Zero-Shot Non-Autoregressive TTS Beyond Autoregressive Models Using Soft Alignment Generation and Residual Modeling

Anonymous ACL submission

Abstract

Autoregressive TTS leverages soft alignment generated by the attention mechanism, which provides the decoder with a well-designed context vector. Subsequently, the decoder receives both the semantic representation and the acoustic representation generated at the previous time step. For this reason, autoregressive TTS achieves strong performance. Thus, we propose novel algorithms to bring similar benefits to non-autoregressive TTS. First, we propose a method to distill soft alignments—originally provided by attention in autoregressive models—into a flow matching model trained between mel-spectrograms and text representations. This allows non-autoregressive models to leverage attention-like context vectors without requiring autoregressive decoding. Second, we introduce an invertible encoder, designed based on normalizing flow, to disentangle semantic and residual acoustic representations. The invertible encoder maps the residual information, which is absent in the context vector, closer to a Gaussian distribution. During inference, we can treat the context vector as the semantic representation and Gaussian noise as the acoustic representation. Lastly, to improve zero-shot TTS performance, we propose a prompt-aware lightweight convolution, where the kernel weights are dynamically adjusted for each speech prompt. With the proposed methods, our non-autoregressive TTS model achieves comparable performance to existing autoregressive models.

1 Introduction

Zero-shot text-to-speech (TTS) (Casanova et al., 2022; Wang et al., 2023) aims to synthesize speech that reflects the voice characteristics of unseen speakers at inference time, without requiring any fine-tuning. Recent advances in large language models (LLMs), such as GPT (Achiam et al., 2023) and T5 (Ao et al., 2021), have inspired TTS architectures to adopt similar transformer-based models

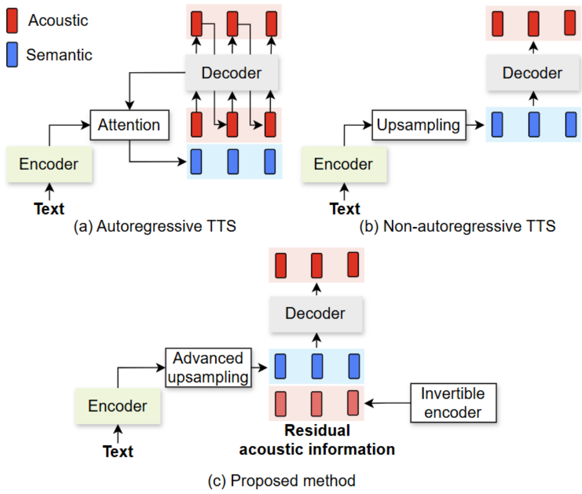

Figure 1: The concepts of our proposed method that supplements missing acoustic information in the semantic representation and refines the existing upsampling process.

and in-context learning strategies. In zero-shot TTS, speaker-specific information is typically captured from a short speech prompt (e.g., 3-second mel-spectrogram), enabling generalization to unseen speakers. This prompt-based paradigm has shown significant performance improvements over traditional speaker embedding approaches. For instance, VALL-E (Wang et al., 2023) leverages a decoder-only transformer and 60K hours of training data to achieve strong zero-shot performance via in-context learning. Most state-of-the-art zero-shot TTS systems (Kim et al., 2024; Lee et al., 2024) adopt autoregressive (AR) (Wang et al., 2017; Shen et al., 2018) architectures due to their ability to model temporal dependencies and leverage rich contextual information. These models generate acoustic features (e.g., mel-spectrograms or codec tokens (Défossez et al., 2022)) sequentially, using previously generated outputs as inputs for subsequent steps. Additionally, attention mechanisms provide soft alignment between text and acoustic features, allowing the decoder to access a fine-grained context vector that has rich context information. Consequently, autoregressive TTS systems typically outperform non-autoregressive (NAR) (Ren et al., 2019, 2020) models in terms of speech quality. Since the AR TTS systems (Neekhara et al., 2024; Battenberg et al., 2024; Kim et al., 2024) try to overcome the instability or low latency of AR, the performance of AR becomes higher.

On the other hands, apart from the general problems arising from the aforementioned AR, NAR models tend to lag behind AR models in performance, mainly due to two reasons: (1) NAR models lack access to previously generated acoustic features during decoding, and (2) most NAR models rely on hard and shallow upsampling techniques (e.g., duration-based duplication) rather than soft, flexible alignment mechanisms. To bridge this gap, as shown in Figure 1, we introduce **R**es**i**dual modeling and **so**ft alignment generation-based NAR-**TT**S that can be **o**n par with performance of AR (RisoTTo) as follows:

Soft Alignment Generation (SAG): In autoregressive TTS, attention mechanisms generate soft alignments between text and acoustic features, allowing the decoder to condition on fine-grained contextual information. However, non-autoregressive models cannot use such attention-based alignment, as acoustic frames are generated in parallel. Instead, they typically rely on hard and shallow upsampling methods. To address this limitation, we introduce SAG, which employs flow matching (Lipman et al., 2022), which uses only text representation, to generate soft alignments between the mel-spectrogram and the text representation. This enables the model to leverage alignment information similar to that of autoregressive attention, enhancing contextual richness during inference.

