MSMC-TTS: Multi-Stage Multi-Codebook VQ-VAE based Neural TTS

Haohan Guo, *Student Member, IEEE*, Fenglong Xie, *Member, IEEE*, Xixin Wu, *Member, IEEE*, Frank K. Soong, *Fellow, IEEE*, Helen Meng, *Fellow, IEEE*

***Abstract*—This paper aims to improve neural TTS with vector-quantized, compact speech representations. We propose a Vector-Quantized Variational AutoEncoder (VQ-VAE) based feature analyzer to encode acoustic features into sequences with different time resolutions, and quantize them with multiple VQ codebooks to form the Multi-Stage Multi-Codebook Representation (MSMCR). The TTS system, MSMC-TTS, is proposed to predict better speech via this representation. In prediction, the multi-stage predictor is trained to map the input text sequence to MSMCRs in stages, by minimizing Euclidean distance and "triplet loss". In synthesis, the neural vocoder converts ground-truth or predicted MSMCRs into speech waveforms. The proposed system is trained with single-speaker TTS datasets and tested in various scenarios for comprehensive evaluation. In TTS evaluation, MSMC-TTS obtains MOS of 4.34 and 4.10 on English and Chinese datasets, which significantly outperforms VITS with scores of 3.78 and 3.90. Meanwhile, compared with Mel-Spectrograms, the domain discrepancy between prediction and ground truth is lower in MSMCRs with the higher Domain-classification Error Rate (DER). Furthermore, this system shows lower modeling complexity and data size requirements, preserving excellent performance even with fewer model parameters or training data. The noticeable improvement in analysis-synthesis and TTS from multiple codebooks and stages also validate them as vital components in seeking a more profitable speech representation and building high-performance neural TTS.**

***Index Terms*—Multi-stage Multi-codebook (MSMC), Speech Representation, VQ-VAE, Neural TTS, Speech Synthesis**

I. Introduction

Text-to-speech synthesis (TTS) is a technology converting text to the corresponding speech audio. It is widely used in human-machine interaction [1], and promotes many cutting-edge technologies, such as Video Dubbing [2, 3], Dysarthric Speech Reconstruction [4], Speech Translation [5], etc. Hence, it is crucial to develop a high-quality TTS system to support various AI applications.

Neural TTS [6], constructed with advanced neural networks, has shown outstanding naturalness and fidelity in synthesized audios, spawning more related research works to build stronger TTS systems. The mainstream framework of neural TTS comprises three parts, i.e. analysis, synthesis, and prediction, as shown in Fig. 1. The system does not directly convert input text into waveform, but maps it to acoustic features via a neural network-based acoustic model in prediction. It then converts the features to the waveform in synthesis. In analysis, the raw waveform is compressed via signal processing into the acoustic feature sequence, such as pitch [7], line spectrum pair [8], Mel cepstra [9], etc., as speech representations. Compared with the waveform, these features can represent speech with shorter sequence lengths and fewer parameters, which are easier to predict from the text. Meanwhile, to reconstruct the waveform from these features well, the neural vocoder is trained to map acoustic features of speech training data to corresponding waveforms. It can generate high-fidelity waveforms using various advanced generative models, such as WaveNet [10], WaveRNN [11], WaveGlow [12], MelGAN [13], etc. In this way, the text can be successfully converted to speech waveforms by processing acoustic features predicted from the acoustic model.

Hanhan Guo and Helen Meng are with the Human-Computer Communications Laboratory (HCCL), Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong SAR, China (e-mail: hguo@se.cuhk.edu.hk; hmmeng@se.cuhk.edu.hk). Xixin Wu is with Stanley Ho Big Data Decision Analytics Research Centre, The Chinese University of Hong Kong, Hong Kong SAR, China (e-mail: xixinwu@cuhk.edu.hk). Fenglong Xie is with Xiaohongshu, Beijing, China (e-mail: fenglongxie@xiaohongshu.com). Frank K. Soong is with Microsoft, China (Retired, e-mail: fkpsoong@outlook.com)

This research was supported by the Center for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission's InnoHK scheme.

Corresponding author: Xixin Wu

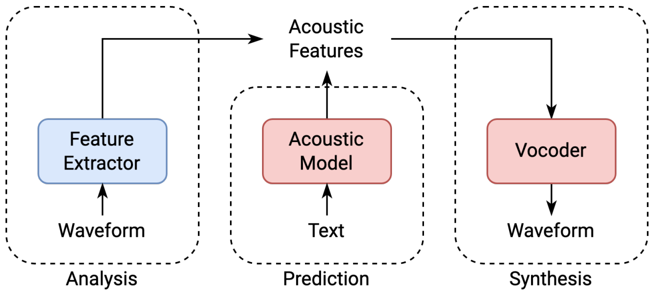

Fig. 1. The framework of the neural TTS system. The feature extractor is implemented with signal processing-based methods. The acoustic model and vocoder are implemented with neural networks.

However, the audio synthesized from predicted acoustic features cannot achieve the same fidelity as the audio generated by analysis-synthesis, i.e. synthesized from the ground-truth acoustic features. Due to the prediction error caused by the acoustic model, the predicted features have lower quality, showing worse fidelity and naturalness in the synthesized audio. More seriously, this error also puts the predicted features in a different distribution from ground-truth data [14, 15]. Since the neural vocoder is trained only with the ground-truth features and may not generalize well to out-of-domain data [16–19], the predicted features cannot be processed well, which further degrades the quality of the synthesized audio.

Recent work mainly concentrate on enhancing the acoustic model to mitigate this problem. First, the prediction error decreases by improving the modeling complexity with more complex sequential models, such as RNN [20, 21] and Self-Attention-based models [22, 23]. Secondly, the generative algorithms are also applied to model the target distribution directly, such as Auto-Regressive models [24, 25], Generative Adversarial Networks (GAN) [26, 27], Flow-based models [28, 29], etc. Moreover, the prediction error can also be mitigated in the neural vocoder by fine-tuning the vocoder on the predicted acoustic features aligned with the target waveforms [30, 31]. The End-to-End TTS [32–34], i.e. mapping text features to the waveform directly, is also being investigated to avoid the use of acoustic features. However, these model-oriented methods either put high requirements on modeling complexity, or need more computing resources for training and inference. Hence, this work proposes re-thinking the research problem from the perspective of speech feature representation to avoid those newly introduced burdens.

The reconstruction quality of analysis-synthesis determines the upper bound of TTS performance [35]. Hence, the speech representation is usually allocated with a higher dimension or larger space volume to preserve richer speech information. However, it also makes modeling harder, as the curse of dimensionality [36] indicates. A feature representation with an overly large space may make models over-fit or under-fit when used as inputs [37] or outputs [38]. It leads to increased prediction error in the acoustic model, and worse generalization of the neural vocoder. Hence, to ensure high-quality TTS performance, a good speech representation must consider these two dimensions:

- Completeness [39]: The target speech can be reconstructed well from this representation (also called self-consistency).
- Compactness: The target speech is represented by fewer parameters or a smaller space volume.

Although high completeness offers high fidelity of speech reconstructed from the representation, it is also critical to ensure high compactness to predict well for TTS.

The autoencoder, widely used in representation learning [40–42], can directly learn a latent representation from the target data with an encoder-decoder model. The input features are converted to the latent representations by the encoder, and reconstructed by the decoder. The better reconstruction quality indicates the higher completeness of this representation. Meanwhile, various constraints can be applied to control the sparsity and compactness of the latent representation, e.g. dimensionality reduction [43], feature sparsify autoencoders [44], and vector quantization [45]. Specifically, the discrete latent representation based on vector-quantized variational autoencoders (VQ-VAE) has shown outstanding compactness in speech codec [46, 47]. This inspires us to explore a VQ-VAE based compact speech representation for TTS.

