A. Rhythm Disentanglement from an Information-Theoretic Perspective

In this appendix, we will provide further explanation on why Equation (3) is necessary for rhythm disentanglement, *i.e.* reducing $I(\boldsymbol{R}; \tilde{\boldsymbol{Z}})$. Before the formal analysis, we state the following assumption:

$$H(\boldsymbol{S}|\boldsymbol{X}) = 0, \quad H(\boldsymbol{R}|\boldsymbol{X}) = 0, \tag{12}$$

which means that the phonetic symbol and the rhythm information is completely determined given the speech utterance. In the following, we will show two theorems, which serve as an elaboration of the brief discussion below Equation (3).

Theorem 1. *Assume that Equations (2) and (12) hold. If $\forall \boldsymbol{X}(t)$ and $\boldsymbol{X}'(t)$ with the same content information $\boldsymbol{S}$, but with different rhythm information, $\boldsymbol{R}$ and $\boldsymbol{R}'$ respectively,*

$$Pr(\tilde{\boldsymbol{Z}} = \tilde{\boldsymbol{Z}}') = 0, \tag{13}$$

then

$$I(\boldsymbol{R}; \tilde{\boldsymbol{Z}}) = I(\boldsymbol{R}; \boldsymbol{X}). \tag{14}$$

Proof. According to (13), when $\boldsymbol{S}$ given, there forms an injective mapping from $\boldsymbol{R}$ to $\tilde{\boldsymbol{Z}}$, and thus

$$H(\boldsymbol{R}|\tilde{\boldsymbol{Z}}, \boldsymbol{S}) = 0. \tag{15}$$

On the other hand, according to Equation (2)

$$I(\boldsymbol{S}; \tilde{\boldsymbol{Z}}) = I(\boldsymbol{S}; \boldsymbol{X}) = H(\boldsymbol{S}) - H(\boldsymbol{S}|\boldsymbol{X}) = H(\boldsymbol{S}), \tag{16}$$

where the last equality is due to Equation (12), and thus

$$H(\boldsymbol{S}|\tilde{\boldsymbol{Z}}) = I(\boldsymbol{S}; \tilde{\boldsymbol{Z}}) - H(\boldsymbol{S}) = 0. \tag{17}$$

Plug Equation (17) into Equation (15), we have

$$H(\boldsymbol{R}|\tilde{\boldsymbol{Z}}) = 0. \tag{18}$$

Therefore

$$\begin{aligned} I(\boldsymbol{R}; \tilde{\boldsymbol{Z}}) &= H(\boldsymbol{R}) - H(\boldsymbol{R}|\tilde{\boldsymbol{Z}}) \\ &= H(\boldsymbol{R}) \\ &= H(\boldsymbol{R}) - H(\boldsymbol{R}|\boldsymbol{X}) \\ &= I(\boldsymbol{R}; \boldsymbol{X}), \end{aligned} \tag{19}$$

where the second line is given by Equation (18); the third line is given by Equation (12). □

Theorem 2. *Assume that Equations (2) and (12) hold. If $\forall \boldsymbol{X}(t)$ and $\boldsymbol{X}'(t)$ with the same content information $\boldsymbol{S}$, but with different rhythm information, $\boldsymbol{R}$ and $\boldsymbol{R}'$ respectively,*

$$Pr(\tilde{\boldsymbol{Z}} = \tilde{\boldsymbol{Z}}') = 1, \tag{20}$$

then

$$I(\boldsymbol{R}; \tilde{\boldsymbol{Z}}) = I(\boldsymbol{R}; \boldsymbol{S}). \tag{21}$$

Proof. According to (20), when $\boldsymbol{S}$ given, $\tilde{\boldsymbol{Z}}$ is a constant regardless of the value of $\boldsymbol{R}$, and thus

$$H(\boldsymbol{R}|\tilde{\boldsymbol{Z}}, \boldsymbol{S}) = H(\boldsymbol{R}|\boldsymbol{S}). \tag{22}$$

Plug Equation (17) into Equation (22), we have

$$H(\boldsymbol{R}|\tilde{\boldsymbol{Z}}) = H(\boldsymbol{R}|\boldsymbol{S}). \tag{23}$$

Therefore

$$\begin{aligned} I(\boldsymbol{R}; \tilde{\boldsymbol{Z}}) &= H(\boldsymbol{R}) - H(\boldsymbol{R}|\tilde{\boldsymbol{Z}}) \\ &= H(\boldsymbol{R}) - H(\boldsymbol{R}|\boldsymbol{S}) \\ &= I(\boldsymbol{R}; \boldsymbol{S}). \end{aligned} \tag{24}$$

□

B. Additional Algorithm Details

In this appendix, we will cover the additional algorithm details of AUTOPST. Our implementation is available here `https://github.com/auspicious3000/AutoPST`.