Invertible Encoder (IE): The invertible encoder disentangles acoustic and semantic information by modeling the residual component between the mel-spectrogram and the context vector. Specifically, it maps the residual acoustic features—those not captured by the semantic representation—into a Gaussian distribution using a normalizing flow. At inference time, acoustic information can be sampled from this distribution, complementing the semantic context and improving synthesis quality.

Prompt-Aware Lightweight Convolution (PAL): Inspired by SC-CNN (Yoon et al., 2023), which improves zero-shot TTS by generating convolutional kernel weights from speaker embeddings, we adopt a lightweight convolutional module whose kernel weights are directly extracted from the speech prompt. This enables prompt-adaptive feature modulation and enhances generalization to unseen speakers in zero-shot settings.

2 Background

2.1 Flow matching with OT path

Flow matching (Lipman et al., 2022) estimates a probability path between data x_1 and prior x_0 distributions. (Lipman et al., 2022) defines optimal transport (OT) path based on Gaussian conditional probability path that forms a straight trajectory between x_0 and x_1. The OT path on time step $t \in [0, 1]$ varies depending on μ_t and σ_t, which can be defined as follows: $\mu_t = tx_1$ and $\sigma_t = 1 - (1 - \sigma_{min})t$. Since a probability distribution on the OT path follows a Gaussian distribution, x_t on time step t can be computed by affine transform as follows: $x_t = tx_1 + (1 - (1 - \sigma_{min})t)x_0$. Consequently, the vector field u_t, which generates a desired probability path, is defined via the ordinary differential equation as follows:

$$\frac{dx_t}{dt} = u_t = x_1 - (1 - \sigma_{min})x_0. \quad (1)$$

Flow matching is trained to predict the vector field u_t corresponding to x_t using mean squared error (MSE) loss function between predicted vector field v_t and the target u_t. Thus, flow matching can generate data from prior distribution by repeating the following equation until the time step t becomes 1 from 0:

$$x_{t+dt} = x_t + v_t dt, \quad (2)$$

where dt is set to $\frac{1}{N}$ and N is the total number of sampling.

2.2 Invertible encoder

A Normalizing flow (NF) network is a type of generative model that uses an inverse function of flow to generate data. Inspired by (Rombach et al., 2020), we construct an NF network based on decoder of GlowTTS (Kim et al., 2020) to extract the latent variable z from the conditional NF network, as illustrated in Figure 2. This conditional NF network takes two inputs: x and c, to generate a conditional data distribution $p(x|c)$ that is normalized to the prior distribution. The log-likelihood of the data distribution $p(x|c)$ is calculated as follows:

$$\log p(x|c) = \log p(z|c) + \log |\det(\frac{\partial f(x)}{\partial x})|, \quad (3)$$

where $p(z|c)$ represents the output of the NF network. To train the NF network, the negative log-likelihood $-\log p(x|c)$ is decomposed into Kullback-Leibler (KL) divergence and entropy as follows:

$$\mathrm{KL}(p(z|c)|q(z)) + H(x|c), \tag{4}$$

where $q(z)$ is the prior distribution (typically standard Gaussian), and $H(x|c)$ denotes the constant data entropy. According to (Alemi et al., 2016), minimizing $\mathrm{KL}(p(z|c)|q(z))$ reduces the mutual information between z and c, effectively disentangling z from c. This allows us to extract residual information z from x, independently of c, since $q(z)$ is selected independently of c. The mutual information $I(z,c)$ between z and c is represented by:

$$\begin{aligned} I(z,c) &= \int p(z,c)\log\frac{p(z,c)}{p(z)p(c)} \\ &= \int p(z,c)\log\frac{p(z|c)}{p(z)} \\ &= \int p(z,c)\log p(z|c) - \int p(z)\log p(z) \\ &\leq \int p(z,c)\log\frac{p(z|c)}{q(z)} = KL(p(z|c)\|q(z)). \end{aligned} \tag{5}$$

3 Method

We propose methods to improve the performance of zero-shot non-autoregressive text-to-speech (TTS) systems. Unlike autoregressive TTS models, non-autoregressive ones face difficulties in generating soft alignments between text and acoustic features due to the absence of autoregressive attention mechanisms and acoustic context during inference. As a result, they typically rely on hard upsampling methods, such as Gaussian upsampling (Donahue et al., 2020), which limits expressiveness and accuracy.

To address this limitation, we introduce a *Soft Alignment Generation (SAG)* network based on flow matching that learns to produce soft alignments between text and mel-spectrograms without access to acoustic features during inference. This allows for more flexible and semantically enriched context modeling. Additionally, to compensate for the lack of acoustic information, we propose an *invertible encoder* that disentangles residual acoustic features from the mel-spectrogram and maps them to a Gaussian distribution. By sampling from this distribution during inference, we can reintroduce acoustic context, bridging the gap between non-autoregressive and autoregressive decoding. Finally, to improve speaker adaptation, we propose a prompt-aware lightweight convolution (LConv) block, which uses speech prompts to modulate the model dynamically in a zero-shot setting.