This work first aims to train a feature analyzer to construct compact speech representations based on Mel spectrograms, a commonly used acoustic feature with sufficient completeness as the input of neural vocoders [48]. The trade-off between completeness and compactness is revealed by investigating VQ-VAE and its derived speech representation. Then, a multi-stage multi-codebook (MSMC) approach is proposed to compensate for the insufficient completeness in this high-compactness representation. Multiple codebooks are employed to quantize a vector to achieve a richer representation, and to quantize the speech sequence into multiple sub-sequences for equal preservation of speech information at different time resolutions. Then, a novel TTS system, MSMC-TTS, is proposed to generate speech audio from the text via the multi-stage multi-codebook representation (MSMCR). In this system, MSMCRs are extracted through the feature analyzer, and reconstructed back to audio by the neural vocoder. The multi-stage predictor is proposed as the acoustic model to predict MSMCRs progressively in resolution from the input text, and trained with the proposed loss function combining MSE and a “triplet loss”.

Experiments are conducted from two aspects – TTS evaluation and analysis-synthesis evaluation. In TTS evaluation, aside from the MOS test used to evaluate different methods subjectively, domain discrepancy is also measured directly via Domain-classification Error Rate (DER) to evaluate the difference between predicted and target features. In standard single-speaker TTS, MSMC-TTS is compared with the Mel-spectrogram based TTS system, FastSpeech, and the SOTA end-to-end TTS system, VITS [34], on both English and Chinese datasets. We also investigate the impact of multi-codebook VQ and multi-stage modeling on TTS synthesis. Moreover, MSMC-TTS is also evaluated in resource-limited scenarios to investigate the impact of modeling complexity and data size on the TTS quality. In analysis-synthesis evaluation, we compare different representations by measuring the reconstruction quality of speech.

In the rest of this paper, Section II introduces the VQ-VAE based compact speech representation. Sections III and IV illustrate the proposed multi-stage multi-codebook VQ-VAE and the corresponding MSMC-TTS system, respectively. Then, experiments are described in Sections V - VII. Finally, Section VIII gives the conclusion to this paper.

II. Vector Quantization-based Compact Speech Representation

In this section, we will introduce Vector-Quantized Variational AutoEncoder (VQ-VAE), and its derived Vector-Quantized Representation (VQR).

A. Vector Quantized Variational AutoEncoder

VQ-VAE aims to learn a discrete latent representation from target data with an encoder-decoder model. As shown in Fig. 2, VQ-VAE comprises three parts, encoder, decoder, and the VQ operation in between. The input speech sequence $\mathbf{x} = \{x_1, x_2, ..., x_L\}$ is first encoded by the encoder $\mathbf{E}$:

$$\tilde{\mathbf{z}} = \mathbf{E}(\mathbf{x}) \tag{1}$$

Then, the vector-quantized latent representations $\mathbf{z}$ are obtained as follows:

$$\mathbf{z} = Q(\tilde{\mathbf{z}}; \mathbf{c}) \tag{2}$$

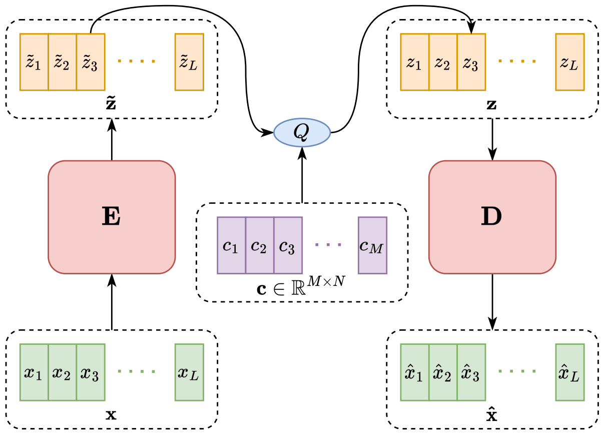
$\tilde{z}_1$ $\tilde{z}_2$ $\tilde{z}_3$ $\tilde{z}_L$
$\tilde{\mathbf{z}}$
Q
z_1 z_2 z_3 z_L
$\mathbf{z}$
E
D
c_1 c_2 c_3 c_M
$\mathbf{c} \in \mathbb{R}^{M\times N}$
x_1 x_2 x_3 x_L
$\mathbf{x}$
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_L$
$\hat{\mathbf{x}}$

Fig. 2. The framework of VQ-VAE for speech data. $\mathbf{E}$ and $\mathbf{D}$ denote the encoder and decoder, respectively. $\mathbf{x}$ and $\hat{\mathbf{x}}$ refer to the input and output speech sequence with the length of L. $\tilde{\mathbf{z}}$ and $\mathbf{z}$ refer to latent sequences before or after vector quantization "Q" with the codebook $\mathbf{c}$ composed of M codewords with the dimension of N.

where $\mathbf{c}$ denotes the codebook containing M codewords with the dimension of N. For each latent vector $\tilde{z}_i$, the quantizer Q compares it with all codewords, and chooses the one nearest to it according to the Euclidean distance as the quantized output z_i, which is written as follows:

$$z_i = c_{j^*} \quad where \quad j^* = \underset{1\leq j\leq M}{\arg\min} \|\tilde{z}_i - c_j\|_2^2 \tag{3}$$

Finally, the output speech sequence $\hat{\mathbf{x}}$ is generated by the decoder $\mathbf{D}$ for reconstruction:

$$\hat{\mathbf{x}} = \mathbf{D}(\mathbf{z}) \tag{4}$$

In training, VQ-VAE learns to reconstruct speech by minimizing the Euclidean distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$. However, due to the non-differentiable VQ operation, the gradient from the decoder cannot be back-propagated to the encoder directly. Hence, another loss term is introduced to train the encoder by minimizing the Euclidean distance between $\mathbf{z}$ and $\tilde{\mathbf{z}}$. The complete loss function is written as follows:

$$\begin{aligned} \mathcal{L}_{vqvae} &= \mathcal{D}_e(\hat{\mathbf{x}}, \mathbf{x}) + \alpha * \mathcal{D}_e(\tilde{\mathbf{z}}, sg(\mathbf{z})) \\ &= \frac{1}{L}\sum_{i=1}^{L} ||\hat{x}_i - x_i||_2^2 + \frac{\alpha}{L}\sum_{i=1}^{L} ||\tilde{z}_i - sg(z_i)||_2^2 \end{aligned} \tag{5}$$

where $\mathcal{D}_e(*,*)$ denotes Euclidean distance between two objects, $sg(*)$ denotes the operation stopping its gradient back-propagating, and α is a coefficient to balance these two loss terms. Meanwhile, codewords in the codebook are updated using the exponential moving average-based method [49] as follows:

$$\begin{aligned} \hat{u}_j^{(t)} &= \beta \hat{u}_j^{(t-1)} + (1-\beta)u_j \\ \hat{v}_j^{(t)} &= \beta \hat{v}_j^{(t-1)} + (1-\beta)v_j \\ c_j &= \frac{\hat{v}_j^{(t)}}{\hat{u}_j^{(t)}} \end{aligned} \tag{6}$$

where u_j and v_j denote the number and sum of hidden vectors $\{\tilde{z}_i\}$ quantized to c_j in the mini-batch at t-th training iteration, β is a coefficient between 0 and 1, and set to 0.99 as default.

B. Compactness of Vector-Quantized Representation

In this framework, an acoustic feature sequence can be represented by an index sequence. When a vector composed of N numbers (B bits per number) is compressed to one integer with the range from 1 to M representing the number of codewords, the compression ratio $\mathcal{R}$ can be calculated as follows:

$$\mathcal{R} = \frac{N * B}{\log_2(M)} \tag{7}$$

In this way, the 80-dim Mel spectrogram (32-bit float-point number) can be compressed $2560/log_2(M)$ times. The higher compression ratio brings higher feature compactness, but also reduces feature completeness due to increased information loss, making it insufficient for high-quality speech reconstruction and prediction. To compensate for this loss, we further optimize VQR, and propose a multi-stage multi-codebook approach.

