

# REPHRASETTS: DYNAMIC LENGTH TEXT BASED SPEECH INSERTION WITH SPEAKER STYLE TRANSFER

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

We propose a method for the task of text-conditioned speech insertion, i.e. inserting a speech sample in an input speech sample, conditioned on the corresponding complete text transcript. An example use case of the task would be to update the speech audio when corrections are done on the corresponding text transcript. The proposed method follows a transformer-based non-autoregressive approach that allows speech insertions of variable lengths, which are dynamically determined during inference, based on the text transcript and tempo of the available partial input. It is capable of maintaining the speaker’s voice characteristics, prosody and other spectral properties of the available speech input. Results from our experiments and user study on LibriTTS show that our method outperforms baselines based on an existing adaptive text to speech method. We also provide numerous qualitative results to appreciate the quality of the output from the proposed method.

## 1 INTRODUCTION

Large amount of audio data is being created and consumed every minute for a variety of pursuits. Yet processing audio is still quite hard and time consuming, *e.g.* if a single mistake is made during recording, the user is forced to rerecord the complete segment. A potential solution to this problem is to be able to use the corresponding text transcript to manipulate the audio data. If a change is needed, the user could change corresponding part of the text and an audio inpainting method could automatically make the corresponding change in the audio signal. In this paper, we address this problem and propose a network which can transfer speaker style and add or replace existing speech segments.

Earlier works in this area take an audio and a text transcript as input. The text transcript generally has a few words added or replaced as compared to the content of the input audio. The output is an audio for the text transcript in the style of the speaker from the input audio. There are two settings for this problem, (i) replace a segment of speech with an audio of equal length (Borsos et al., 2022) *i.e.* the length of the replaced audio fragment and hence the overall audio remains the same, (ii) insert dynamic length audio segments (Tang et al., 2021; Yin et al., 2022), thus increasing or decreasing the size of the output audio. Our proposed work belongs to the latter area. This problem is very challenging as it requires the system to automatically align the phonemes across text and speech, and does not place any restrictions on the length of the hole to be replaced — the user can potentially replace a single word with multiple words of longer duration. The inserted speech in the present case may be of different length than what was present in the original sample, *e.g.* in an original sample of “Fred was present in the meeting”, the user might want to replace “Fred” with “Fred Flintstone”.

In addition to the dynamic length of the part to be inserted, further challenges come in recreating the speaker and speech characteristics in the inpainted parts. Here, the problem bears similarities to adaptive text to speech problem (Choi et al., 2020; Casanova et al., 2021; Min et al., 2021), where a small reference sample of speech for a particular speaker is given and the task is to perform text to speech with new text segments in the style of the reference speech. In the speech insertion problem the context around the hole to be inserted provides the speaker characteristics. The task then is not only to generate a plausible speech segment for the replaced text, but also to match the context in terms of tempo, prosody and other higher level speaker characteristics of speech. The proposed method is designed to incorporate such aspects.

Our proposed network is based on FastSpeech2 (Ren et al., 2020), and adds an audio stream in parallel to the phoneme stream. The audio encoder processes the input audio context in form of mel-spectrograms where frames to be inserted are removed. We also propose a cross-modal attention module for extracting speech characteristics from the audio stream and using them to enhance the phoneme representation. We train the network using local and global adversarial losses with multi-layer discriminator feature matching and a style matching loss to improve the quality of synthesized speech samples. Similar to (Borsos et al., 2022; Yin et al., 2022), we derive a speech insertion baseline based on one of the state of the art Adaptive TTS methods Meta-StyleSpeech (Min et al., 2021), and compare our proposed method against it.

Our main contributions are as follows. (i) We propose a novel deep neural network for dynamic length speech insertion using cross-modal attention with adversarial local, global and style losses. (ii) We perform an exhaustive user study of the output generated by the proposed network on a public dataset. (iii) We empirically evaluate and perform ablation studies to highlight the effectiveness of the proposed method. (iv) We provide numerous qualitative results (in supplementary) to demonstrate the effectiveness of our method *c.f.* compared methods.

## 2 RELATED WORK

**Text-Conditioned Speech Inpainting.** Text-conditioned speech inpainting was first proposed by Prablanc et al. (2016). Their method first synthesizes speech from text and then uses voice conversion mapping on the synthesized speech to match the style of the observed speech. A disadvantage of this approach is that it requires a significant amount of data for each target speaker for learning the speaker specific conversion mappings.

Recently, Borsos et al. (2022) have proposed a method for the text-conditioned speech inpainting, which is based on multi-modal network Perceiver IO (Jaegle et al., 2021). Similar to our method, their method does not require any additional data other than the observed parts of the input speech. However, their setup of text-conditioned speech inpainting has a disadvantage—a fixed mask has to be provided in the input and it can only inpaint speech which has the same duration as the mask. In contrast, in our setup, our model automatically infers the duration of the gap that is to be inpainted based on the text transcript containing the replacement text.

**Text-to-Speech Synthesis.** Advances in neural text-to-Speech synthesis (Shen et al., 2018; Ren et al., 2020) have shown the capability of synthesizing natural speech free of artifacts. FastSpeech2 (Ren et al., 2020) has shown significant speedup in synthesis of speech samples by using non-autoregressive decoding of speech. Our method follows the design of FastSpeech2 to leverage such advantages.

**Adaptive Multi-Speaker TTS.** A closely related problem to text-conditioned speech inpainting is adaptive multi-speaker TTS (also known as voice cloning) (Arik et al., 2018; Chen et al., 2018; Jia et al., 2018). Recent advances in adaptive multi-speaker TTS (Choi et al., 2020; Casanova et al., 2021; Min et al., 2021) have made it possible for text-to-speech systems to synthesize speech in the style of a provided reference sample, while maintaining the speaker identity, prosody of the speech and characteristics of the recording environment. These methods can be potentially used to address the task proposed here. We construct an inpainting baseline based on adaptive TTS method MetaStyleSpeech (Min et al., 2021) and compare our method with it.