3.1 Model description

As shown in Figure 2, our non-autoregressive TTS model consists of several modules. The speech prompt encoder receives a speech prompt, which is a randomly segmented 3-second mel-spectrogram extracted from the reference speech, and encodes it into a fixed-size vector s representing speaker characteristics. This vector s is then used in the prompt-aware LConv block, which integrates both text and speaker information. Consequently, the text encoder extracts a speaker-dependent text representation h_s. The representation h_s is upsampled by the Soft Alignment Generation (SAG) network, which comprises a Conv2D-UNet-based flow matching network and an attention mechanism. The upsampled h_s using SAG becomes a context vector c_s, which has the same length as the mel-spectrogram. The context vector c_s serves as a conditional feature for the invertible encoder, which takes the mel-spectrogram as input. Invertible encoder extracts residual information between c_s and mel-spectrogram, and mel decoder predicts the mel-spectrogram conditioned on both the context vector and residual information. Finally, the predicted mel-spectrogram is refined to a higher-quality output using a flow matching-based post-processing network.

Text encoder comprises 6 Transformer blocks and 6 PAL blocks. Both the input and output dimensions of the Transformer and PAL blocks are set to 256. Text encoder is conditioned by s and extracts speaker-dependent text representation h_s

Mel decoder mirrors the architecture of the text encoder, employing the same configuration of Transformer and PAL blocks. In the decoder's the last Transformer blocks, self-attention is replaced with cross-attention (Transformer w/ CA) to incorporate both the speech prompt and the output from the final PAL block. A linear projection layer in the mel decoder maps the 256-dimensional hidden representations to the 80-dimensional mel-spectrogram space. The predicted mel-spectrogram x_{pred} is then compared to the target mel-spectrogram x_{target} using the $l2$ loss as follows: $\mathcal{L}_{\text{mel}} = ||x_{target} - x_{pred}||_2^2$.

Post-Processing network (PostNet) v_{post} is based on flow matching conditioned on a prior distri-

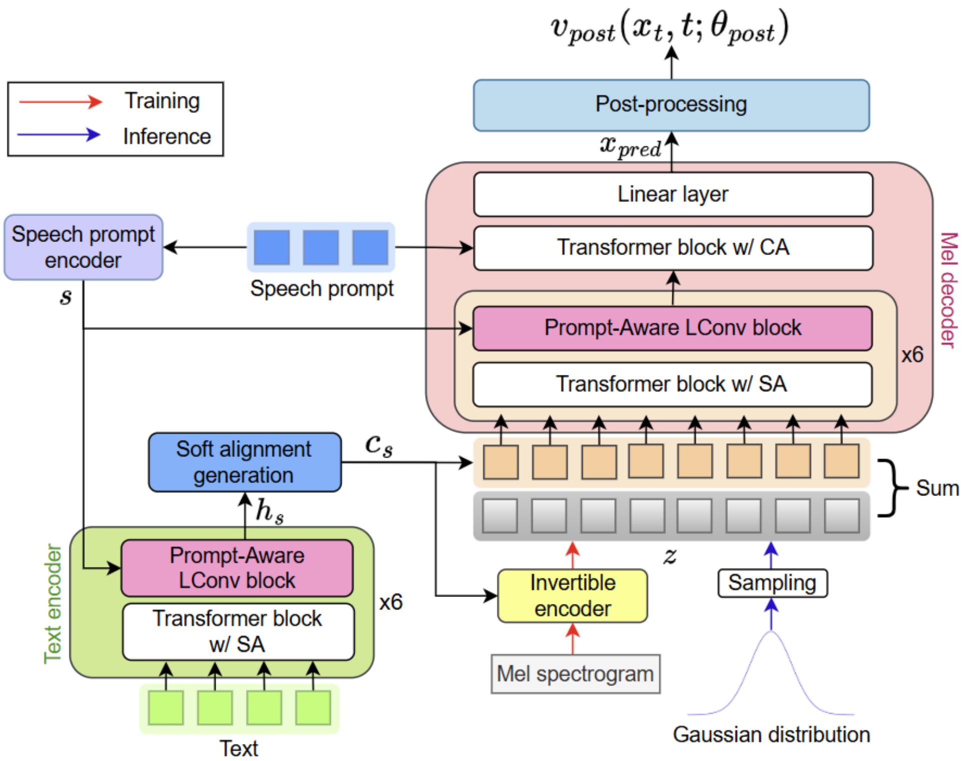

Figure 2: Overall architecture of RisoTTo. CA and SA denote cross and self attentions, respectively.

bution, which is obtained by adding Gaussian noise ϵ sampled from $N(0, I)$ to the predicted mel-spectrogram x_{pred}. This results in a prior distribution defined as $N(x_{pred}, I)$. The vector field for the flow matching-based PostNet is then formulated based on this prior as follows:

$$u_t^{post} = x_{target} - (1 - \sigma_{\min})(x_{pred} + \epsilon). \quad (6)$$

v_{post} is trained by $\mathcal{L}_{\text{post}}$ as follows:

$$\mathcal{L}_{\text{post}} = ||u_t^{post} - v_{post}(x_t, t; \theta_{post})||, \quad (7)$$

where θ_{post} is learnable parameters, and PostNet adopts the same Conv1D-UNet-based flow matching architecture as employed in Matcha-TTS (Mehta et al., 2024). To extract speaker information, we utilize a speech prompt encoder that generates a fixed-size embedding, following the reference encoder design of Grad-TTS (Popov et al., 2021). As discussed previously, such fixed-size representations may be less effective than directly leveraging the speech prompt for in-context learning. To address this limitation, we further incorporate in-context learning by using the speech prompt within the mel decoder. Additional details regarding other modules are provided in following sections.