III. Multi-Stage Multi-Codebook VQ-VAE

The MSMC-VQ-VAE based feature analyzer aims to encode a speech sequence stage-wise into the Multi-Stage Multi-Codebook Representation (MSMCR), i.e. a set of sub-sequences at different time resolutions, and quantized by multiple codebooks respectively. This approach comprises two parts: multi-head vector quantization and multi-stage autoencoder.

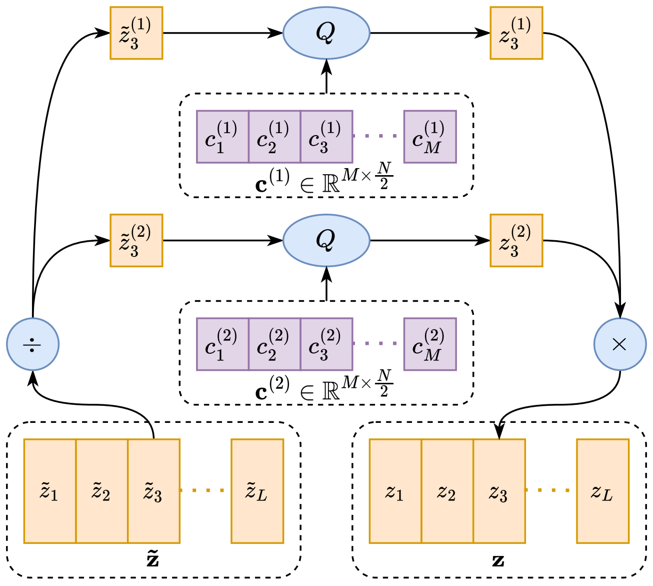
$\tilde{z}_3^{(1)}$
Q
$z_3^{(1)}$
$c_1^{(1)}$ $c_2^{(1)}$ $c_3^{(1)}$ $c_M^{(1)}$
$\mathbf{c}^{(1)} \in \mathbb{R}^{M\times \frac{N}{2}}$
$\tilde{z}_3^{(2)}$
Q
$z_3^{(2)}$
$c_1^{(2)}$ $c_2^{(2)}$ $c_3^{(2)}$ $c_M^{(2)}$
$\mathbf{c}^{(2)} \in \mathbb{R}^{M\times \frac{N}{2}}$
$\div$
$\times$
$\tilde{z}_1$ $\tilde{z}_2$ $\tilde{z}_3$ $\tilde{z}_L$
$\tilde{\mathbf{z}}$
z_1 z_2 z_3 z_L
$\mathbf{z}$

Fig. 3. An example of Multi-Head Vector Quantization with $H = 2$, where $\div$ and $\times$ denote chunking and concatenation operation. The codebook $\mathbf{c}$ is divided into two sub-codebooks $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$. The input vector is also chunked into two sub-vectors, and quantized by corresponding sub-codebooks, finally concatenated together to form the output vector.

A. Multi-Head Vector Quantization

To obtain the balance between compactness and completeness of VQR, reducing the compression ratio may give a richer representation, but is difficult in VQ-VAE. For example, to half the compression ratio, we need to square the codebook size due to the $log_2(*)$ operation in Eq. 7, which exponentially increases data and computation requirements for sufficiently training each codeword. To avoid such a dilemma, we employ product quantization [50], also called Multi-Head Vector Quantization (MHVQ) here, inspired by the multi-head attention mechanism [51]. In this approach, one VQ codebook $\mathbf{c} \in \mathbb{R}^{M \times N}$ is chunked evenly into H sub-codebooks $\{\mathbf{c}^{(i)} \in \mathbb{R}^{M \times \frac{N}{H}} \ for \ i = 1, 2, ..., H\}$. The input vector $x \in \mathbb{R}^N$ is quantized as follows:

$$\begin{aligned} x^{(1)}, ..., x^{(H)} &= Chunk(x, H) \\ y^{(i)} &= Q(x^{(i)}; \mathbf{c}^{(i)}) \ for \ i = 1, ..., H \\ y &= Concat(y^{(1)}, ..., y^{(H)}) \end{aligned} \tag{8}$$

x is firstly chunked into H sub-vectors with the dimensionality of $\frac{N}{H}$, and then quantized by corresponding codebooks, respectively. Finally, these quantized sub-vectors are concatenated together as the output vector y. In this approach, the compression ratio is calculated by:

$$\mathcal{R} = \frac{1}{H} * \frac{N * B}{\log_2(M)} \tag{9}$$

In this way, without introducing more parameters, the compression ratio can be halved by doubling the number of codebooks, avoiding exponentially increasing requirements caused by $log_2(*)$, and reducing the compression ratio more effectively.

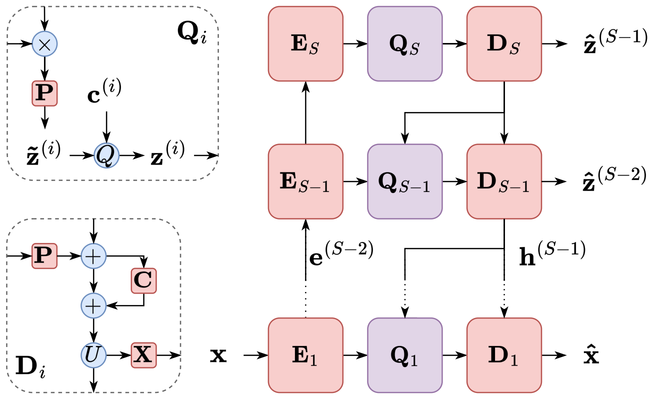

Fig. 4. The architecture of the multi-stage multi-codebook VQ-VAE. $\mathbf{E}_S$, $\mathbf{Q}_S$, $\mathbf{D}_S$ denote the S-th encoder, quantizer, and decoder. $\mathbf{Q}_i$ and $\mathbf{D}_i$ block show the detailed structures of the quantizer and decoder. $\mathbf{P}$ and $\mathbf{C}$ refer to a projection and convolutional layer. $\mathbf{X}$ is an arbitrary neural network based module. U, Q, "$\times$", and "$+$" denote operations of up-sampling, quantization, concatenation, and addition.

B. Multi-Stage AutoEncoder

In addition to improving completeness, it is crucial to well-represent all speech attributes at different time resolutions, such as suprasegmental prosody and segmental pronunciations [52, 53]. Otherwise, any overly compressed attribute may lead to severe degradation in overall quality, as the "Cannikin Law" [54] reveals. In this work, we propose the multi-stage autoencoder to model the speech sequence at different time resolutions [49].