**Text-based speech insertion.** This line of work is closest to ours and is very recent. A few representative works in this area are Tang et al. (2021) and Yin et al. (2022). Tang et al. (2021) propose a zero shot text based speech insertion mechanism. They use ground-truth duration for existing phonemes to predict the duration of edited phonemes and align mel-spectrogram and phoneme representation. On the other hand, we do not use ground-truth duration for existing phonemes, and during inference, only the broad segmentation of audio and text are needed (B, I, A). Further, we use cross-modal attention to enhance phoneme representation with speech style from audio representations as it doesn't need explicit alignment between the phoneme and audio representation. We also add adversarial and style matching losses to enhance the quality of our samples, while Tang et al. (2021) uses only L2 loss. Yin et al. (2022) works on a two stage training pipeline for inserting dynamic length fragments. It uses the second stage to increase the quality of the reconstructed speech introduced by mean-square-error loss used during first stage of the training. Our method

directly outputs a high quality speech output with dynamic length test and does not require a second stage of enhancement.

### 3 APPROACH

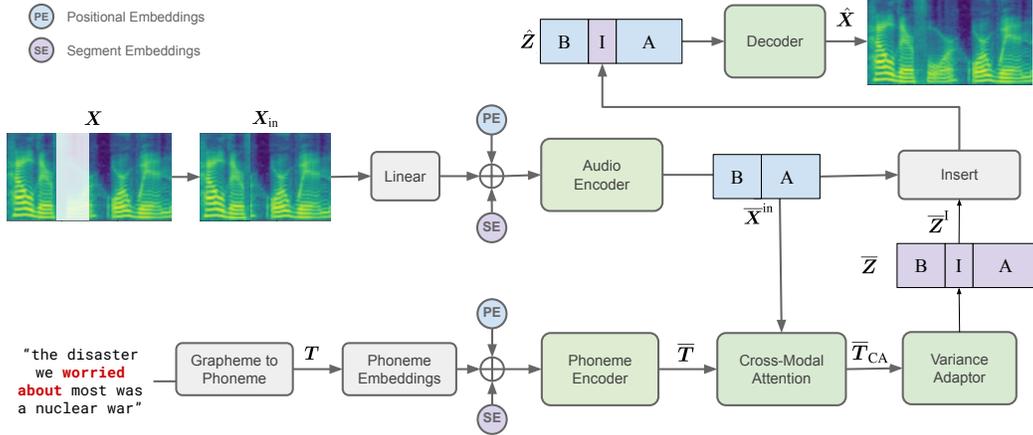


Figure 1: Overview of our proposed RephraseTTS model. It follows the general framework of FastSpeech2 of Ren et al. (2020) with transformer based encoders for encoding mel-spectrograms and phonemes. A cross-modal attention module is used for infusing the style information from the audio representations into the phoneme representations. The phoneme-level speech characteristics (pitch, energy and duration) are predicted by the variance adaptor and added to the phoneme representations. The final representation, obtained by inserting the middle segment of the variance adaptor output into the audio encoder output, are decoded into the output mel-spectrogram by the decoder. A pretrained vocoder (not shown here) is finally used to produce the output audio waveform.

#### 3.1 PROBLEM FORMULATION

Consider an audio-text pair  $(X, T)$ , where  $X \in \mathbb{R}^{L \times d_{\text{mel}}}$  is the mel-spectrogram representation of the audio, and  $T$  is the text transcript represented by the sequence of phonemes  $T = \{p_1, p_2, \dots, p_K\}$ ,  $p_i \in \mathbb{P}$ .  $L$  and  $d_{\text{mel}}$  are the number of frames and the number of frequency channels in the mel-spectrogram respectively.  $\mathbb{P}$  is the phoneset i.e. the predefined set of all phonemes, and  $K$  is the number of phonemes in the text. We assume that the phoneme-level alignment between the audio  $X$  and the text transcript  $T$  is known beforehand, as it can be extracted using tools such as Montreal Forced Aligner (MFA) (McAuliffe et al., 2017) with good accuracy.

To define the input and the target for our speech insertion model, we divide  $X$  and  $T$  into three segments as  $X = (X^B, X^I, X^A)$  and  $T = (T^B, T^I, T^A)$ .  $X^I$  is the speech segment that we aim to resynthesize, while  $X^B$  and  $X^A$  are the segments that come before and after that segment.  $T^B$ ,  $T^I$  and  $T^A$  represent the subsequences of phonemes corresponding to  $X^B$ ,  $X^I$  and  $X^A$  respectively.

The goal of speech insertion task is to reconstruct the full audio  $X$  from the partial audio context  $X^{in} = (X^B, X^A)$  and the full text transcript  $T = (T^B, T^I, T^A)$ ,

$$\hat{X} = G(X^{in}, T) = (\hat{X}^B, \hat{X}^I, \hat{X}^A) \quad (1)$$

Here,  $G$  denotes the speech insertion network and  $\hat{X}$  denotes the full synthesized mel-spectrogram as reconstructed by  $G$ .

#### 3.2 MODEL ARCHITECTURE

Our speech insertion model RephraseTTS (Figure 1) denoted by  $G$  follows the text-to-speech (TTS) framework of FastSpeech2 (Ren et al., 2020). An audio encoder and a phoneme encoder (Section 3.2.1) encode the input mel-spectrogram  $X^{in}$  and the phoneme sequence  $T$  respectively. Then,

a cross-modal attention block (Section 3.2.2) infuses the audio-context information from the output of audio encoder into the phoneme representations produced by the phoneme encoder. A variance adaptor (Section 3.2.3) then predicts the phone-level pitch and energy information and adds them to the phoneme representations. After adding pitch and energy information to the phoneme representations, the variance adaptor predicts duration of each phoneme in terms of the number mel-spectrogram frames and expands the phoneme representations by replicating each phoneme embedding by the duration predicted for that particular phoneme. The expanded phoneme-representation is then fed into the decoder (Section 3.2.4) to get the mel-spectrogram reconstruction  $\hat{X}$ .