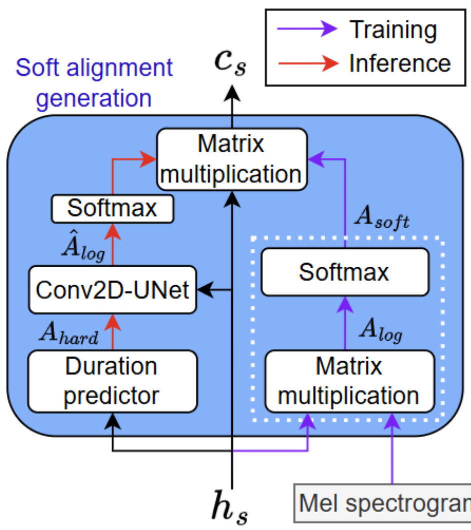

Figure 3: Architecrue of soft alignment generation network. $\hat{A}_{log}$ and white dotted box denote reconstructed A_{log} and attention mechanism, respectively.

3.2 Soft Alignment Generation

As shown in Figure 3, soft Alignment Generation (SAG) consists of an attention mechanism, a Conv2D-UNet-based flow matching network v_{SAG}, and a duration predictor. The attention mechanism computes the matrix multiplication between the speaker-dependent text representation h_s and the mel-spectrogram, providing a soft alignment that maximizes the speech likelihood of the TTS model, such as autoregressive TTS models:

$$\text{Attention}(h_s, x_{target}) = \text{softmax}(x_{target}h_s^T)h_s = c_s, \quad (8)$$

where c_s denotes the context vector obtained by upsampling h_s to match the length of the mel-spectrogram x_{target}. As shown in Figure 3, the product $x_{target}h_s$ forms A_{log}, which serves as the target for the v_{SAG}. We define a vector field from A_{hard} to A_{log} via the following ordinary differential equation (ODE):

$$u_t^{SAG} = A_1 - (1 - \sigma_{min})A_0, \quad (9)$$

where A_1 and A_0 denote A_{log} and $A_{hard} + \epsilon$, respectively. The hard alignment matrix A_{hard} is obtained from A_{log} using monotonic alignment search (Kim et al., 2020). The network v_{SAG} is trained to predict the vector field by minimizing the $\mathcal{L}_{\text{SAG}}$ between the predicted vector field $v_{SAG}(A_t, h_s, t; \theta_{SAG})$ and the target vector field u_t:

$$\begin{aligned} \mathcal{L}_{\text{SAG}} = \left\| u_t^{\text{SAG}} - v_{\text{SAG}}(A_t, h_s, t; \theta_{\text{SAG}}) \right\|_2^2, \\ (A_t = tA_1 + (1 - (1 - \sigma_{\text{min}})t)\, A_0) \end{aligned} \quad (10)$$

where θ_{SAG} denotes the learnable parameters of the v_{SAG}. During inference, the soft alignment matrix A_{soft} can be obtained by applying v_{SAG} using only h_s and A_{hard}, which is predicted from the duration predictor. The vector h_s has 256 dimensions and length N corresponding to the phoneme sequence. It is repeated T times such as (Elias et al., 2021a), where T is the length of the mel-spectrogram. Thus, the repeated h_s has a dimension of $N \times T \times 256$ and is concatenated with A_t, resulting in a feature of dimension $N \times T \times 257$, which is then fed into v_{SAG}. The network then predicts the vector field and generates A_{soft} from A_{hard} by iteratively applying

$$A_{t+dt} = A_t + v_{SAG}(A_t, h_s, t; \theta_{SAG})dt, \quad (11)$$

where $dt = \frac{1}{N_{SAG}}$ and N_{SAG} is the total number of sampling steps for v_{SAG}, with t progressing from 0 to 1. v_{SAG} is implemented as a Conv2D-based U-Net[1], with the input channel size set to 257. The number of feature channels in each block is reduced by a factor of 8 compared to the original implementation.

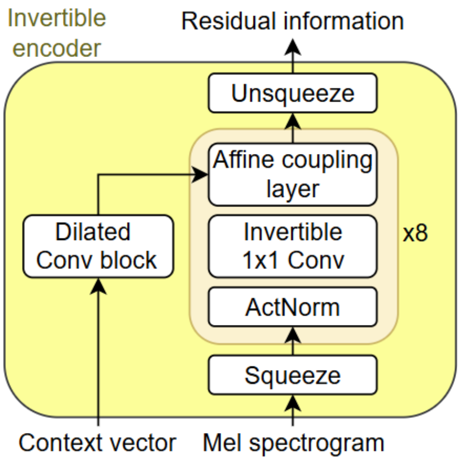

Figure 4: Normalizing flow-based invertible encoder. The residual information is projected from 80 to 256 dimensions to enable summation with the context vector.