As shown in Fig. 4, this model aims to encode the input sequence $\mathbf{x}$ into the multi-stage vector-quantized representation $Z = \{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(S)}\}$ using the codebook group $C = \{\mathbf{c}^{(1)}, ..., \mathbf{c}^{(S)}\}$, where S denotes the number of stages. Notably, the codebook $\mathbf{c}^{(i)}$ will also denote the group of sub-codebooks $\{\mathbf{c}^{(i,1)}, ..., \mathbf{c}^{(i,H)}\}$ if MHVQ is applied. The model first encodes the input speech sequence $\mathbf{x}$ to encoding sequences $\{\mathbf{e}^{(1)}, ..., \mathbf{e}^{(S)}\}$ with encoders progressively:

$$\mathbf{e}^{(i)} = \begin{cases} \mathbf{E}_i(\mathbf{x}) & \text{if } i = 1 \\ \mathbf{E}_i(\mathbf{e}^{(i-1)}) & \text{if } i > 1 \end{cases} \tag{10}$$

Before each encoder block, a strided convolutional layer down-samples the sequence to a lower time resolution. Subsequently, sequences are quantized in stages from the highest stage with the lowest resolution:

$$\mathbf{z}^{(i)} = \begin{cases} \mathbf{Q}_i(\mathbf{e}^{(i)}, \phi, \mathbf{c}^{(i)}) & \text{if } i = S \\ \mathbf{Q}_i(\mathbf{e}^{(i)}, \mathbf{h}^{(i+1)}, \mathbf{c}^{(i)}) & \text{if } i < S \end{cases} \tag{11}$$

where ϕ means no input, $\mathbf{c}^{(i)}$ refers to the corresponding codebook, and $\mathbf{c}^{(i)} = \{\mathbf{c}^{(i,1)}, ..., \mathbf{c}^{(i,H)}\}$ when MHVQ is applied. Specifically, in the quantizer block $\mathbf{Q}_i$, $\mathbf{e}^{(i)}$ is concatenated with the hidden sequence $\mathbf{h}^{(i+1)}$ (except when $i = S$) from the higher-stage decoder, and then transformed by a projection layer to obtain $\tilde{\mathbf{z}}^{(i)}$ for the following quantization.

The quantized sequence $\mathbf{z}^{(i)}$ is processed by the decoder $\mathbf{D}_i$ to obtain $\mathbf{h}^{(i)}$ for the following quantization and decoding, and to predict the next-stage quantized sequence $\mathbf{z}^{(i-1)}$ or speech sequence $\mathbf{x}$:

$$\begin{cases} \mathbf{h}^{(i)}, \hat{\mathbf{z}}^{(i-1)} = \mathbf{D}_i(\mathbf{z}^{(i)}, \phi) & \text{if } i = S \\ \mathbf{h}^{(i)}, \hat{\mathbf{z}}^{(i-1)} = \mathbf{D}_i(\mathbf{z}^{(i)}, \mathbf{h}^{(i+1)}) & \text{if } 2 \leq i \leq S-1 \\ \mathbf{x} = \mathbf{D}_i(\mathbf{z}^{(i)}, \mathbf{h}^{(i+1)}) & \text{if } i = 1 \end{cases} \tag{12}$$

The decoder $\mathbf{D}_i$ first transforms the quantized sequence $\mathbf{z}^{(i)}$ with a projection, and adds it with $\mathbf{h}^{(i+1)}$ when $i < S$. A residual convolutional layer further processes it, which is then up-sampled by repetition to $\mathbf{h}^{(i)}$. The output sequence is also processed by another neural network based module $\mathbf{X}$ for prediction.

The loss function of this model is written as follows:

$$\begin{aligned} \mathcal{L}_{msmc} = \ & \mathcal{D}_e(\mathbf{x}, \hat{\mathbf{x}}) \\ & + \alpha * \frac{1}{S} \sum_{j=1}^{S} \mathcal{D}_e(\tilde{\mathbf{z}}^{(j)}, sg(\mathbf{z}^{(j)})) \\ & + \beta * \frac{1}{S-1} \sum_{j=1}^{S-1} \mathcal{D}_e(\hat{\mathbf{z}}^{(j)}, sg(\mathbf{z}^{(j)})) \end{aligned} \tag{13}$$

where α and β denote weight coefficients. The first two terms are similar to $\mathcal{L}_{vqvae}$, but the second term calculates $\mathcal{D}_e$ for all latent sequences. The third term makes the predicted latent sequences approach target ones. Due to the introduction of multi-stage modeling, this deep network is vulnerable to over-fitting to lower-stage modules, making higher-stage latent sequences meaningless. Hence, deeper modules are trained to predict the low-stage representation from the high-stage sequence, which is validated as a practical approach in Very Deep VAEs [55].

IV. MSMC-TTS

This section will introduce the MSMC-TTS system, including the system framework and the proposed multi-stage predictor.

A. System Framework

Fig. 5 shows the framework of the proposed system, composed of three modules, analysis, prediction, and synthesis. In analysis, a speech signal $\mathbf{s}$ is converted to MSMCR Z. Since signal processing-based acoustic features have shown promising completeness in neural vocoders, the feature analyzer MSMC-VQ-VAE does not directly model $\mathbf{s}$, but acoustic features $\mathbf{x}$, e.g. Mel-Spectrograms used in our experiments. This model is first trained to minimize the loss function $\mathcal{L}_{msmc}$, and then provides MSMCR Z and codebook group C for synthesis and prediction. In synthesis, the neural vocoder aims to convert MSMCR to the corresponding speech waveform. Since sequences in the MSMCR have different time resolutions, they are first up-sampled to the same resolution by repetition, and then concatenated to form the input sequence. This model is trained with ground-truth data extracted from the training set, and then serves the prediction module to generate the audio from the predicted MSMCR P.

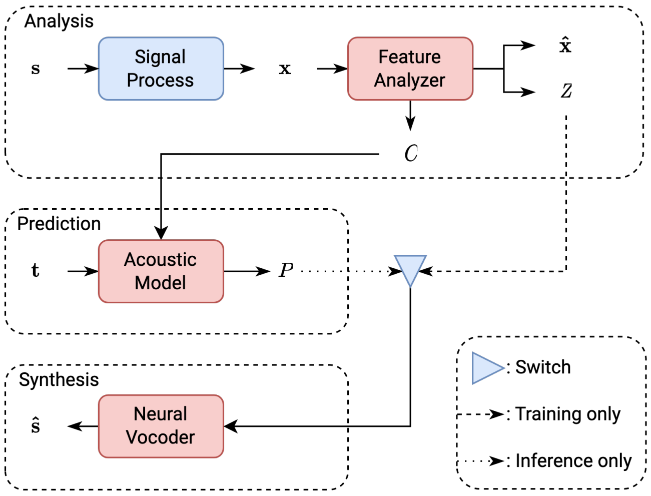

Fig. 5. The framework of MSMC-TTS. In analysis, acoustic features $\mathbf{x}$ are extracted from the speech signal $\mathbf{s}$ using signal processing-based methods. The proposed feature analyzer processes them and provides the reconstructed features $\hat{\mathbf{x}}$, MSMCR Z, and codebook group C for the following operations. In prediction, the acoustic model utilizes C to predict the MSMCR P from the text sequence $\mathbf{t}$. In synthesis, the neural vocoder generates speech signals $\hat{\mathbf{s}}$ from Z or P according to the running mode, training, or inference.

In prediction, the acoustic model aims to convert the textual sequence $\mathbf{t}$ to the corresponding MSMCR P of multiple sequences. It cannot be achieved directly using current mainstream acoustic models designed for one-to-one mapping. Hence, there is a dire need for a one-to-many acoustic model. Besides, since latent sequences in MSMCR are extracted in order of higher stage to lower stage, the generation process should also consider this pattern to maintain the correlation between sequences on the timescale. Consequently, we propose the Multi-Stage Predictor (MSP) according to these requirements as the acoustic model to predict these sequences in stages from the text.

B. Multi-Stage Predictor

The MSP is proposed based on FastSpeech [22], a Non-AutoRegressive (NAR) acoustic model with explicit duration modeling. We will introduce the model architecture, and the corresponding loss function for training.