### 3.2.1 ENCODERS

The audio encoder first converts the mel-spectrogram frames in  $\mathbf{X}_{\text{in}} \in \mathbb{R}^{L_{\text{in}} \times d_{\text{mel}}} = (\mathbf{X}^{\text{B}}, \mathbf{X}^{\text{A}})$  from  $d_{\text{mel}}$  to a sequence of  $d$ -dimensional vectors using a linear layer. To feed the segment information to the encoder i.e. whether a frame comes from  $\mathbf{X}_B$  or  $\mathbf{X}_A$ , we learn two segment embedding vectors  $e_X^{\text{B}}, e_X^{\text{A}} \in \mathbb{R}^d$ . One of the two segment embedding vectors is added to each projected mel-spectrogram encoding vector depending on the segment it lies on. For positional information, sinusoidal positional encodings  $E_X^{\text{pos}} \in \mathbb{R}^{L_{\text{in}} \times d}$  (Vaswani et al., 2017; Ren et al., 2020) are also added to the mel-spectrogram encoding vectors. Finally, the resulting audio representation is passed through a stack of feed-forward transformer blocks, to get the encoded audio representation  $\bar{\mathbf{X}}^{\text{in}}$ ,

$$\bar{\mathbf{X}}^{\text{in}} = \text{Audio-Encoder}(\mathbf{X}^{\text{in}}), \quad \bar{\mathbf{X}}^{\text{in}} \in \mathbb{R}^{L_{\text{in}} \times d} \quad (2)$$

We follow a similar procedure for encoding the input phoneme sequence  $\mathbf{T} = (\mathbf{T}^{\text{B}}, \mathbf{T}^{\text{I}}, \mathbf{T}^{\text{A}})$ . First,  $\mathbf{T}$  is converted into a sequence of  $d$ -dimensional embedding vectors using a look-up table of learnable phoneme embeddings. Then, similarly to mel-encoder, the segment information is incorporated using three embedding vectors  $e_T^{\text{B}}, e_T^{\text{I}}, e_T^{\text{A}} \in \mathbb{R}^d$ . Sinusoidal positional encodings  $E_T^{\text{pos}} \in \mathbb{R}^{K \times d}$  are also added to the phoneme embeddings. And finally these phoneme embeddings are fed to a stack of feed-forward transformer blocks, to get the encoded phoneme representation  $\bar{\mathbf{T}}$ ,

$$\bar{\mathbf{T}} = \text{Phoneme-Encoder}(\mathbf{T}), \quad \bar{\mathbf{T}} \in \mathbb{R}^{K \times d} \quad (3)$$

### 3.2.2 CROSS-MODAL ATTENTION

The phoneme encodings in  $\bar{\mathbf{T}}$  do not contain any information regarding the style of the speech i.e. the non-textual characteristics of the speech such as speaker’s voice timbre, prosody of the speech, background noise profile and other characteristics of the recording environment. This information must be inferred from the audio representation  $\bar{\mathbf{X}}^{\text{in}}$ . To extract this information from  $\bar{\mathbf{X}}^{\text{in}}$  and infuse it into the phoneme representations, we use a multi-headed cross-attention block.

We use  $H$  cross-attention blocks. For each head, we first compute queries  $\mathbf{Q}_T^i \in \mathbb{R}^{K \times d_k}$  from  $\bar{\mathbf{T}}$ , and keys  $\mathbf{K}_X^i \in \mathbb{R}^{L_{\text{in}} \times d_k}$  and values  $\mathbf{V}_X^i \in \mathbb{R}^{L_{\text{in}} \times d_v}$  from  $\bar{\mathbf{X}}^{\text{in}}$  via linear projections. Here,  $i$  is the index number of the head. The cross-attention head then computes a scaled dot-product attention (Vaswani et al., 2017) on  $\mathbf{Q}_T^i$ ,  $\mathbf{K}_X^i$  and  $\mathbf{V}_X^i$ . Then, we concatenate the output of each attention head and project it onto a  $d$ -dimensional space, to get the final phoneme representation  $\bar{\mathbf{T}}_{\text{CA}}$ , which now also contains the style information of the speech.

$$\mathbf{H}_i = \text{Cross-Attention}(\mathbf{Q}_T^i, \mathbf{K}_X^i, \mathbf{V}_X^i), \quad i \in 1, \dots, H \quad (4)$$

$$= \text{Softmax}\left(\frac{\mathbf{Q}_T^i \mathbf{K}_X^{i\top}}{\sqrt{d_k}}\right) \cdot \mathbf{V}_X^i, \quad (5)$$

$$\bar{\mathbf{T}}_{\text{CA}} = \text{Concat}(\mathbf{H}_1, \dots, \mathbf{H}_H). \quad (6)$$

### 3.2.3 VARIANCE ADAPTOR

We also use the variance adaptor from FastSpeech2 in RephraseTTS. The variance adaptor first predicts the phoneme-level pitch and energy information from  $\bar{\mathbf{T}}_{\text{CA}}$ . It then adds the pitch and energy predictions, encoded by  $d$ -dimensional vectors, to the individual phoneme representations in

$\bar{T}_{CA}$ .