B.1. Self-Expressive Autoencoder (SEA)

SEA aims to derive an representation $\boldsymbol{A}(t)$, which is very similar among similar frames, and very disimilar among disimilar frames. SEA consists of one encoder, which derives $\boldsymbol{A}(t)$ from input speech, and two decoders. One decoder reconstructs the input based on $\boldsymbol{A}(t)$. The other decoder reconstructs the input based on another representation $\boldsymbol{B}(t)$, which is computed as follows

$$\boldsymbol{B}(t) = \sum_{t' \neq t} \left(\frac{\boldsymbol{A}^T(t')\boldsymbol{A}(t)}{\|\boldsymbol{A}(t')\|_2 \|\boldsymbol{A}(t)\|_2} \right) \cdot \boldsymbol{A}(t'). \tag{25}$$

The entire pipeline is trained jointly to minimize the reconstruction loss of both decoders. The intuition behind this self-expressive mechanism is that in order to achieve good reconstructions at both decoders, $\boldsymbol{A}(t)$ should be similar to $\boldsymbol{B}(t)$. According to Equation (25), $\boldsymbol{A}'(t)$ is essentially a linear combination of the representation of all the other frames, and the combination weight is the cosine similarity. If frame t' is dissimilar to the current frame t to be expressed, the weight has to be close to zero, otherwise $\boldsymbol{B}(t)$ would be made dissimilar to $\boldsymbol{A}(t)$; if frame t' is similar to the current frame t to be expressed, the weight has to be close to one to ensure it is contributing sufficiently to $\boldsymbol{B}(t)$.

B.2. F0 and UV Conditioning

One of the biggest challenges of AUTOPST is to infer the F0 contour and voiced/unvoiced (UV) states of the utterances based solely on the input MFCC features, because it is very hard for the system to elicit lexical, semantic and syntactic information without text transcriptions. To mitigate this problem, we introduce two optional conditioning, one on the input F0 contour and one on UV states. The sequence being conditioned upon is concatenated with the encoder output along the channel dimension *before* being sent to the similarity-based resampling module. The concatenation is valid because the conditioning sequence has the same temporal length as the encoder output, which are both equal to the temporal length of the input.

Although the F0 conditioning will largely resolve the ambiguity of reconstructing the F0 contour, it will make the algorithm unable to convert the F0 aspect of prosody, which is undesirable in many applications. On the other hand, UV conditioning can still resolve the ambiguity of reconstructing the F0 information to some extent, while maintaining AUTOPST's ability to disentangle the pitch aspect. In practice, we can choose F0 conditioning, UV conditioning, or neither depending on different applications.

B.3. Domain Identity Conditioning

According to Equation (4), the decoder takes the domain identity, D, as a second input. This is achieved by first feeding D to a feedforward layer. Then, the feedforward layer is appended to the first time step of the encoder output sequence, $\tilde{\boldsymbol{Z}}(t)$, as well as to the first time step of the memory of the decoder (Our decoder is a Transformer; the memory is the output of the Transformer encoder). In other words, the total length of the encoder output and the memory will increase by one after the appending, with the first time step being the appended domain identity. Since the decoder is a Transformer, it can elicit the domain identity information by attending to the first time step.