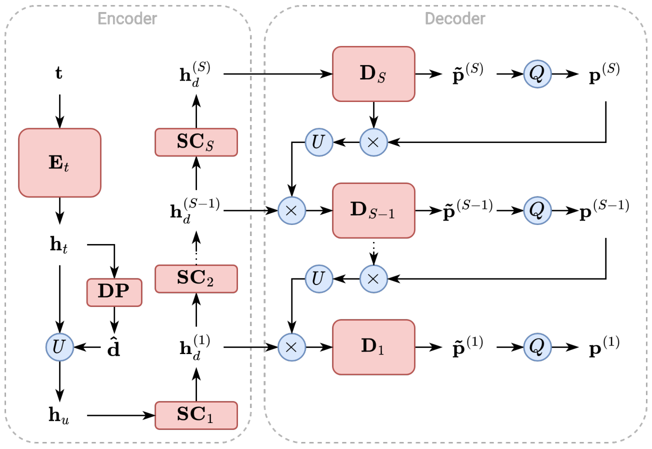

Fig. 6. The framework of Multi-Stage Predictor. $\mathbf{t}$, $\hat{\mathbf{d}}$, $\mathbf{h}_t$, $\mathbf{h}_u$, $\mathbf{h}_d^{(S)}$ denote the text sequence, predicted durations, encoder output, up-sampled encoder output, S-th down-sampled encoder output. $\tilde{\mathbf{p}}^{(S)}$ and $\mathbf{p}^{(S)}$ are S-th predicted sequences before or after quantization. $\mathbf{E}_t$, $\mathbf{DP}$, $\mathbf{D}_S$ denote the text encoder, duration predictor, and S-th decoder. $\mathbf{SC}$ refers to a 1-D strided convolutional layer for down-sampling operation.

1) Model Architecture: As shown in Fig. 6, MSP is composed of two parts, encoder, and decoder. The text sequence $\mathbf{t}$, i.e. phoneme sequence in our work, is first encoded to hidden vectors $\mathbf{h}_t = \mathbf{E}_t(\mathbf{t})$, and then up-sampled by repetition according to duration sequence $\hat{\mathbf{d}} = \mathbf{DP}(\mathbf{h}_t)$. Notably, $\hat{\mathbf{d}}$ is a non-negative integer sequence, where each integer means the number how many frames its corresponding phoneme lasts in $\mathbf{x}$. $\mathbf{h}_u$ is down-sampled in stages by strided convolutional layers to the corresponding stages $\{\mathbf{h}_d^{(1)}, ..., \mathbf{h}_d^{(S)}\}$.

The decoder predicts P from the highest stage. First, $\mathbf{h}_d^{(S)}$ is fed to the decoder D_S to obtain the predicted sequence $\tilde{\mathbf{p}}^{(S)}$, which is then quantized by the corresponding codebook $\mathbf{c}^S$ to $\mathbf{p}^{(S)}$. The last hidden output sequence in the decoder and $\mathbf{p}^{(S)}$ are up-sampled by repetition to concatenate with $\mathbf{h}_d^{(S-1)}$ as the input of the next decoder $\mathbf{D}_{S-1}$. The remaining sequences are generated recursively in the same way to form the predicted MSMCR P.

In training, the ground truth MSMCR Z and ground-truth durations $\mathbf{d}$ are available in advance. Hence, we adopt teacher-forcing to train MSP, i.e. replacing $\hat{\mathbf{d}}$ with $\mathbf{d}$, and inputting $\{\mathbf{z}^{(S)}, ..., \mathbf{z}^{(2)}\}$ to decoders.

2) Loss Function: To map the input text to the expected codewords, the model is trained to estimate them directly in continuous space. Like VQ-VAE, we first purpose to approach targets by minimizing the Euclidean distance:

$$\mathcal{L}_e = \frac{1}{S}\sum_{i=1}^{S}\mathcal{D}_e(\mathbf{p}^{(i)}, \mathbf{z}^{(i)}) \quad (14)$$

We also adopt the "triplet loss" [56], an effective ranking loss used in metric learning, to make the Euclidean distance between the input vector x and target codeword $t \in \mathbf{c}$ smaller than any other codewords $\mathbf{c}/t = \{w | w \in \mathbf{c}, w \neq t\}$. It ensures that the predicted vector can be quantized to the expected codeword. Combining with MHVQ, the "triplet loss" is written as:

$$\mathcal{D}_t(x, t, \mathbf{c}) = \frac{1}{M} \sum_{w \in \mathbf{c}/t} max(0, \|x - t\|_2^2 - \|x - w\|_2^2 + \epsilon)$$
$$\mathcal{L}_t = \frac{1}{S} \sum_{j=1}^{S} \frac{1}{H} \sum_{k=1}^{H} \frac{1}{L_j} \sum_{i=1}^{L_j} \mathcal{D}_t(p_i^{(j,k)}, z_i^{(j,k)}, \mathbf{c}^{(j,k)}) \tag{15}$$

where ϵ is a constant value called the margin number. In this way, the output vector will not only be closer to the target, but will also lie farther away from non-target codewords. Finally, the loss function for MSP is written as:

$$\mathcal{L}_{msp} = \mathcal{L}_e + \gamma * \mathcal{L}_t \tag{16}$$

where γ is a weight coefficient.

V. Experimental Protocol

Our experiments are conducted to evaluate MSMC-TTS from two aspects – analysis-synthesis and TTS. In this section, we will introduce the experimental configurations, including the datasets and the implementation details of the TTS system.

A. Datasets

The experiments are mainly conducted based on the English TTS corpus, Nancy. Besides, a Chinese TTS corpus is also used to validate the effectiveness of the proposed method in various languages. The details of these two datasets are shown below.

1) Nancy [57]: Nancy is a standard English single-speaker TTS dataset used for Blizzard Challenge 2011. It includes 16-hour clean recordings of a female voice with corresponding transcripts. There are 11,265 utterances used to train the English TTS system, and 95 utterances out of the training set used for testing. The phoneme sequence with punctuation is extracted from the text as the input sequence.[1] The phoneme-level durations are extracted from a pre-trained Tacotron2 model with stepwise monotonic attention [58] using the same method described in FastSpeech [22].

2) CSMSC: CSMSC is a standard open-source Mandarin single-speaker TTS dataset.[2] It includes around 12 hours of recordings of a female voice with the corresponding transcripts, phonemes, and durations, which can be directly used for Chinese TTS system training.

[1]The G2P tool is available at https://github.com/Kyubyong/g2p

[2]The data is available at https://www.data-baker.com/data/index/source

B. Implement Details

1) Signal Analysis: All audios are down-sampled to 16kHz sampling rate, pre-emphasized with the coefficient of 0.97, and then converted to 1025-dim magnitude spectrograms by STFT with a window length of 50ms, the frameshift of 12.5ms, and the FFT size of 2048. Finally, the 80-dim log-scale Mel spectrograms are extracted and normalized to the range $[-4, 4]$ by min-max normalization.

2) Feature Analyzer: MSMC-VQ-VAE is implemented based on Feed-Forward Transformer in FastSpeech. In each encoder, the input sequence is processed by a projection layer, and added with position encodings, then fed to 4 FeedForward Transformer blocks. Specifically, the number of heads in multi-head attention is 2, and the feedforward module is composed of two convolutional layers with a kernel size of 3 and a ReLU activation function between them. The $\mathbf{X}$ module in the decoder $\mathbf{D}_1$ predicting $\mathbf{x}$ is also implemented in this way. In other decoders, $\mathbf{X}$ is implemented with a projection layer. The dimension of the models and codebooks are 256.

Each model is trained for 200,000 iterations with the Adam optimizer [59] ($\beta_1 = 0.9, \beta_2 = 0.98$) and the batch size of 64. The learning rate is exponentially decayed using the function:

$$lr = \begin{cases} max(lr_{final}, 0.5^{\frac{i-w}{\lambda}}) & \text{if } i > w \\ lr_{init} & \text{if } i <= w \end{cases} \tag{17}$$

where the initial learning rate lr_{init}, final learning rate lr_{final}, decay rate λ, warmup iterations w are set to 2×10^{-4}, 10^{-6}, 20000, 20000, respectively.