$$\hat{E}^{\text{pitch}} = \text{Pitch-Predictor}(\bar{T}_{CA}), \quad (7)$$

$$\hat{E}^{\text{energy}} = \text{Energy-Predictor}(\bar{T}_{CA} + \hat{E}^{\text{pitch}}), \quad (8)$$

$$\bar{T}_{CA}^{p,e} = \bar{T}_{CA} + \hat{E}^{\text{pitch}} + \hat{E}^{\text{energy}}. \quad (9)$$

Once the pitch and energy information are added to the representations, it predicts the duration for each phoneme in terms of number of mel spectrogram frames. Finally it upsamples the phoneme representations by replicating each phoneme embedding by the duration predicted for that particular phoneme,

$$\bar{Z} = \text{Length-Regulator}(\bar{T}_{CA}^{p,e}, \hat{d}), \quad \bar{Z} \in \mathbb{R}^{L' \times d} \quad (10)$$

where,  $\hat{d} = \{\hat{d}_1, \dots, \hat{d}_K\}$ ,  $\hat{d}_i \in \mathbb{Z}^+$  are the phoneme durations predicted by the duration-predictor conditioned on  $\bar{T}_{CA}^{p,e}$ . The number of vectors in the output representation is  $L' = \sum_{i=1}^K \hat{d}_i$ . The length regulator is used to obtain a one-to-one alignment between the feature vectors in  $\bar{Z}$  and the output mel-spectrogram frames. This allows a fast non-autoregressive decoding of the mel-spectrogram frames from  $\bar{Z}$ .

### 3.2.4 DECODER

Since audio representations for segments before and after the missing segment are already available in  $\bar{X}^{\text{in}}$ , we only keep the embedding vectors  $\bar{Z}^1$  from  $\bar{Z}$  that correspond to the phonemes in segment  $T^1$ . We insert  $\bar{Z}^1$  into the audio representation  $\bar{X}^{\text{in}} = (\bar{X}^{\text{B}}, \bar{X}^{\text{B}})$  to obtain the final representations  $\hat{Z} = (\bar{X}^{\text{B}}, \bar{Z}^1, \bar{X}^{\text{B}})$ .

We feed  $\hat{Z}$  to the transformer-based decoder (Ren et al., 2020) to compute the final reconstruction of the full mel-spectrogram,

$$\hat{X} = \text{Decoder}(\hat{Z}), \quad \hat{X} \in \mathbb{R}^{L_{\text{out}} \times d_{\text{mel}}} \quad (11)$$

## 3.3 TRAINING STRATEGY

Our method allows the reconstructed spectrogram  $\hat{X}$  to have different length from the target spectrogram  $X$ . As a result, it is not straightforward to compare and compute loss between them. To avoid this, following (Ren et al., 2020), we use ground-truth pitch, energy and duration in eq. (9) and 10 during training. At the same time, we use them as supervision signals for training the variance predictors. During inference, when the pitch, energy and duration for phonemes is unknown, we use the output of the variance predictors in eq. (9) and 10.

### 3.3.1 L1 RECONSTRUCTION LOSS

We train our model primarily with the L1 loss between  $\hat{X}$  and  $X$ . We give additional weight to synthesis of the speech segment that is not available in the input audio by adding an additional loss term that computes the L1 error only between  $X^I$  and  $\hat{X}^I$

$$\mathcal{L}_{\text{rec}} = \frac{1}{L \cdot d_{\text{mel}}} (\|X - \hat{X}\|_1) + \frac{\lambda_1}{L^I \cdot d_{\text{mel}}} (\|X^I - \hat{X}^I\|_1) \quad (12)$$

### 3.3.2 LOCAL AND GLOBAL DISCRIMINATORS

Training our model only with the L1 loss in eq. (12) already achieves speech insertion results that are highly intelligible and match the style of the input speech. However, they still show significant robotic artifacts. To remedy this, we employ adversarial losses based on global and local discriminators.

The global discriminator  $D_g$  is a convolutional network that takes as input the full spectrogram of the speech samples, ground-truth or synthesized by our model, and outputs a single scalar value that

indicates whether the input is from a real or fake spectrogram. We use the LSGAN loss from Mao et al. (2016) to train  $D_g$ ,

$$\min_{D_g} \mathcal{L}(D_g) = \mathbb{E} \left[ \left( D_g(\mathbf{X}) - 1 \right)^2 + \left( D_g(\hat{\mathbf{X}}) - 0 \right)^2 \right]. \quad (13)$$

Since we want our model to output mel-spectrograms that are as close to real mel-spectrograms as possible, we add the following adversarial loss term to the main objective,

$$\mathcal{L}_{\text{adv},g} = \mathbb{E} \left[ \left( D_g(\hat{\mathbf{X}}) - 1 \right)^2 \right] \quad (14)$$

We also add a feature matching loss, where we add L1 errors between the intermediate discriminator features of synthesized and reconstructed mel-spectrogram,

$$\mathcal{L}_{\text{feat},g} = \mathbb{E} \left[ \sum_{i=1}^{L_{D_g}} \left\| D_g(\mathbf{X})_i - D_g(\hat{\mathbf{X}})_i \right\|_1 \right]. \quad (15)$$

Here,  $L_{D_g}$  is the number of discriminator layers we selected for extracting the features, and  $D_g(\cdot)_i$  are the features extracted from the  $i$ -th layer.

We follow a similar process for implementing the local adversarial losses. The local discriminator  $D_l$  shares the same architecture as  $D_g$ , but instead of taking the full spectrogram as input, it only takes short windows, sampled evenly from  $\mathbf{X}^1$  and  $\hat{\mathbf{X}}^1$ . This allows our speech insertion network to focus on the low-level local characteristics of the inserted speech segment and make them indistinguishable from the real target speech segments.