B.4. Output and Losses

In addition to outputting the reconstructed spectrogram, $\hat{\boldsymbol{S}}(t)$, the AUTOPST decoder also outputs a stop token prediction. Stop token, denoted as $P(t)$, guides when the sequential spectrogram generation should stop. It is a scalar sequence that equals zero at time steps within the lengths of the ground truth spectrogram, and equals one at time steps after the ground truth spectrogram has ended. Denote the predicted stop token as $\hat{P}(t)$. Then the total loss function consists of the ℓ_2 loss for spectrogram prediction and the cross-entropy loss for stop token prediction:

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_{train}\Big[\sum_{t=1}^{T}\|\hat{\boldsymbol{S}}(t)-\boldsymbol{S}(t)\|_2^2+\\ &\sum_{t=1}^{T+k} wP(t)\log\hat{P}(t)+(1-P(t))\log(1-\hat{P}(t))\Big].\end{aligned} \tag{26}$$

The expectation is taken over the training set. T denotes the total length of the spectrogram. Notice that in the second summation, t runs to from 1 to $T+k$, which k time steps longer than the spectrogram. This is because the ground truth stop token is always zero for $t \leq T$. It is only equal to one for $t > T$. So we set k steps for the positive samples.

Nevertheless, the negative examples still occur much more often than the positive examples. To fix the unbalanced label problem, we add a positive weight, w, to the positive class.

C. Experiment Details

In this appendix, we will cover the additional details of our experiments.

C.1. Architecture

The AUTOPST encoder is a simple 8-layer 1D convolutional network, where each layer uses 5×1 filters with 1 stride, SAME padding and ReLU activation. GroupNorm (Wu & He, 2018) is applied to every layer. The number of filters is 512 for the first five layers, and the last three layers have 128, 32, and 4 filters respectively. The AUTOPST decoder is a Transformer (Vaswani et al., 2017), which has four encoder layers and four decoder layers. The model dimension is 256 and the number of heads is eight.

C.2. Training

We implement our model using Pytorch 1.6.0, and we train our model on a single Tesla V100 GPU using Adam optimizer. Synchronous training takes 3×10^5 steps with a batch size of 4. Asynchronous training takes 6×10^5 steps with a batch size of 4. We use Pre-LN (Xiong et al., 2020) without the warm-up stage.

C.3. Datasets

For the VCTK dataset, our test set consists of parallel utterances from 24 speakers. In order to identify speakers with the fastest and slowest speech rates, we take the average of the log duration of the test utterances. Since all the utterances are spoken in parallel by all of the speakers, the speaker with the smallest log duration can be considered as the fastest speaker, and the speaker with the largest log duration can be considered as the slowest speaker. We use the log duration instead of duration because the differences between the average log duration can be nicely interpreted as the average percentage difference in duration of the same sentence uttered by the two speakers. We then select the two fastest speakers (P231 and P239), and the two slowest speakers (P270 and P245) for our main evaluation. For further evaluation, we select two speaker pairs with smaller rhythm differences, one with speakers ranking 25% and 75% (P244 and P226) in speech rate; the other with ranking 40% and 60% (P240 and P256) in speech rate. For the Emo-VDB dataset, we follow a similar protocol of finding the fastest emotion, which is neutral, and the slowest emotion, which is sleepy, for our evaluation.

C.4. Subjective Evaluation

For the subjective evaluation, 18 sentences are generated for each fast-slow speaker pair (9 for fast-to-slow conversions and 9 for slow-to-fast conversions), and there are four fast-slow speaker pairs, summing to 72 utterances for each algorithm. Each utterance is assigned to five subjects. When evaluating voice similarity, the subjects are explicitly asked to focus only on voice but not on prosody; when evaluating prosody similarity, the subjects are asked to focus only on prosody but not on voice; when evaluating the overall speaker similarity, the subjects are asked to pay attention to all the aspects of speech, including voice and prosody.

For each test, the reference utterances (one from the source speaker and one from the target speaker) are randomized and named speaker 1 and speaker 2. The reference utterances are different from the test utterance in terms of content. The subjects are asked to assign a score of 1-5 on whether the aforementioned aspects sound more similar to speaker 1 or speaker 2, with 1 meaning completely like speaker 1 and 5 meaning completely like speaker 2. These scores are then converted to the similarity between the source and target speakers, with 1 meaning completely like the source speaker and 5 meaning completely like the target speaker.

D. Additional Experiment Results

In this section, we will present some additional visualization and ablation study results.

D.1. Similarity-based Resampling Visualization

In order to intuitive show the effect of our similarity-based resampling module, we design the following experiment. We first train a variant of AUTOPST *without* the random resampling module, so that it can generate a spectrogram that is synchronous with the hidden representations. Then, we performed the similarity-based random resampling on the hidden representation, and generate a *time-synchronous* spectrogram from the resampled representations. In this way, we can intuitively see how much each segment is being shortened/lengthened by observing the time-synchronous spectrogram.

