected flawlessly without having to re-record the speech.

VI. 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. Nguyen and F. Cardinaux, "NVC-Net: End-To-End Adversarial Voice Conversion," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2022, pp. 7012–7016, iSSN: 2379-190X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9747020

[2] 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. [Online]. Available: https://doi.org/10.1145/3441852.3471211

[3] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094730X16300067

[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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0887618509001236

[5] T. Kourkounakis, A. Hajavi, and A. Etemad, "Detecting Multiple Speech Disfluencies Using a Deep Residual Network with Bidirectional Long Short-Term Memory," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 6089–6093, iSSN: 2379-190X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9053893

[6] 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, iSSN: 2076-1465. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9616063

[7] T. Kourkounakis, A. Hajavi, and A. Etemad, "FluentNet: End-to-End Detection of Speech Disfluency with Deep Learning," Sep. 2020, arXiv:2009.11394 [eess]. [Online]. Available: http://arxiv.org/abs/2009.11394

[8] 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. [Online]. Available: https://doi.org/10.1121/1.1915893

[9] 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. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S187705092031125X

[10] H. Wang, H. Wang, Y. Guo, Z. Li, C. Du, X. Chen, and K. Yu, "Why Do Speech Language Models Fail to Generate Semantically Coherent Outputs? A Modality Evolving Perspective," Dec. 2024, arXiv:2412.17048 [eess]. [Online]. Available: http://arxiv.org/abs/2412.17048

[11] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222012772

[12] A. Dash, N. Subramani, T. Manjunath, V. Yaragarala, and S. Tripathi, "Speech Recognition and Correction of a Stuttered Speech," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Bangalore: IEEE, Sep. 2018, pp. 1757–1760. [Online]. Available: https://ieeexplore.ieee.org/document/8554455/

[13] V. K. Y Padma Sai, "Design and Implementation of Silent Pause Stuttered Speech Recognition System," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 04, no. 03, pp. 1253–1260, Mar. 2015. [Online]. Available: http://ijareeie.com/upload/2015/march/12_Design.pdf

[14] A. S. Ajibola, N. K. Bin Alang Md. Rashid, W. Sediono, and N. N. W. Nik Hashim, "A NOVEL APPROACH TO STUTTERED SPEECH CORRECTION," *Jurnal Ilmu Komputer dan Informasi*, vol. 9, no. 2, p. 80, Jun. 2016. [Online]. Available: http://jiki.cs.ui.ac.id/index.php/jiki/article/view/382

[15] N. Mishra, A. Gupta, and D. Vathana, "Optimization of stammering in speech recognition applications," *International Journal of Speech Technology*, vol. 24, no. 3, pp. 679–685, Sep. 2021. [Online]. Available: https://link.springer.com/10.1007/s10772-021-09828-w

[16] D. Mujtaba and N. Mahapatra, "Fine-Tuning ASR for Stuttered Speech: Personalized vs. Generalized Approaches," Jun. 2025, arXiv:2506.00853 [cs]. [Online]. Available: http://arxiv.org/abs/2506.00853

[17] 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, arXiv:2302.09044 [cs]. [Online]. Available: http://arxiv.org/abs/2302.09044

[18] R. Alnashwan, N. Alhakbani, A. Al-Nafjan, A. Almudhi, and W. Al-Nuwaiser, "Computational Intelligence-Based Stuttering Detection: A Systematic Review," *Diagnostics*, vol. 13, no. 23, p. 3537, Nov. 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10706171/

[19] X. Zhang, I. Vallés-Pérez, A. Stolcke, C. Yu, J. Droppo, O. Shonibare, R. Barra-Chicote, and V. Ravichandran, "Stutter-TTS: Controlled Synthesis and Improved Recognition of Stuttered Speech," Nov. 2022, arXiv:2211.09731 [cs]. [Online]. Available: http://arxiv.org/abs/2211.09731

[20] O. Shonibare, X. Tong, and V. Ravichandran, "Enhancing ASR for Stuttered Speech with Limited Data Using Detect and Pass," Feb. 2022, arXiv:2202.05396 [eess]. [Online]. Available: http://arxiv.org/abs/2202.05396

[21] Z. Jiang, Q. Yang, J. Zuo, Z. Ye, R. Huang, Y. Ren, and Z. Zhao, "FluentSpeech: Stutter-Oriented Automatic Speech Editing with Context-Aware Diffusion Models," in *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada: Association for Computational Linguistics, 2023, pp. 11 655–11 671. [Online]. Available: https://aclanthology.org/2023.findings-acl.741

[22] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," Dec. 2022, arXiv:2212.04356 [eess]. [Online]. Available: http://arxiv.org/abs/2212.04356