More formally, we sample short mel-spectrogram windows of fixed length with a fixed hop length from the segments  $\mathbf{X}^1$  and  $\hat{\mathbf{X}}^1$ . We denote this set of windows by  $\mathcal{W} = \{(\mathbf{W}_j, \hat{\mathbf{W}}_j)\}_{j=1}^J$ , where  $\hat{\mathbf{W}}_j$  is the  $j$ -th window sampled from  $\hat{\mathbf{X}}^1$  and  $\mathbf{W}_j$  is its corresponding ground-truth window. We use the following LSGAN loss for training  $D_l$ ,

$$\min_{D_l} \mathcal{L}(D_l) = \mathbb{E} \left[ \sum_{j=1}^J \left( D_l(\mathbf{W}_j) - 1 \right)^2 + \left( D_l(\hat{\mathbf{W}}_j) - 0 \right)^2 \right] \quad (16)$$

And similar to eq. (14) and eq. (15) the local LSGAN and feature matching losses are given by,

$$\mathcal{L}_{\text{adv},l} = \mathbb{E} \left[ \sum_{j=1}^J \left( D_l(\hat{\mathbf{W}}_j) - 1 \right)^2 \right], \quad \mathcal{L}_{\text{feat},l} = \mathbb{E} \left[ \sum_{j=1}^J \sum_{i=1}^{L_{D_l}} \left\| D_l(\mathbf{W}_j)_i - D_l(\hat{\mathbf{W}}_j)_i \right\|_1 \right] \quad (17)$$

### 3.3.3 STYLE MATCHING LOSS

We add a style matching loss (Figure 2) to encourage our model to synthesize mel-spectrograms that match the style of the input speech. To implement this loss, we train a style network  $F_s$  that takes short windows sampled from the mel-spectrograms and extracts their style as a vectors with  $d_{\text{style}}$  dimensions.

The style extractor is trained jointly with our model. In each iteration of the training, we sample windows with fixed number of frames from all examples in the batch of ground-truth  $\{\mathbf{X}_k\}_{k=1}^B$ , where  $\mathbf{X}_k$  is the ground-truth mel-spectrogram of the  $k$ -th example and  $B$  is the batch size. We also sample windows from the mel-spectrograms predicted by our model  $\{\hat{\mathbf{X}}_k\}_{k=1}^B$ . We extract style vectors for all sampled windows and mine triplets of anchor, positive and negative  $(\mathbf{a}, \mathbf{p}, \mathbf{n})$  from the extracted style vectors. Specifically, for an anchor window sampled from a real mel-spectrogram  $\mathbf{X}_k$ , all other windows sampled from the same mel-spectrogram are considered as positive, while windows sampled from another real mel-spectrograms in the batch or synthesized mel-spectrograms are considered as negative. We denote the set of triplets mined from the training batch in this way as  $T_s$ . We use triplet margin loss with a margin  $m$  to train  $F_s$ ,

$$l(\mathbf{a}, \mathbf{p}, \mathbf{n}) = \max(\|\mathbf{a} - \mathbf{p}\|_2 - \|\mathbf{a} - \mathbf{n}\|_2 + m, 0) \quad (18)$$

$$\min_{F_s} \mathcal{L}(F_s) = \mathbb{E} \left[ \sum_{(\mathbf{a}, \mathbf{p}, \mathbf{n}) \in T_s} l(\mathbf{a}, \mathbf{p}, \mathbf{n}) \right] \quad (19)$$

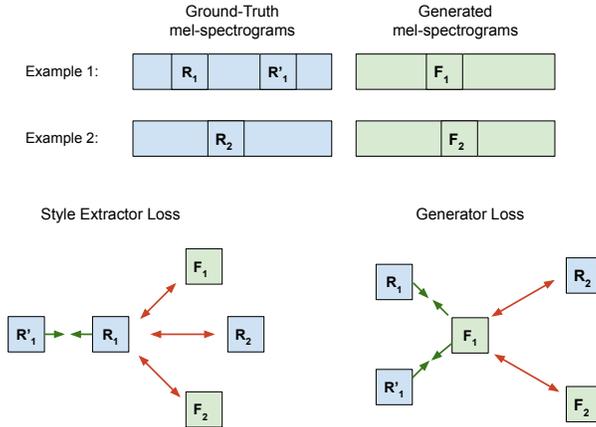


Figure 2: Illustration of style matching loss. Two examples in the training batch are considered and the ground-truth and generated mel-spectrograms are shown in blue and green respectively (top). The triplet mining strategy for training the style extractor (bottom-left) vs. the speech insertion model (bottom-right) are shown below. The anchor-positive relationships and anchor-negative relationships in the triplets are indicated with the green and red arrows respectively.

Note that this triplet loss considers a pair of synthesized and real mel-spectrogram windows from a single example as a negative and encourages  $F_s$  to predict style vectors that are far from each other. Since, we want our model  $G$  to predict speech that closely resembles the style of the real speech input, we add a style matching loss to our main objective. This style matching loss is also based on the triplet margin loss in eq. (18). However, we mine the set of triplets differently. For an anchor window from a synthesized mel-spectrogram  $\hat{X}_k$ , all windows sampled from the corresponding real mel-spectrogram  $X_k$  are considered as positive, while windows sampled from other real or synthesized mel-spectrograms are considered as negative. The set of triplets mined in this way are denoted by  $T_G$ . Thus, the style matching loss becomes,

$$\mathcal{L}_{\text{style}} = \mathbb{E} \left[ \sum_{(\mathbf{a}, \mathbf{p}, \mathbf{n}) \in T_G} l(\mathbf{a}, \mathbf{p}, \mathbf{n}) \right] \quad (20)$$

### 3.3.4 OVERALL OBJECTIVE

We train our RephraseTTS model in two phases. In the first phase, we only train with the L1 reconstruction loss  $\mathcal{L}_{\text{rec}}$ , eq. (12). In the second phase, we also add the adversarial LSGAN and feature matching losses as well as the style matching loss to our main objective. The overall loss in the second phase is given by the following,