Figure 13 shows the reconstructed spectrograms from the resampled code of the utterance *"Please call Stella"*. The left figures corresponds to the downsampling case, with τ dropping from 0.98 down to 0.9. The right figures corresponds to the upsampling case, with τ increasing from 1.02 up to 1.1. The top figure (Figure 13(a)) is the reference spectrogram without resampling. There are two observations. First, the total length of the code decreases as τ decreases, and increases as τ increases. Second, the relatively steady segments get stretched or shortened most, such as the *"a"* segment in the second word.

Table 2. The relative duration difference with respect the original utterance. F2S denotes fast-to-slow conversion; S2F denotes slow-to-fast conversion. "No Up" denotes AUTOPST with upsampling removed; "No Down" denotes AUTOPST with downsampling removed. The numbers outside the parentheses represent the average; the numbers in the parentheses represent the standard deviation. The desired outcome should be a positive average for fast-to-slow and a negative average for slow-to-fast.

	AUTOPST	No Up	No Down
F2S	26.53 (23.05)	32.79 (12.68)	-13.65 (6.46)
S2F	-16.95 (23.70)	13.47 (21.96)	-27.46 (5.00)

D.2. Upsampling v.s. Downsampling

AUTOPST adopts both similarity-based upsampling (Section 4.2) and similarity-based downsampling (Section 4.4). This section shows why both are necessary. In particular, we trained two variants of AUTOPST, one without upsampling, and one without downsampling. All the other settings remain the same. We then perform fast-to-slow and slow-to-fast conversions the same way as in Section 5.3. For each conversion, we compute the relative duration difference with respect to the *original source speech*, *i.e.*

$$\begin{aligned}\text{F2S Relative Duration Diff} &= (L_{F2S} - L_{source})/L_{source}\\ \text{S2F Relative Duration Diff} &= (L_{S2F} - L_{source})/L_{source}.\end{aligned} \tag{27}$$

Note that this is different from the relative duration difference computed in Equation (11). If an algorithm truly converts rhythm to the correct direction, it should have a positive F2S relative duration difference and a negative S2F relative duration difference.

Table 2 shows the relative duration differences. As can be seen, AUTOPST can correctly change the rhythm to the desired direction. However, without either of the resampling module, the rhythm conversion becomes incorrect. If upsampling is removed, both fast-to-slow and slow-to-fast will increase the duration. If downsampling is removed, both fast-to-slow and slow-to-fast will decrease the duration. One possible explanation for this is that random resampling is only enforced during training. During testing, the random resampling will be removed (equivalent to the $\tau = 1$ case). If either downsampling or upsampling is removed, the test case, $\tau = 1$, becomes a corner case, undesirably passing a duration bias. By having both upsampling and downsampling, we can ensure $\tau = 1$ is a well-represented mode among the training instances.

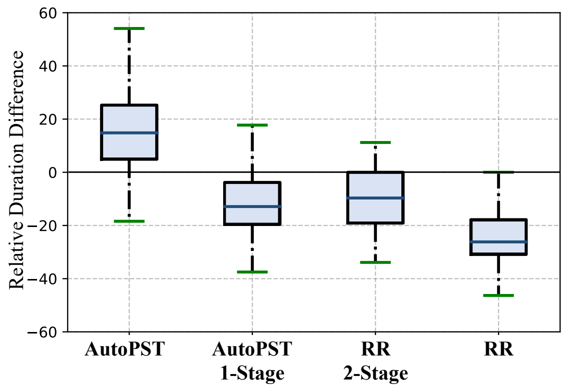


Figure 9. Ablation study on two-stage training using the same relative duration difference experiment shown in Figure 7. AUTOPST 1-Stage denotes the AUTOPST algorithm trained in an end-to-end manner (without two-stage training). RR 2-Stage denotes the RR baseline with the two-stage training.

D.3. Removing Two-Stage Training

As discussed, there are two mechanisms that promote prosody disentanglement. The first is similarity-based resampling; the second is two-stage training. In this section, we will explore how much each mechanism contributes to the performance advantage of AUTOPST. In particular, we implement two variants of the algorithms. The first variant, called AUTOPST 1-Stage, removes the two-stage training of AUTOPST, while all the other settings remain the same as AUTOPST. The second variant, called RR 2-Stage, supplements the RR baseline with two-stage training. We then create the same box plot of relative duration difference as discussed Section 5.3.