[23] 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, arXiv:1703.10135 [cs]. [Online]. Available: http://arxiv.org/abs/1703.10135

[24] 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. [Online]. Available: https://ieeexplore.ieee.org/document/9004025/

[25] 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, arXiv:1506.04214 [cs]. [Online]. Available: http://arxiv.org/abs/1506.04214

[26] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," Sep. 2019, arXiv:1909.09586 [cs]. [Online]. Available: http://arxiv.org/abs/1909.09586

[27] 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, arXiv:1712.05884 [cs]. [Online]. Available: http://arxiv.org/abs/1712.05884

[28] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," Jun. 2015, arXiv:1506.07503 [cs]. [Online]. Available: http://arxiv.org/abs/1506.07503

[29] S. Toshniwal, H. Tang, L. Lu, and K. Livescu, "Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition," Apr. 2017, arXiv:1704.01631 [cs]. [Online]. Available: http://arxiv.org/abs/1704.01631

[30] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," Feb. 2020, arXiv:1904.09675 [cs]. [Online]. Available: http://arxiv.org/abs/1904.09675

[31] C. Nang, D. Hersh, K. Milton, and S. R. Lau, "The Impact of Stuttering on Development of Self-Identity, Relationships, and Quality of Life in Women Who Stutter," *American Journal of Speech-Language Pathology*, vol. 27, no. 3S, pp. 1244–1258, Oct. 2018. [Online]. Available: http://pubs.asha.org/doi/10.1044/2018_AJSLP-ODC11-17-0201

[32] 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. [Online]. Available: https://brieflands.com/articles/mejrh-21494.html

[33] 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, arXiv:2401.00662 [cs]. [Online]. Available: http://arxiv.org/abs/2401.00662

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

[35] O. Bloodstein, N. B. Ratner, and S. B. Brundage, *A handbook on stuttering*. San Diego, CA: Plural Publishing, Inc, 2021.

[36] C. Lea, V. Mitra, A. Joshi, S. Kajarekar, and J. P. Bigham, "SEP-28k: A Dataset for Stuttering Event Detection From Podcasts With People Who Stutter," Feb. 2021, arXiv:2102.12394 [eess]. [Online]. Available:

http://arxiv.org/abs/2102.12394

[37] 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, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/23/19/8033

[38] 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, arXiv:2210.03538 [cs]. [Online]. Available: http://arxiv.org/abs/2210.03538

[39] T. Walczyna and Z. Piotrowski, "Overview of Voice Conversion Methods Based on Deep Learning," *Applied Sciences*, vol. 13, no. 5, p. 3100, Jan. 2023, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2076-3417/13/5/3100

[40] 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. [Online]. Available: http://ieeexplore.ieee.org/document/7472621/

[41] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results," Dec. 2014, arXiv:1412.1602 [cs]. [Online]. Available: http://arxiv.org/abs/1412.1602

[42] 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, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2073-8994/11/8/1018

[43] 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, arXiv:1412.5567 [cs]. [Online]. Available: http://arxiv.org/abs/1412.5567

[44] D. Wang, "A Scalable Encoder-Decoder Framework for Disordered Speech Reconstruction," Ph.D., The Chinese University of Hong Kong (Hong Kong), Hong Kong, 2022, iSBN: 9798368422909 Publication Title: PQDT - Global 30323248. [Online]. Available: http://ezp.bentley.edu/login?url=https://www.proquest.com/dissertations-theses/scalable-encoder-decoder-framework-disordered/docview/2776675591/se-2?accountid=8576

[45] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville, "Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks," Jan. 2017, arXiv:1701.02720 [cs]. [Online]. Available: http://arxiv.org/abs/1701.02720

[46] 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, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v32/graves14.html

[47] M. J. Hunt, "Figures of merit for assessing connected-word recognisers," *Speech Communication*, vol. 9, no. 4,

pp. 329–336, Aug. 1990. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/016763939090008W

[48] 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. [Online]. Available: https://ieeexplore.ieee.org/document/4775883

[49] B. Rafaely, S. Weinzierl, O. Berebi, and F. Brinkmann, "Loss functions incorporating auditory spatial perception in deep learning – a review," Jun. 2025, arXiv:2506.19404 [eess]. [Online]. Available: http://arxiv.org/abs/2506.19404

[50] 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, arXiv:2406.07256 [cs]. [Online]. Available: http://arxiv.org/abs/2406.07256

[51] 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. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1516-18462018000300281&lng=en&tlng=en

[52] M. Ozker and P. Hagoort, "Susceptibility to auditory feedback manipulations and individual variability," *PLOS One*, vol. 20, no. 5, p. e0323201, May 2025. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC12057900/

[53] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 2. Salt Lake City, UT, USA: IEEE, 2001, pp. 749–752. [Online]. Available: http://ieeexplore.ieee.org/document/941023/

[54] J. G. Beerends, A. P. Hekstra, A. W. Rix, and M. P. Hollier, "Perceptual Evaluation of Speech Quality (PESQ), the new ITU standard for end-to-end speech quality assessment. Part II – Psychoacoustic model."