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{adv},g} \mathcal{L}_{\text{adv},g} + \lambda_{\text{feat},g} \mathcal{L}_{\text{feat},g} + \lambda_{\text{adv},l} \mathcal{L}_{\text{adv},l} + \lambda_{\text{feat},l} \mathcal{L}_{\text{feat},l} + \lambda_{\text{style}} \mathcal{L}_{\text{style}}. \quad (21)$$

## 4 EXPERIMENTS

**Dataset.** We train our speech insertion model on the train-clean-360 subset of the LibriTTS (Zen et al., 2019) dataset. It has over 100k speech utterances spanning 190 hours from 904 speakers. The text transcripts for all utterances are available. For evaluation, we use dev-clean and the dev-other subsets from the LibriTTS dataset. The dev-clean and the dev-other subset have speech utterances from 40 and 33 speakers respectively, none of which appear in the training set. For quantitative evaluation, we used a subset of 512 random utterances from each test set. For human user study, we further selected a sample of 15 utterances from the dev-clean subset.

**Implementation Details** Kindly see Appendices for implementation details.

**Baselines.** We evaluate our model’s performance against several baselines. For all baseline methods as well as our own method, we only keep the inserted speech segment from the method and stitch it with the ground-truth audio of the available part. The baseline methods are described as follows.

**(1) GT-Mel + Vocoder.** The missing part of the speech is reconstructed from the ground-truth mel-spectrograms using the HifiGAN vocoder and then stitched with the ground-truth audio of the available part. This baseline represents an upper bound for our model’s performance as we use the same pretrained vocoder to synthesize the missing part of the speech. **(2) Average-Mel.** In this naive baseline, all mel-spectrogram frames for the missing part are replaced by the average of frames in the available segments of ground-truth mel-spectrogram. **(3) Meta-StyleSpeech (Min et al., 2021).** Meta-StyleSpeech is an adaptive multi-speaker TTS method that, given a reference audio from a speaker, can synthesize speech from a given text in the style of that speaker. We use a publicly available pretrained Meta-StyleSpeech model<sup>1</sup>. For each test example, we provide the ground-truth audio (after removing the segment that is to be inserted) as reference audio to Meta-StyleSpeech’s speaker encoder to extract the speaker encoding vector. We synthesize the speech conditioned on the input text-transcript and the extracted speaker encoding vector. Finally, we crop the segment corresponding to the masked phonemes from the synthesized audio and stitch it with the ground-truth audio of the available part. A similar baseline has been used in Borsos et al. (2022); Yin et al. (2022). **(4) Meta-StyleSpeech-Full.** Meta-StyleSpeech-Full is same as the baseline Meta-StyleSpeech except we provide the full ground-truth audio which also includes the segment that we want to synthesize and insert. Note that in many test examples, we provide an audio context of as little as a single word of audio. In such examples, Meta-StyleSpeech-Full gets significant advantage over our proposed method as it has the full ground-truth audio available to model the speaker’s style. This is a favorable setting, not applicable on actual use of the system where the full audio is simply not available.

## Evaluation Metrics

**Mean Opinion Score (MOS).** We selected 15 examples from LibriTTS dev-clean subset. To evaluate our method with different lengths of inserted text, we defined three settings: short, medium and long. We used length of inserted phoneme sequences as less than 10 for short, between 10 and 20 for medium, and greater than 20 for the long setting. We selected five examples for each setting. We collected a set of 45 speech samples: 15 each for Ground-Truth, results of Meta-StyleSpeech baseline and our proposed method. We asked 6 users to rate the naturalness of each speech sample in a five-point scale (1-5) ranging from Bad to Excellent. All speech samples were presented to users in a random order. We average the ratings from all users to get the mean opinion score (MOS) for each method.

**Mel-Cepstral Distortion (MCD).** Although quantitative evaluation of speech insertion task or speech synthesis problem in general is quite challenging, we measure how well our speech insertion results match the ground truth samples using Mel-Cepstral Distortion metric (Kubichek, 1993). It computes the error between the mel-frequency cepstral coefficients (MFCC) of two audio signals. If the two signals are not aligned, as it is the case in the speech insertion task, we align the signals with dynamic time warping and compute the mel-cepstral distortion with the best alignment. We present MCD results on both the dev-clean and the dev-other dataset. Note that MCD only measures how close a predicted speech sample is to the corresponding ground-truth sample and is not a reliable measure for judging the naturalness of the speech sample.

## 4.1 RESULTS

### 4.1.1 USER STUDY

Table 1 shows the results of our user study. Our method outperforms the MetaStyleSpeech baseline in terms of naturalness MOS in all three categories. For short insertion category, MOS of samples from our method is same as that of the ground-truth samples (4.2), indicating that samples synthesized by our method do not have any noticeable temporal inconsistencies. Samples from our method do not deteriorate significantly with the length of inserted text segment, as demonstrated by the 3.97 MOS on long insertion category *c.f.* 4.47 of ground truth, and 2.8 of MetaStyleSpeech.

<sup>1</sup><https://github.com/KevinMIN95/StyleSpeech>

Table 1: Results of the naturalness user study. MSS and GT stand for the Meta-StyleSpeech baseline and ground-truth respectively.

MOS@short			MOS@medium			MOS@long			MOS		
GT	MSS	Ours	GT	MSS	Ours	GT	MSS	Ours	GT	MSS	Ours
4.20	3.50	4.20	4.30	2.80	3.63	4.47	2.80	3.97	4.32	3.03	3.93

Table 2: Comparison of our proposed method’s speech insertion performance with different baselines.