Figure 9 shows the results. As can be seen, without either two-stage training or similarity-based random resampling, the performance drops significantly, which implies that both mechanisms are essential for a successful rhythm disentanglement.

D.4. Generalization to Unseen Emotions

As mentioned in Section 5.6, for the emotion conversion experiment, we deliberately remove certain emotion categories for each speaker from the training set. As a result, some speakers do not have training examples of either neutral or sleepy, or both. To examine whether AUTOPST generalize to unseen speakers, we break the AUTOPST samples in Figure 7(d) into three groups. The first group consists of the speaker who has training examples of both neutral and sleepy. The second group consists of speakers who have training examples of only sleepy. The third group consists of the speaker who has training examples of only neutral.

Figure 10 shows the box plot of the relative duration dif-

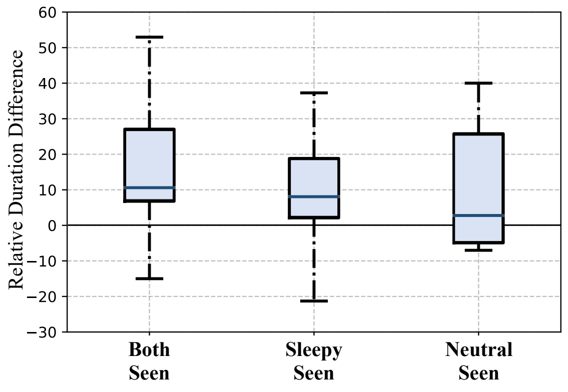


Figure 10. Breakdown of Figure 7(d) into 1) the speaker who has training examples of both neutral and sleepy, 2) the speakers who have training examples of sleepy only; and 3) the speaker who has training examples of neutral only.

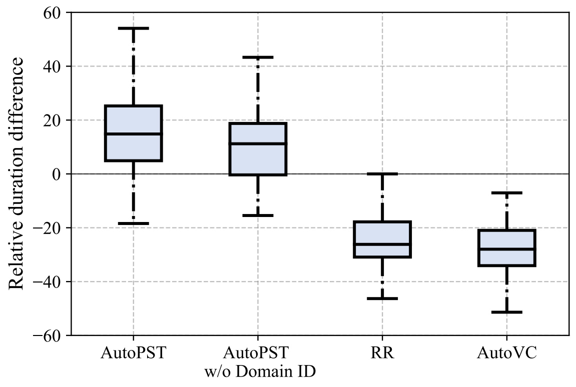


Figure 11. The box plot of the relative duration difference between slow-to-fast conversion and fast-to-slow conversion of the same utterance pairs as used in Figure 7(a). AUTOPST w/o Domain ID denotes AUTOPST variant trained using domain embedding.

ference (same as Figure 7(d)) for these three groups. As can be seen, there is a slight performance advantage if both emotion categories are seen. However, even if there is one unseen emotion, the performance is still pretty competitive, demonstrating good generalizability to unseen emotions.

D.5. Training Domain Embedding

The domain ID is to assumed to present during testing, but we would like to explore the possibility of doing zero-shot conversion. Thus, we trained a variant of AUTOPST where a domain encoder replaces the one-hot domain embedding. During testing, we only need to feed the domain encoder with a target speaker's utterance without needing the domain ID. Figure 11 shows the relative duration difference of this variant, which performs slightly worse than the original AUTOPST due to the increased difficulty, but still significantly better than the baselines.

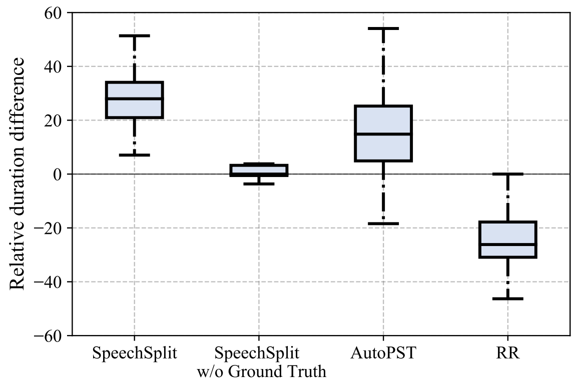
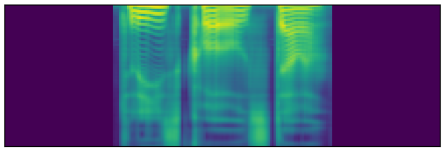