3.3 Invertible Encoder

As mentioned in subsection 2.2, the invertible encoder models the residual information z between the input and the conditional feature. Therefore, we utilize it to extract acoustic information absent from the context vector, by processing the mel-spectrogram. The architecture of the invertible encoder is illustrated in Figure 4, which is normalizing flow (Kingma and Dhariwal, 2018). The normalizing flow takes the mel-spectrogram x_{target} as input and uses the context vector c_s as a conditional feature. Accordingly, we minimize the mutual information between c_s and z as follows:

$$\mathcal{L}_{\text{NF}} = \text{KL}(p(z|c_s)|q(z)), \quad (12)$$

where $q(z)$ denotes the prior distribution of the invertible encoder. By minimizing $\mathcal{L}_{\text{NF}}$, we align z with the prior distribution. When $\mathcal{L}_{\text{NF}}$ is minimized, the mutual information between the residual information z and the context vector c_s is also minimized according to Eq. (5) as follows:

$$I(z, c_s) \leq \mathcal{L}_{\text{NF}}. \quad (13)$$

Consequently, z captures acoustic information that is not contained in c_s but exists in the mel-spectrogram. This approach compensates for the insufficient information present in c_s.
During inference, the invertible encoder is not required; instead, residual information can be sampled directly from the prior distribution. We set the prior distribution $q(z)$ to a Gaussian distribution $\mathcal{N}(0, 1)$. Since the decoder of RisoTTo is trained

[1] https://github.com/milesial/Pytorch-UNet

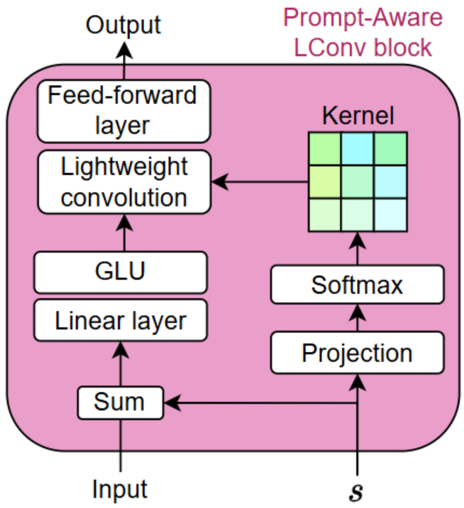

Figure 5: Architecture of prompt-aware lightweight convolution block.

with z distributed according to this Gaussian prior, similar effects can be observed by using Gaussian noise sampled from the prior distribution, as empirically demonstrated in prior numerical work (Lee et al., 2022; Li et al., 2025; Lee and Kim, 2019; Lee et al., 2020). This is conceptually similar to the use of z in VAEs, where the latent variable carries compressed but informative characteristics of x_{target}. However, a known limitation of VAE (Kingma et al., 2013) is that the range of information extraced is inherently dependent on dimension of z. In contrast, our method effectively models the residual information between c_s and x_{target} through Eq. (13).

3.4 Prompt-Aware lightweight convolution

Lightweight convolution (Wu et al., 2019) employs a fixed context window and reuses the same weights for all context elements, regardless of the current time step. This property can be particularly advantageous for TTS, where relevant context elements tend to be more local compared to tasks such as machine translation or other language processing tasks, as discussed in (Elias et al., 2021b). Therefore, we adopt a lightweight convolution block composed of lightweight convolution, a gated linear unit (GLU) (Veness et al., 2021), and a feed-forward layer with residual connections as shown in Figure 5. To further improve the performance of zero-shot TTS, we extend the lightweight convolution with a speaker-adaptive mechanism inspired by SC-CNN (Yoon et al., 2023). Specifically, SC-CNN utilizes speaker embeddings to modulate the convolutional kernel weights dynamically. Following this approach, the speech prompt encoder receives a speech prompt—a 3-second mel-spectrogram segment extracted from the reference speech—and generates a speaker embedding s. This embedding s is then reshaped to serve as the kernel weights of the lightweight convolution. For the text encoder, which uses a 3×1 lightweight convolution with 8 heads, s is reshaped from a 256-dimensional vector to a 3×8 matrix. Similarly, when applied to the mel decoder, which employs a 17×1 lightweight convolution with 8 heads, s is reshaped to 17×8. Through this process, the lightweight convolution adapts its kernel weights dynamically based on the speech prompt, allowing it to extract local contexts that are tailored to each speaker. This adaptive mechanism contributes to improved zero-shot TTS performance.