[55] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. Dallas, TX, USA: IEEE, Mar. 2010, pp. 4214–4217. [Online]. Available: https://ieeexplore.ieee.org/document/5495701/

[56] 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. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0149763403000630

[57] 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, arXiv:1706.03762 [cs]. [Online]. Available: http://arxiv.org/abs/1706.03762

[58] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," May 2016, arXiv:1409.0473 [cs]. [Online]. Available: http://arxiv.org/abs/1409.0473

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs]. [Online]. Available: http://arxiv.org/abs/1512.03385

[60] 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.

[61] 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, arXiv:1905.09263 [cs]. [Online]. Available: http://arxiv.org/abs/1905.09263

[62] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094730X17300931

[63] E. Gusó, J. Pons, S. Pascual, and J. Serrà, "On loss functions and evaluation metrics for music source separation," Feb. 2022, arXiv:2202.07968 [cs]. [Online]. Available: http://arxiv.org/abs/2202.07968

VII. Acknowledgment

This research project originated from personal initiative as the Student has a stutter himself. This project was carried out over the course of one year as part of the Millburn High School Science Research Program, an AP-level course under the supervision and guidance of Mr. Cook. The entire project was an independent endeavor in which every stage of the process — including selecting the scientific topic, acquiring and preparing data, designing and implementing models, training and testing systems, and writing this report — was conducted entirely by the Student. All data analysis, computation, and training was conducted on the Student's personal desktop machine, using an RTX 3080 graphics card.

During the research process, a major difficulty was managing GPU video random access memory (VRAM) to handle training the end-to-end models. Both StutterZero and StutterFormer are fairly large models well over 10 million parameters, and the Student only had access to 10 GB of VRAM. The Student implemented methods such as VRAM garbage collection, gradient accumulation to mitigate these issues.

Firstly, the author wants to sincerely thank Mr. Cook, teacher of the Millburn High School Science Research Program. Mr. Cook provided detailed guidance on how to engage with the scientific process — how to identify meaningful research questions, establish clear goals, conduct a comprehensive literature review, and write in accordance with academic standards. Most importantly, he encouraged and instructed students on how to reach out to professors and domain experts through cold emails. His mentorship was central in shaping the Student's ability to carry out this project independently yet with professional-level rigor.

The Student also extends sincere gratitude to Professor Chen Zeng of The George Washington University. The Student had previously collaborated with Professor Zeng on a completely unrelated research project, and it was through this earlier connection that the Student sought his expertise for the present study. Professor Zeng generously offered his time in virtual meetings, held biweekly during the summer. In these sessions, he carefully reviewed the Student's methodology, provided feedback on the suitability of the datasets employed, and examined the deep learning model architectures being developed. His feedback not only validated the soundness of the Student's approach but also offered important perspectives on refining the models further. Professor Zeng also contributed ideas for future research directions, including ways this work might be extended to broader questions in computational linguistics and accessibility technologies. Importantly, Professor Zeng did not write any of the code or draft any part of this manuscript. His role was purely advisory, provided without compensation, and his mentorship was invaluable in helping the Student strengthen the rigor and depth of the research.

The Student is equally grateful to Professor JoAnne Cascia and Dr. Iyad Ghanim of Kean University. Both are specialists in speech-language pathology who kindly agreed to meet virtually with the Student. During these discussions, they provided critical insights into the clinical relevance and potential real-world applications of this research. Their perspectives as practitioners grounded the project in lived experience, highlighting how computational models might interact with therapeutic practices and address the needs of individuals who stutter. Although neither Professor Cascia nor Dr. Ghanim contributed code or writing to this project, their advice significantly expanded the Student's understanding of how computational approaches could serve not only as technical solutions but also as

meaningful tools in clinical and educational contexts. Their willingness to share their expertise was instrumental in bridging the gap between computer science research and speech-language pathology.

All of the guidance provided by Professors Zeng, Cascia, and Ghanim, as well as Mr. Cook, was entirely uncompensated. Their support reflects both generosity and a commitment to fostering scientific curiosity in students who are eager to learn. For their time, patience, and willingness to share their expertise, the Student is profoundly thankful.

Finally, the Student wishes to acknowledge the broader research community that made this project possible. The datasets used — SEP-28k and LibriStutter — are publicly available and de-identified, developed and maintained by researchers committed to advancing accessibility in speech technology. The Student is deeply appreciative of the teams who created these resources, without which this project would not have been feasible.