3) Neural Vocoder: We adopt HifiGAN [60] as the neural vocoder to up-sample speech features to the corresponding audio with the sample rate of 16kHz. The hyperparameters of the generator are based on HifiGAN-V1, and we set upsample scales and kernel sizes to $[5, 5, 4, 2]$ and $[11, 11, 8, 4]$. Moreover, time-domain and frequency-domain discrimination are used to train the model for higher fidelity. The discriminator proposed in UnivNet [61] is employed in our work. In the multi-resolution spectrogram discriminator, the magnitude spectrograms extracted with three STFT parameter sets, FFT size $[256, 512, 1024]$, frameshift $[40, 80, 160]$, frame length $[120, 320, 640]$, are used here. The multi-period waveform discriminator has the same configuration as UnivNet-c32.[3]

The vocoder is trained for 400,000 iterations by the AdamW [62] optimizer used in HifiGAN. The learning rate is decayed using the same method mentioned in Sec.V-B2, with $lr_{init} = 2 \times 10^{-4}$, $lr_{final} = 10^{-5}$, $\lambda = 200000$, $w = 200000$. In each training iteration, 16 audio segments with a length of 1 second are randomly selected from 16 audio files as one mini-batch. In the loss function, the weight coefficients of Mel-spectrogram loss and feature matching loss are set to 45 and 2, the same as in the official implementation.

4) Acoustic Model: The acoustic model is also implemented based on FastSpeech,[4] which uses FeedForward Transformer as the text encoder and decoders. The model dimension

[3]The implementations of HifiGAN generator and UnivNet discriminator are available at https://github.com/jik876/hifi-gan and https://github.com/mindslab-ai/univnet.

[4]The implementation of FastSpeech is available at https://github.com/NVIDIA/NeMo

and the number of blocks are set to 600 and 6 for better modeling capability. The text encoder first uses the embedding layer to convert the discrete phoneme sequence into continuous representations. Models are trained with the same training configurations as the feature analyzer for 100,000 iterations. Notably, the duration predictor predicts the linear-scale phoneme-level durations. An MSE loss between predicted durations and ground-truth ones with a weight coefficient of 0.1 is calculated to update the model.

Meanwhile, FastSpeech, as the baseline TTS system in our experiments, is also trained with the same model hyperparameters and training configurations. It has a similar model structure as MSMC-TTS, but uses Mel spectrograms as the target of the acoustic model, and then converts them to the waveform via the HifiGAN vocoder. We also compare MSMC-TTS with the SOTA end-to-end TTS system, VITS [34]. In this system, a VAE model encodes the speech via a CNN-based encoder and reconstructs the waveform by the HifiGAN-based decoder. Then, the acoustic model based on Glow-TTS [63] predicts the latent representations for TTS synthesis. All modules in the end-to-end system are jointly trained for 1 million iterations on 4 GPUs with a batch size of 64 utterances.

VI. TTS Evaluation

This section will present the evaluation results of the proposed approach in TTS, including evaluation metrics, and the performance in standard single-speaker TTS and resource-limited scenarios. The two-stage MSMCR with down-sample rates [1, 4], four heads (codebooks) per stage, 512 codewords per head, is used in MSMC-TTS as default.

A. Evaluation Metrics

1) Subjective Evaluation: The goal of TTS is to generate a speech that listeners find satisfactory. Hence, we conduct subjective tests to compare the performance of different TTS models directly. The Mean Opinion Score (MOS), a commonly used measurement, is adopted in our experiments. As for a set of samples corresponding to the exact input text, the listener needs to rate each sample according to its intelligibility, naturalness, and fidelity. The score ranges from 1 to 5 with an increment of 0.5, where 1 means very poor and 5 means excellent. Finally, all scores to the same method are averaged as the final score.[5]

2) Domain Discrepancy: As mentioned in Section I, the TTS quality is affected by the discrepancies between ground-truth features and the predicted ones. In this work, we measure the discrepancy directly to evaluate the proposed method. Two domains can be distinguished using a classification model by supervised learning if given sufficient data from these two domains. The more significant the discrepancy, the higher the classification accuracy. Conversely, a failed classification indicates that two domains may share the same distribution. Here, a 3-layer MLP model with a hidden size of 100 and the ReLU activation function is trained as the classifier to distinguish if the input vector is ground-truth or predicted by TTS. Finally, the Domain-classification Error Rate (DER) on the train and test sets are presented to show the discrepancy. The training and test sets in English experiments have 80 and 20 utterances. And Chinese experiments use 160 and 40 utterances to form the training and test sets. All of these utterances are not used in TTS training.

[5]Samples are available at https://hhguo.github.io/DemoMSMCTTSJournal

B. Standard TTS

To evaluate the proposed approach in different languages, this experiment implements the standard single-speaker TTS system using two standard TTS datasets, Nancy and CSMSC.

TABLE I
TTS Evaluation: Standard Single-Speaker TTS

Language	Name	MOS (95% CI)	DER (%)	
			Train Set	Test Set
English	Recording	4.46 ± 0.09	-	-
	Mel-FS	3.38 ± 0.11	1.14	5.09
	Mel-FS-FT	3.42 ± 0.12	-	-
	VITS	3.78 ± 0.13	-	-
	MSMC-TTS	4.34 ± 0.09	26.81	30.47
Chinese	Recording	4.57 ± 0.10	-	-
	Mel-FS	3.51 ± 0.11	1.91	5.53
	VITS	3.90 ± 0.12	-	-
	MSMC-TTS	4.10 ± 0.10	31.66	34.65

1) System Comparison: Table I shows the MOS test results. In English TTS, Mel-FS obtains an MOS of 3.38, which has an apparent gap compared with the original recordings with an MOS of 4.46. Its DER, 1.14%, and 5.09%, also show that the predicted features are easily distinguishable from the real ones, indicating the significant discrepancy between them. Mel-FS-FT refers to Mel-FS with the vocoder fine-tuned for 100,000 more iterations on the synthetic Mel spectrograms, generated from the training set using ground-truth durations. This method adapts the vocoder to the predicted features to compensate for this discrepancy. Nevertheless, it only slightly improves audio quality to the MOS of 3.42, which shows the limitation of this approach. Compared with the two previous systems, MSMC-TTS shows significantly improved performance with an MOS of 4.41, which is very close to the quality of the original recordings. The DER, 26.81%, and 30.47%, are also much higher than Mel-FS, showing a much more minor domain discrepancy. The SOTA end-to-end TTS approach, VITS, based on VAE representations, performs better than FastSpeech, achieving an MOS of 3.78. But it also needs more computing resources for end-to-end training. In this regard, MSMC-TTS requires fewer computing resources while showing better TTS performance than VITS.

We also obtain the same conclusion in Chinese TTS. The result shows that VITS and MSMC-TTS perform better than FastSpeech. And VITS is enhanced by applying ground-truth durations in training. Nonetheless, MSMC-TTS still shows the best performance with an MOS of 4.10. It still keeps low domain discrepancy with the DER of 31.66% and 34.65% in the train and test sets.