3.5 Loss function

In this section, we describe the loss functions used to train RisoTTo. The loss $\mathcal{L}_{\text{SAG}}$, defined in Eq. (10), is used to train the SAG network to predict the vector field between the hard alignment A_{hard} and the soft alignment A_{soft}. The loss function $\mathcal{L}_{\text{dur}}$ for training the duration predictor is defined as the L2 distance between the target and predicted durations. Target is obtained from A_{log}. Thus, the total loss $\mathcal{L}_{total}$ function is described as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{\text{mel}} + \mathcal{L}_{\text{ref}} + \mathcal{L}_{\text{dur}} + \lambda\mathcal{L}_{\text{NF}} + \mathcal{L}_{\text{SAG}}, \quad (14)$$

where λ is hyper-parameter set to 10.

4 Experiments

We used LibriTTS-R (Koizumi et al., 2023) datasets for training RisoTTo. We selected 2,395 speakers from LibriTTS-R consisting of 580 hours of data from 2,456 speakers. Sampling rate was set to 22,050Hz, and mel-spectrogram was extracted with a hop size of 256 and a window size of 1024. Among residual speakers, 20 and 41 speakers became validation and test sets, respectively. In addition, we employ a NVIDIA RTX 3090 GPU with a batch size of 32, and Adam optimizer (Kingma and Ba, 2014) is used with a scheduled learning rate same as FastSpeech2 (Ren et al., 2020). Finally, we used *g2p-en*[2] to convert the text into phoneme sequence. Also, HiFi-GAN (Kong et al., 2020), which is trained with same train set of RisoTTo, was used as vocoder that converts mel-spectrogram into waveform.

[2] https://github.com/Kyubyong/g2p

Evaluation metrics: For fair evaluation, we employed pre-trained NISQA-MOS (Mittag et al., 2021) to evaluate naturalness of speech, instead of human evaluation. For objective evaluation, speaker embedding cosine similarity (SECS), representing speaker similarity, was computed using ECAPA-TDNN (Desplanques et al., 2020) pre-trained with Voxceleb (Nagrani et al., 2017). Also, we evaluated speech intelligibility with word error rate (WER), using a pre-trained speech recognition model from the official implementation of whisper (Radford et al., 2023) to measure transcription errors from generated samples. In the following experiments, the sampling numbers of flow matching in SAG and PostNet are 2 and 5, respectively.

4.1 Alignment modeling via SAG

We compared our soft alignment generation method with hard and Gaussian upsampling approaches to evaluate the effectiveness of the proposed method. The hard upsampling method, used in most NAR models such as (Ren et al., 2020; Han et al., 2024), increases the length of the text representation by simply duplicating each phoneme according to its duration. In contrast, the Gaussian upsampling (Donahue et al., 2020) performs differentiable upsampling, which can improve the speech likelihood of the TTS model. This is achieved by computing a weighted sum of the text representations to generate the context vector, optimized to minimize the mel-spectrogram loss ($\mathcal{L}_{\text{mel}}$). However, the range of the weighted sum is limited by the variance of the Gaussian distribution, which constrains the flexibility of the filter. Since our method used attention mechanism that conducts weighted sum for whole text representation, which provides more flexibility than Gaussian upsampling. In inference, flow matching, which observes optimized soft alignment about text representation, generates soft alignment without mel-spectrogram. Table 1 shows the results of RisoTTo using a different upsampling module instead of the SAG network. Attention mechanism in Table 1 denotes soft alignment produced from attention mechanism with target mel-spectrogram. For this evaluation, we randomly selected 6 unseen speakers for each upsampling method from the test set and generated 5 utterances per speaker. Gaussian upsampling outperformed hard upsampling, but Attention mechanism provides more delicate soft alignment than it. Thus, SAG network is trained using soft alignment of attention mechanism and shows better performance of Gaussian upsampling.

Upsampling	MOS(CI)	WER	SECS
Attention	4.24±0.09	4.83	0.694
Hard	3.85±0.05	5.11	0.649
Gaussian	4.07±0.11	5.37	0.672
SAG (ours)	4.19±0.10	5.03	0.681

Table 1: Zero-shot TTS performance of RisoTTo according to upsampling mtehods. “CI” represents 95% confidence intervals.

4.2 Residual modeling via invertible encoder

In this subsection, we demonstrate that the invertible encoder effectively captures the residual information between the mel-spectrogram and the context vector. This residual information should be disentangled from the context vector. To validate this, we employed Maximum Mean Discrepancy (MMD)(Gretton et al., 2012), a statistical distance metric that measures the difference between two probability distributions based on their sample means in a reproducing kernel Hilbert space. A lower MMD value indicates greater similarity between the distributions. Table 2 presents the MMD scores between the context vector c_s and the residual variable z, comparing the use of the invertible encoder and a variational autoencoder (VAE) for residual modeling. The results show that the z extracted by the invertible encoder exhibits lower statistical similarity with c_s than that of the VAE, indicating better disentanglement. We also computed the MMD between z and samples $\epsilon \sim N(0,1)$ drawn from the prior Gaussian distribution. The z from the invertible encoder is closer to the prior distribution than that from the VAE. In contrast, the VAE tends to produce a z that is too close to the prior, leading to posterior collapse—where z becomes uninformative. Therefore, z from the VAE should not be forced to align too closely with the prior distribution. This is a

Algorithm	(c_s, z)	(z, ϵ)
Invertible encoder	2.613	0.207
VAE	1.941	0.611

Table 2: The results of MMD score when using invertible encoder and VAE. (a,b) denotes MMD score between a and b. These score was computed using 50 utterances of validation set

critical distinction between the invertible encoder and the VAE. The invertible encoder extracts a deterministic latent representation z, whereas the VAE produces a distribution from which stochastic variable z are sampled. Because the invertible encoder generates deterministic representations, it is inherently free from posterior collapse. As a result, z can be more closely aligned with the prior distribution, which helps reduce the mismatch of z between training and inference phases in the TTS model.