Method	MCD ↓ (clean)	MCD ↓ (other)
<b>GT-Mel+Vocoder</b>	<b>0.3725</b>	<b>0.3758</b>
Average-Mel	0.9149	0.9731
MetaStyleSpeech	0.8328	0.9178
MetaStyleSpeech-Full	0.6061	0.6334
Ours	<b>0.5790</b>	<b>0.6244</b>

#### 4.1.2 QUANTITATIVE EVALUATION

We compare the speech insertion performance of our method against other baselines on the dev-clean and the dev-other subsets using the MCD metric. Results are shown in Table 2. Our method achieves lowest the MCD among all baselines, *e.g.* 0.5790 *vs.* 0.8328 for MetaStyleSpeech on dev-clean and 0.6244 *vs.* 0.9178 for MetaStyleSpeech on dev-other. This indicates that our method is better at matching the ground-truth spectral characteristics than the other baselines.

The results are still quite far from ground truth MCD, *e.g.* 0.3725 and 0.3758 for dev-clean and dev-other respectively. This also indicates that there is a non-trivial scope of improvement for the task.

**Ablation Study** Kindly see Appendices for ablation study.

#### 4.2 QUALITATIVE EXAMPLES

We present a sample of speech insertion results from both dev-clean and the dev-other subset in the supplementary material. The ground-truth, Meta-StyleSpeech and Meta-StyleSpeech-Full baselines are also included for comparison. We observed that our method is able to synthesize intelligible and natural sounding speech segments. Our method is also able to match the speaker’s characteristics from the short audio context available. When comparing with the ground-truth samples, we found that results from our method while natural-sounding do not match the expressiveness of ground-truth examples.

We also compare few examples with different combination of losses. We observed that the model trained only with the L1 loss shows significant robotic artifacts in the synthesized speech samples. And addition of the local and global discriminator based losses greatly improve the naturalness of the synthesized speech samples.

### 5 CONCLUSION

We proposed a novel method for dynamic length speech insertion using cross-modal attention and adversarial local, global and style losses. The method is capable of maintaining the speaker characteristics in the generated parts of the speech. We evaluate the method on a large scale public dataset, LibriTTS, and show that it performs better than state of the art methods adaptive TTS methods. We also provide a user study showing that subjectively human participants also favour the output of the proposed network. Finally we also provide numerous qualitative results comparing our method with state of the art existing methods.

## REFERENCES

- Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. *Advances in Neural Information Processing Systems*, 31, 2018.
- Zalán Borsos, Matt Sharifi, and Marco Tagliasacchi. Speechpainter: Text-conditioned speech inpainting. *arXiv preprint arXiv:2202.07273*, 2022.
- Edresson Casanova, Christopher Shulby, Eren Gölge, Nicolas Michael Müller, Frederico Santos de Oliveira, Arnaldo Candido Junior, Anderson da Silva Soares, Sandra Maria Aluisio, and Moacir Antonelli Ponti. Sc-glowtts: an efficient zero-shot multi-speaker text-to-speech model. *arXiv preprint arXiv:2104.05557*, 2021.
- Yutian Chen, Yannis Assael, Brendan Shillingford, David Budden, Scott Reed, Heiga Zen, Quan Wang, Luis C Cobo, Andrew Trask, Ben Laurie, et al. Sample efficient adaptive text-to-speech. *arXiv preprint arXiv:1809.10460*, 2018.
- Seungwoo Choi, Seungju Han, Dongyoung Kim, and Sungjoo Ha. Attention: Few-shot text-to-speech utilizing attention-based variable-length embedding. *arXiv preprint arXiv:2005.08484*, 2020.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.
- Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31, 2018.
- Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.
- Robert Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE pacific rim conference on communications computers and signal processing*, volume 1, pp. 125–128. IEEE, 1993.
- Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *arxiv. arXiv preprint arXiv:1611.04076*, 2016.
- Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, volume 2017, pp. 498–502, 2017.
- Dongchan Min, Dong Bok Lee, Eunho Yang, and Sung Ju Hwang. Meta-stylespeech: Multi-speaker adaptive text-to-speech generation. In *International Conference on Machine Learning*, pp. 7748–7759. PMLR, 2021.
- Pierre Prablanc, Alexey Ozerov, Ngoc QK Duong, and Patrick Pérez. Text-informed speech inpainting via voice conversion. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 878–882. IEEE, 2016.
- Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*, 2020.
- Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

Chuanxin Tang, Chong Luo, Zhiyuan Zhao, Dacheng Yin, Yucheng Zhao, and Wenjun Zeng. Zero-shot text-to-speech for text-based insertion in audio narration. *arXiv preprint arXiv:2109.05426*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Dacheng Yin, Chuanxin Tang, Yanqing Liu, Xiaoqiang Wang, Zhiyuan Zhao, Yucheng Zhao, Zhiwei Xiong, Sheng Zhao, and Chong Luo. Retrievertts: Modeling decomposed factors for text-based speech insertion. *arXiv preprint arXiv:2206.13865*, 2022.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

## A IMPLEMENTATION DETAILS

We resample all audio signals in the LibriTTS dataset to a sampling rate of 22050. We trim longer audio files so that all audio signals have a maximum length of 10 seconds. We extract mel-spectrograms  $\mathbf{X}$  from resampled audio files with a hop size of 256 samples, FFT window size of 1024 samples and  $d_{\text{mel}} = 80$  frequency bins.