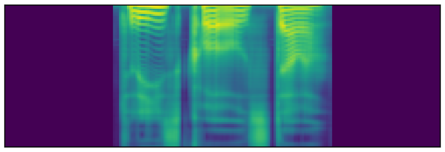
Figure 12. The box plot of the relative duration difference between slow-to-fast conversion and fast-to-slow conversion of the same utterance pairs as used in Figure 7(a). SpeechSplit w/o

D.6. SPEECHSPLIT Baseline

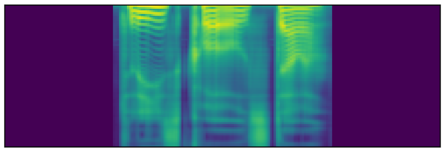
Although SPEECHSPLIT can also perform prosody style transfer, it requires ground truth target rhythm, *i.e.* the target speaker speaking the source utterance. On the other hand, AUTOPST seeks to perform prosody style transfer without the ground truth, which is a much harder task. Nevertheless, we show the relative duration difference of SPEECHSPLIT in Figure 12. Note since the converted rhythm is very close to the ground truth, SPEECHSPLIT is the performance upper-bound of any prosody conversion, including AUTOPST. However, when there is no ground truth available, SPEECH-SPLIT becomes vulnerable. To show this, Figure 12 also shows the result of SPEECHSPLIT with a random target speaker utterance, instead of the ground truth target utterance, fed into its rhythm encoder. As can be observed, the relative duration difference almost completely concentrate around zero, which indicates that SPEECHSPLIT completely fails in this case.



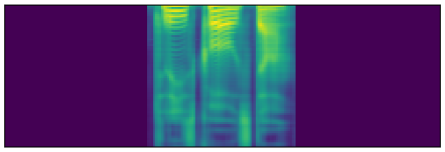
(a) $\tau = 1.0$



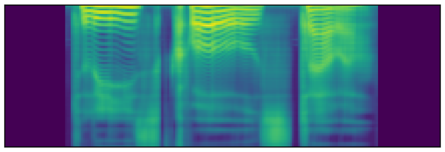
(b) $\tau = 0.98$



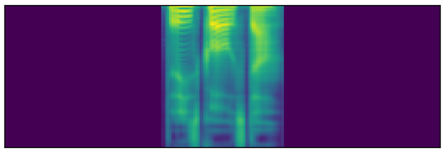
(c) $\tau = 1.02$



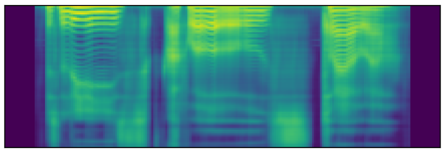
(d) $\tau = 0.96$



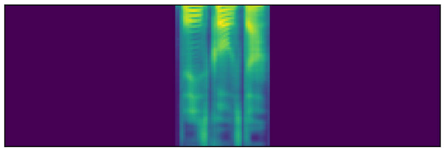
(e) $\tau = 1.04$



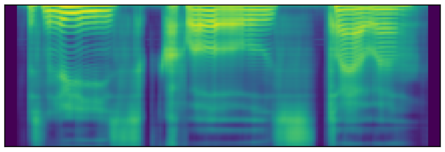
(f) $\tau = 0.94$



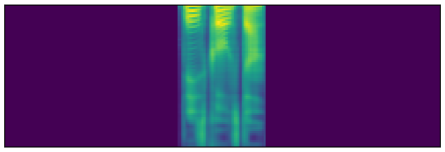
(g) $\tau = 1.06$



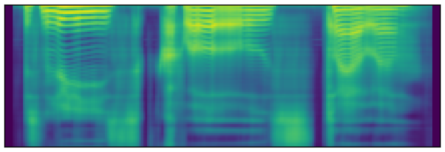
(h) $\tau = 0.92$



(i) $\tau = 1.08$



(j) $\tau = 0.90$



(k) $\tau = 1.10$

Figure 13. Temporally-aligned spectrograms constructed from resampled hidden representation as a visualization of the resampling operations. The utterance is *Please call Stella*.