We further demonstrate that incorporating an invertible encoder can enhance the performance of non-autoregressive TTS models. In Table 3, the autoregressive and non-autoregressive baselines refer to Transformer TTS (Li et al., 2019) and FastSpeech, which used linear layer-based feed forward layer, respectively. All models in Table 3 were trained on the LJSpeech-1.1 dataset (Ito and Johnson, 2017), which contains 13,100 utterances recorded by a single female speaker. We used 12,000 utterances for training, and 50 each for validation and testing. From each trained model, 15 utterances were generated and evaluated. As presented in Table 3, the use of an invertible encoder leads to a more substantial performance improvement in non-autoregressive TTS compared to the conventional VAE-based approach.

Model	MOS(CI)	WER
AR	3.73±0.11	6.93
NAR	3.38±0.13	7.05
NAR w/ IE	3.64±0.10	6.96
NAR w/ VAE	3.51±0.08	7.14

Table 3: The results on TTS models trained with LJSpeech-1.1 dataset.

4.3 Evaluation

Recently, many zero-shot TTS models were introduced but official implementation code is not released. Thus, we used audio samples obtained from official demo page. For the most fair evaluation possible we selected comparison models, which have audio samples generated by same dataset (not LibriTTS-R), among representative zero-shot TTS models. Thus, VALL-E, T5-TTS, and NaturalSpeech2 releases audio samples generated by VCTK (Christophe et al., 2017) that is not used to train RissoTTo and these comparison models. Audio samples of RisoTTo were 30 utterances generated by speech prompt of 5 unseen speakers of VCTK dataset. VALL-E (Wang et al., 2023) is one of the most well-known zero-shot TTS models, introducing a decoder-only architecture to model discrete tokens from a neural audio codec. T5-TTS (Neekhara et al., 2024), based on an encoder-decoder architecture, was also included as a representative autoregressive model. Lastly, NaturalSpeech2 (Shen et al., 2023) is non-autoregressive TTS using latent diffusion for generating latent representation of Encodec (Défossez et al., 2022). As shown in Table 4, we compared RisoTTo with the aforementioned models in terms of MOS, SECS, and WER. The results demonstrate that RisoTTo achieves superior performance compared to VALL-E and NaturalSpeech2, despite being a non-autoregressive model. Also, RisoTTo without PostNet, which means that RisoTTo generated mel-spectrogram using mel decoder, is better than NaturalSpeech. While T5-TTS outperforms RisoTTo in terms of MOS and WER, RisoTTo exhibits a higher SECS, indicating better perceived speaker similarity by prompt-aware lightweight convolution (PAL).

Model	MOS(CI)	SECS	WER
GT	4.62±0.04	0.840	3.88
VALL-E	3.92±0.10	0.541	6.42
T5-TTS	4.21±0.09	0.613	4.91
NaturalSpeech2	3.71±0.22	0.587	5.52
RisoTTo	4.18±0.11	0.629	5.31
RisoTTo w/o PostNet	3.94±0.10	0.553	5.84
RisoTTo w/o PAL	4.14±0.08	0.602	5.40

Table 4: The results on zero-shot TTS using VCTK dataset. GT denotes ground truth of waveform.

5 Conclusion

We proposed three algorithm that can improve performance of zero-shot non-autoregressive TTS. Soft alignment generation upsamples text representation to richer context vector, and invertible encoder effectively models residual information about acoustic representation. Then, prompt-aware lightweight convolution enhances speaker similarity via kernel weight depending on speech prompt. Thus, RisoTTo achieved better performance compared with representative zero-shot autoregressive TTS. Our future work considers improvement of latency and more diverse comparisons to evaluate performance.

Limitations

One limitation of this study is that the comparison models were not trained on the same dataset as ours. Instead, we relied on audio samples provided on the official demo pages of those models. Consequently, direct comparisons may not fully reflect performance under identical training conditions. Furthermore, the evaluation scope was limited, as it did not include a detailed investigation of inference latency. A more comprehensive comparison—incorporating a wider range of baseline models and controlled latency measurements—would further strengthen the findings.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, and 1 others. 2021. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *arXiv preprint arXiv:2110.07205*.

Eric Battenberg, RJ Skerry-Ryan, Daisy Stanton, Soroosh Mariooryad, Matt Shannon, Julian Salazar, and David Kao. 2024. Very attentive tacotron: Robust and unbounded length generalization in autoregressive transformer-based text-to-speech. *arXiv preprint arXiv:2410.22179*.

Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. 2022. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International conference on machine learning*, pages 2709–2720. PMLR.