Moreover, to further compare Mel-FS and MSMC-TTS, which have a similar model structure but mainly differ in speech representations, we visualize the audio generated by

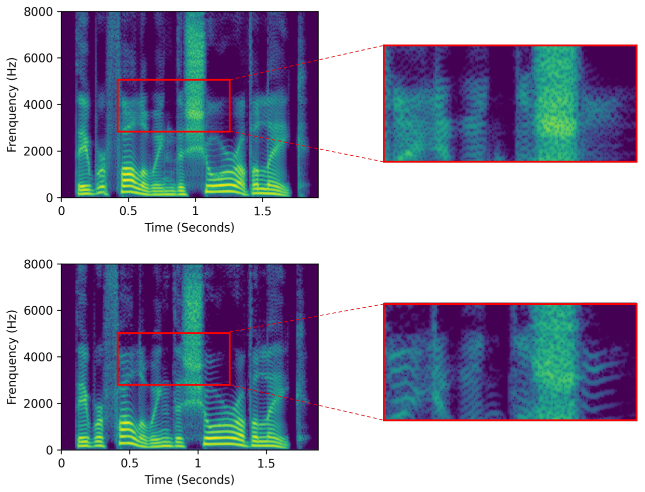

Fig. 7. The spectrograms of the audios synthesized by Mel-FS (above) and MSMC-TTS (below). The areas in the left red rectangles are zoomed in to the right part.

Mel-FS and MSMC-TTS. As shown in Fig. 7, the audio generated by Mel-FS shows a relatively fuzzy spectrogram. However, the audio generated by MSMC-TTS presents smoother and clearer harmonics in low-frequency and middle-frequency parts. It makes the pronunciations clearer and improves fidelity.

2) VQ-VAE based Approaches: Some previous works [64–66] attempt to enhance TTS using VQ-VAE approaches but mainly differ in training criteria and representation design from our work. For example, the vanilla VQ-VAE (single-stage, single-codebook) is commonly used to extract discrete units from the speech. Then, acoustic models are trained using cross-entropy to translate the text into these units. This section compares TTS models using different training criteria and representation configurations. All models are trained on Nancy in this experiment.

We first compare acoustic models trained with different loss functions. The single-stage single-codebook representation, i.e. vanilla VQR, is adopted as the output of acoustic models in this experiment. For models trained with cross-entropy, the predicted vector is a 512-dim posterior probability distribution to the codebook, and is processed by a Softmax function for normalization. The codeword with the highest probability will be chosen as the output vector. $\mathcal{L}_{msp}$ with $\gamma = 0$ means the Mean Squared Error (MSE), i.e. only minimizing the Euclidean distance. As shown in Table II, CrossEntropy shows the worse results with an MOS of 3.79. It cannot sufficiently consider the relationships among different codewords in the continuous space, producing severe problems in smoothness when classification fails. When MSE is applied as the loss function, the MOS is significantly improved to 4.13, validating the effectiveness of estimating codewords in the continuous space. The "triplet loss" with a weight of 1 further improves the MOS to 4.23, enabling TTS to generate more expressive speech. But the choice of an appropriate weight is also essential. Too-large weight, e.g. $\mathcal{L}_{msp}(\gamma = 100)$ with the MOS of 4.19, may degrade the quality instead. The overly high expressiveness also causes unnatural prosody in the synthesized speech.

TABLE II
MOS TEST: COMPARISON OF DIFFERENT LOSS FUNCTIONS

Method	γ	MOS (95% CI)
CrossEntropy	-	3.79 ± 0.08
$\mathcal{L}_{msp}$	0	4.13 ± 0.07
$\mathcal{L}_{msp}$	1	4.23 ± 0.08
$\mathcal{L}_{msp}$	100	4.19 ± 0.08

Secondly, to show the effectiveness of multi-codebook VQ and multi-stage modeling in TTS. We compare MSMC-TTS based on three different MSMCRs: One-Stage One-Codebook (S1C1), One-Stage Four-Codebook (S1C4), and Two-Stage Four-Codebook (S2C4). S1C1 is the vanilla VQ-VAE commonly used in previous VQ-VAE based TTS approaches. S1C4 uses 4 codebooks in multi-head quantization. S2C4 extends S1C4 to two stages with the down-sampling rates of 1 and 4. Table III shows the result of the MOS test. S1C1, quantized by only one codebook, has the highest compactness and obtains the highest DER in TTS. But its low completeness also limits TTS synthesis, leading to the lowest MOS of 3.74. By applying MHVQ, we can better balance the compactness and the completeness of VQR, and significantly enhance TTS to the MOS of 4.36 with a slightly reduced DER. When multi-stage representation S2C4 is applied, TTS is further enhanced to the MOS of 4.61, with a noticeable improvement in prosody at both coarse-grain and fine-grain levels.

TABLE III
SUBJECTIVE TEST: MSMC-TTS WITH DIFFERENT REPRESENTATIONS

Name	MOS (95% CI)	DER (%)	
		Train Set	Test Set
S1C1	3.74 ± 0.09	44.37	45.44
S1C4	4.36 ± 0.08	33.32	37.20
S2C4	4.61 ± 0.07	26.81	30.47

Fig. 8 shows spectrograms and pitch contours of audios synthesized by S1C1 and S2C4 based TTS systems. In the single-stage TTS, pitch generation is not stable enough, especially when switching syllables, as shown in **the red part**. The representation is easy to over-fit the short-time information, and pays less attention to the larger context, causing discontinuous prosody. But this problem can be solved in S2C4 based TTS system, as shown in the right part of the figure. Multi-stage modeling and prediction force the model to pay sufficient attention to short- and long-time contextual information at different time resolutions. Hence, the prosody becomes more smooth and more natural.

C. Resource-Limited Scenarios

MSMC-TTS has been shown as a better approach to producing high-quality speech under the scenario with high-complexity models and sufficient data. To further investigate the requirements of MSMC-TTS for modeling complexity and data size, we evaluate it in resource-limited scenarios, including lightweight TTS and low-resource TTS.

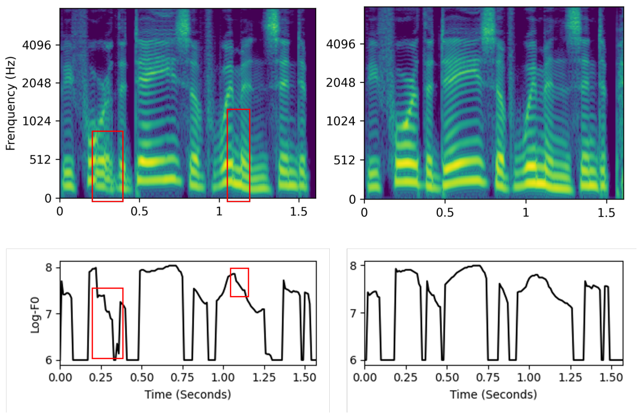

Fig. 8. The spectrograms and pitch contours of the waveforms synthesized from S1C4 (left) and S2C4 (right) based TTS systems. The red parts emphasize some problems in the audio.

1) Lightweight TTS: In this experiment, we evaluate the impact of the decreased modeling complexity on the proposed approach in lightweight TTS using fewer parameters or simpler model structures. We conduct another MOS test to compare Mel-FS and MSMC-TTS systems with these three model configurations as the encoder or decoder of the acoustic model:

- M1: 4 layers of 600-dim Transformer blocks.
- M2: 3 layers of 128-dim Transformer blocks.
- M3: 4 layers of 128-dim 1-D convolutional layers.

where M1 is the standard FastSpeech model with the most parameters, M2 is a lightweight version of M1 with fewer parameters, and M3 is more lightweight with a more simplified structure by replacing Transformer blocks with CNN.

The test results are shown in Table IV. The output quality of Mel-FS is seriously degraded from 4.08 to 2.10 when the parameters reduce from 72.50 MB to 1.75 MB. When Transformer blocks are replaced with the less complex CNNs, the MOS decreases further to 1.86. Both intelligibility and fidelity are seriously affected by the reduced modeling complexity. However, MSMC-TTS still performs well in all situations with only slight degradation as the parameters are reduced. MSMC-TTS trained with M3 (fewest parameters and simplest structure) still significantly outperforms Mel-FS with M1.