We convert all text transcripts into phoneme sequences  $\mathbf{T}$  and get the phoneme-level alignments using the Montreal Forced Alignment (MFA) (McAuliffe et al., 2017). We segment the transcript by randomly sampling a sequence of up to 7 words from the text transcripts and assigning the corresponding phonemes as  $\mathbf{T}^I$ . The phoneme sequences before and after  $\mathbf{T}^I$  become  $\mathbf{T}^B$  and  $\mathbf{T}^A$  respectively. We use the phoneme-level alignment estimated by MFA to get the segmentation of  $\mathbf{X}$  as  $(\mathbf{X}^B, \mathbf{X}^I, \mathbf{X}^A)$ . We remove the segment  $\mathbf{X}^I$  from  $\mathbf{X}$  to get the input  $\mathbf{X}^{\text{in}}$ . For test utterances, the phoneme and mel-spectrogram segmentation is performed in advance and frozen, while for training set, it is done randomly in each training iteration.

We follow the same architecture for phoneme encoder, decoder, and variance adaptor as in FastSpeech2. We use identical architecture for the phoneme and the audio encoder. Specifically, the encoders and decoders consist of 4 FFT blocks, while predictors in variance adaptors are two-layer 1-D convolutional networks with ReLU activation and layer normalization. All hidden embeddings in FFT blocks, positional embeddings and segment embeddings have dimensionality of  $d = 256$ . In our cross-modal attention block, we use  $h = 2$  heads and for each head, the dimensions of key, query and value vectors is  $d_k = d_v = 128$ .

For both global and local discriminators we use a modified ResNet18 architecture (He et al., 2016). We remove the final softmax layer and change the output dimension of the final fully-connected layer to 1. For the global discriminator, the input dimensions are  $d_g \times d_{\text{mel}}$ , where  $d_g$  is the number of frames in the largest spectrogram in the input batch. Shorter mel-spectrograms in the batch are centered and padded with  $\log(\epsilon)$  to fit into a  $d_g \times d_{\text{mel}}$  matrix. The input dimensions for local discriminator are  $d_l \times d_{\text{mel}}$ . The number of frames in the input windows is fixed to  $d_l = 96$ . The windows are only sampled from  $\mathbf{X}^I$  or  $\hat{\mathbf{X}}^I$  segments. Windows are sampled with a hop length of 48 frames. If the mel-spectrogram segment has less frames then it is centered and padded to fit the dimensions. For feature matching losses, we extract features at the end of each of the five convolution blocks (conv1-conv5) and the average pool layer.

Architecture for the style extractor is also adapted from ResNet18. The final softmax layers is removed and the output dimension is changed to  $d_{\text{style}} = 512$ . The dimensions of input mel-spectrogram windows are same as it is for the local discriminator i.e.  $96 \times d_{\text{mel}}$ . To create synthesized examples, we only sample windows from the  $\hat{\mathbf{X}}^I$  segments as we are mostly interested in improving the quality of the inserted segments. For real examples, however, we sample windows randomly from full spectrograms to increase the diversity of positive pairs.

In the first phase of training, we train our model with  $\mathcal{L}_{\text{rec}}$  (12) for 75k iterations with a batch-size of 16 utterances. In the second phase, we train with all losses combined as given by (21) for 125k iterations. We use  $\lambda_1 = 2$  in the L1 loss, weights  $\lambda_{\text{feat},l} = \lambda_{\text{feat},g} = 2$  for feature matching losses,

Table 3: Ablation study. The local and global loss refers to the adversarial and feature matching losses based on the *local* and *global* discriminator respectively. The *style* loss refers to the triples loss based on the style extractor. CMA refers to the Cross-Modal Attention.

Method	MCD ↓ (clean)	MCD ↓ (other)
Ours	<b>0.5790</b>	0.6244
Ours – <i>local</i> loss	0.5992	0.6375
Ours – <i>global</i> loss	0.6016	0.6166
Ours – <i>style</i> loss	0.5923	0.6268
Ours (Only L1)	0.5884	<b>0.6097</b>
Ours – Segment Embs.	0.6007	0.6199
Ours – CMA + Speaker Encoder	0.5892	0.6125

weights  $\lambda_{adv,l} = \lambda_{adv,g} = 1$  for LSGAN losses, and  $\lambda_{style} = 2$  for the style matching loss. All models are trained with the Adam optimizer. We use a learning rate of 0.001 for the discriminators and the style extractor. Following FastSpeech2, we use a step-wise learning schedule for training our RephraseTTS model, where we reduce the initial learning rate of 0.0625 by a factor of 0.3 after 75k, 125k and 150k iterations. On a single Nvidia GTX 1080 Ti GPU, our model takes 30 hours to train.

We use a pretrained HifiGAN model (Kong et al., 2020) to convert the predicted mel-spectrograms into waveforms.

## B ABLATION STUDY

In the first part of our ablation experiment, we train our method with different combination of losses and compare them with the MCD metric. The results are shown in the top block of Table 3. On the dev-clean subset, we observed that removing any of the losses increases the MCD. However, on the dev-other subset, which is a more challenging subset of the two, we observed that removing some of the losses improves the MCD.

On qualitative assessment of the synthesized speech samples, we found that the model trained without any adversarial or style losses (Ours (only L1) in Table 3) shows significant artifacts and all proposed losses help in reducing the artifacts and the naturalness of the samples. We encourage readers to see the supplementary material for qualitative examples with different combination of losses.

In the second part (Table 3 bottom), we test the effectiveness of some of the architectural choices we made for our RephraseTTS model. We observed that removing the segment embeddings from phoneme and audio encoder increases the MCD on the dev-clean subset. To investigate the effectiveness of cross-modal attention module, we train a model where we replace it by a speaker encoder. We use a 1D ConvNet as the speaker encoder that extracts the global speaker characteristics from  $\bar{X}^{in}$  in form of a single style vector and adds them to the phoneme representation  $\bar{T}$ . We found that replacing the cross-modal attention with the speaker style encoder worsens the MCD on the dev-clean subset. On dev-other subset, however, we see an opposite trend, the MCD metric improves after removing both segment embeddings and cross-modal attention.