Veaux Christophe, Yamagishi Junichi, and MacDonald Kirsten. 2017. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. https://datashare.ed.ac.uk/handle/10283/2651.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2022. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*.

Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. 2020. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. *arXiv preprint arXiv:2005.07143*.

Jeff Donahue, Sander Dieleman, Mikołaj Bińkowski, Erich Elsen, and Karen Simonyan. 2020. End-to-end adversarial text-to-speech. *arXiv preprint arXiv:2006.03575*.

Isaac Elias, Heiga Zen, Jonathan Shen, Yu Zhang, Ye Jia, RJ Skerry-Ryan, and Yonghui Wu. 2021a. Parallel tacotron 2: A non-autoregressive neural tts model with differentiable duration modeling. *arXiv preprint arXiv:2103.14574*.

Isaac Elias, Heiga Zen, Jonathan Shen, Yu Zhang, Ye Jia, Ron J Weiss, and Yonghui Wu. 2021b. Parallel tacotron: Non-autoregressive and controllable tts. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5709–5713. IEEE.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Bing Han, Long Zhou, Shujie Liu, Sanyuan Chen, Lingwei Meng, Yanming Qian, Yanqing Liu, Sheng Zhao, Jinyu Li, and Furu Wei. 2024. Vall-e r: Robust and efficient zero-shot text-to-speech synthesis via monotonic alignment. *arXiv preprint arXiv:2406.07855*.

Keith Ito and Linda Johnson. 2017. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/.

Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. 2020. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077.

Jaehyeon Kim, Keon Lee, Seungjun Chung, and Jaewoong Cho. 2024. Clam-tts: Improving neural codec language model for zero-shot text-to-speech. *arXiv preprint arXiv:2404.02781*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P. Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Proc. Advances in neural information processing systems* (NeurIPS).

Diederik P Kingma, Max Welling, and 1 others. 2013. Auto-encoding variational bayes.

Yuma Koizumi, Heiga Zen, Shigeki Karita, Yifan Ding, Kohei Yatabe, Nobuyuki Morioka, Michiel Bacchiani, Yu Zhang, Wei Han, and Ankur Bapna. 2023. Libritts-r: A restored multi-speaker text-to-speech corpus. *arXiv preprint arXiv:2305.18802*.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033.

Ji-Hyun Lee, Sang-Hoon Lee, Ji-Hoon Kim, and Seong-Whan Lee. 2022. Pvae-tts: Adaptive text-to-speech via progressive style adaptation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6312–6316. IEEE.

Keon Lee, Dong Won Kim, Jaehyeon Kim, and Jaewoong Cho. 2024. Ditto-tts: Efficient and scalable zero-shot text-to-speech with diffusion transformer. *arXiv preprint arXiv:2406.11427*.

Yoonhyung Lee, Joongbo Shin, and Kyomin Jung. 2020. Bidirectional variational inference for non-autoregressive text-to-speech. In *International conference on learning representations*.

Younggun Lee and Taesu Kim. 2019. Robust and fine-grained prosody control of end-to-end speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5911–5915. IEEE.

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. 2019. Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6706–6713.

Yinghao Aaron Li, Cong Han, and Nima Mesgarani. 2025. Styletts: A style-based generative model for natural and diverse text-to-speech synthesis. *IEEE Journal of Selected Topics in Signal Processing*.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2022. Flow matching for generative modeling. *International conference on learning representations*.

Shivam Mehta, Ruibo Tu, Jonas Beskow, Éva Székely, and Gustav Eje Henter. 2024. Matcha-tts: A fast tts architecture with conditional flow matching. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11341–11345. IEEE.

Gabriel Mittag, Babak Naderi, Assmaa Chehadi, and Sebastian Möller. 2021. Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. *arXiv preprint arXiv:2104.09494*.

Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. 2017. Voxceleb: a large-scale speaker identification dataset. *arXiv preprint arXiv:1706.08612*.

Paarth Neekhara, Shehzeen Hussain, Subhankar Ghosh, Jason Li, Rafael Valle, Rohan Badlani, and Boris Ginsburg. 2024. Improving robustness of llm-based speech synthesis by learning monotonic alignment. *arXiv preprint arXiv:2406.17957*.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. 2021. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International conference on machine learning*, pages 8599–8608. PMLR.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32.

Robin Rombach, Patrick Esser, and Bjorn Ommer. 2020. Network-to-network translation with conditional invertible neural networks. *Advances in Neural Information Processing Systems*, 33:2784–2797.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, and 1 others. 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE.

Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. 2023. Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. *arXiv preprint arXiv:2304.09116*.

Joel Veness, Tor Lattimore, David Budden, Avishkar Bhoopchand, Christopher Mattern, Agnieszka Grabska-Barwinska, Eren Sezener, Jianan Wang, Peter Toth, Simon Schmitt, and 1 others. 2021. Gated linear networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10015–10023.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, and 1 others. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, and 1 others. 2017. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Hyungchan Yoon, Changhwan Kim, Seyun Um, Hyun-Wook Yoon, and Hong-Goo Kang. 2023. Sc-cnn: Effective speaker conditioning method for zero-shot multi-speaker text-to-speech systems. *IEEE Signal Processing Letters*, 30:593–597.