DERs of these models also show that low-performance models are easier to produce speech with a more significant discrepancy to the ground-truth data. As for the Mel-spectrogram following the distribution far from the target one, the vocoder cannot synthesize it sufficiently, producing more noise and seriously degrading the overall quality, including fidelity, intelligibility, and naturalness. However, in MSMC-TTS, the fidelity is always kept high, even if DER is reduced to 9.82. Since the vocoder sees almost all codewords in training, the biased distribution only slightly degrades prosody and intelligibility. In general, the proposed approach shows a much lower requirement for the modeling complexity of the acoustic model, and offers greater potential in lightweight scenarios, like on-device TTS [67].

TABLE IV
TTS EVALUATION: LIGHTWEIGHT TTS

Model	Param (MB)	MOS (95% CI)	DER(%) Train Set	DER(%) Test Set
Mel-M1	72.50	4.08 ± 0.10	1.14	5.09
Mel-M2	1.75	2.10 ± 0.09	0.90	1.18
Mel-M3	0.52	1.86 ± 0.09	0.40	0.78
MSMC-M1	116.58	4.65 ± 0.08	26.81	30.47
MSMC-M2	4.81	4.58 ± 0.08	13.34	15.39
MSMC-M3	3.12	4.47 ± 0.09	8.88	9.82

2) Low-Resource TTS: Except for the lightweight TTS with limited modeling resources, we also evaluate the performance of the proposed method in low-resource TTS [68] to investigate the impact of the data size on it. In this experiment, we build two more low-resource TTS datasets based on Nancy, which are described as follows:

- D1: 1,000 pairs of text and audio.
- D2: 1,000 pairs of text and audio + 10,000 audios without transcripts.

D1 means that all models in the TTS system, including the feature analyzer, acoustic model, and vocoder, can be only trained with 1000 utterances with both transcript and audio. However, in D2, except for the acoustic model trained with 1000 utterances, the feature analyzer and vocoder are trained with 11,000 audios.

TABLE V
TTS EVALUATION: LOW-RESOURCE TTS

Model	Dataset	MOS (95% CI)
Mel-FS	D1	3.28 ± 0.09
	D2	3.42 ± 0.08
MSMC-TTS	D1	4.59 ± 0.07
	D2	4.74 ± 0.07

Fig. V shows the result of the MOS test. Mel-FS with D1 only obtains the MOS of 3.28, showing unnatural prosody and insufficiently clear pronunciation. D2 improves this system to the MOS of 3.42. Although the acoustic model is still trained with limited data, the vocoder trained with 11,000 audios helps synthesize higher-fidelity audio for low-quality prediction. MSMC-TTS with D1 already performs better with an MOS of 4.59 over Mel-FS with D1 or D2. Both prosody and fidelity are presented well. After using D2 to train the feature analyzer and vocoder, this system obtains a higher MOS of 4.74. This result validates that MSMC-TTS has a lower data requirement.

VII. ANALYSIS-SYNTHESIS EVALUATION

In this section, we compare different speech representations via analysis-synthesis. Models are trained with the Nancy dataset to evaluate their performance in single-speaker modeling.

A. Evaluation Metrics

Representations are mainly evaluated in terms of compactness, completeness. Compactness refers to representing speech

TABLE VI

OBJECTIVE METRICS FOR EVALUATING THE COMPACTNESS AND COMPLETENESS OF DIFFERENT SPEECH REPRESENTATIONS. Mel REFERS TO THE MEL-SPECTROGRAM. Z_1 - Z_9 ARE VECTOR-QUANTIZED SPEECH REPRESENTATIONS WITH DIFFERENT CONFIGURATIONS, INCLUDING CODEBOOK SIZE M, NUMBER OF HEADS H, NUMBER OF STAGES S, AND DOWN-SAMPLE RATE IN EACH STAGE $\mathbf{r}_d$.

Name	Model Configuration				Compactness		Completeness			
	M	H	S	$\mathbf{r}_d$	$\mathcal{R}$	Bitrate (bps)	PESQ ↑	MCD(dB) ↓	F0-RMSE(Hz) ↓	F0-VUV(%) ↓
Mel	-	-	-	-	-	204800	3.91	1.62	2.11	1.82
Z_1	128	1	1	[1]	365.71	560	3.32	2.90	3.87	3.23
Z_2	256	1	1	[1]	320	640	3.39	2.82	3.68	3.20
Z_3	512	1	1	[1]	284.44	720	3.41	2.79	3.52	3.09
Z_4	512	2	1	[1]	142.22	1440	3.63	2.43	2.74	2.61
Z_5	512	4	1	[1]	71.11	2880	3.74	2.19	2.43	2.38
Z_6	512	16	1	[1]	17.78	11520	3.84	1.87	2.24	2.08
Z_7	512	4	2	[1,4]	56.89	3600	3.72	2.22	2.42	2.33
Z_8	512	4	3	[1,2,2]	40.63	5040	3.58	2.43	3.11	2.51
Z_9	512	4	3	[1,4,4]	51.72	3960	3.61	2.46	2.69	2.63

with fewer parameters. Hence, we use both compression ratio $\mathcal{R}$ and the bitrate (bites per second, bps) to represent compactness. The completeness is measured via analysis-synthesis. All MSMCRs are decoded to the Mel-spectrogram, and then converted to audio via a pre-trained vocoder. The better the reconstructed audio, the higher the feature completeness. There are four objective metrics used to evaluate the reconstruction quality – Perceptual Evaluation of Speech Quality (PESQ) [69], Mel Cepstral Distortion (MCD) [70], F0-RMSE, and F0-VUV [71]. PESQ can rate the audio to assess the voice quality perceived by human beings. MCD is a measurement widely used in speech synthesis [72]. It calculates the difference between two speech signals in Mel cepstral domain. F0-RMSE and F0-VUV (voiced / unvoiced error rate) focus on the difference between two audios in prosody and tone, which are important for speech. Pitch sequences with a frameshift of 5 ms are extracted from the audio to measure these two terms.[6]

B. Results

Table VI shows the experimental results of analysis-synthesis of nine representations with different configurations in terms of the codebook size M, the number of heads H, the number of stages K, and downsampling rates $\mathbf{r}_d$ for each stage. The acoustic feature Mel shows the upper bound of feature completeness with the highest PESQ of 3.91 and the lowest MCD of 1.62 dB, F0-RMSE of 2.11 Hz, and F0-VUV of 1.82%. It also shows the worst compactness with the highest bitrate of 204800 bps. $Z_1 - Z_9$ are all vector-quantized representations learned from Mel, hence showing lower reconstruction quality.

1) VQ-VAE: The single-codebook single-stage representation Z_1 is quantized from Mel using only 128 codewords, showing the highest compactness with the highest compression ratio of 365.71 and the lowest bitrate of 560 bps. However, it also achieves the worst completeness with the lowest PESQ of 3.32 and the highest MCD, F0-RMSE, and F0-VUV of 2.90 dB, 3.87 Hz, and 3.23%, resulting in low-quality speech reconstruction. As the codebook size increases to 512 in Z_3, the bitrates slowly increase from 560 to 900. The completeness is also improved slightly to the MCD of 2.79 dB, which is only 0.11 less than Z_1. It shows the difficulty of controlling completeness by only changing the codebook size, hence limiting its capability in TTS synthesis.

2) MHVQ: When MHVQ is applied, this problem is alleviated significantly. As the number of heads increases, the bitrate increases proportionally. Meanwhile, the completeness is also improved more obviously. Z_3 with 4x codebook size of Z_1 can only decrease MCD by 0.11 dB. But Z_5 with 2-head vector quantization can already decrease the MCD by 0.36 dB further over Z_3. Similarly, PESQ, F0-RMSE, and F0-VUV are also improved significantly. And the completeness can be further improved by giving more VQ heads to increase the bit rate. It validates MHVQ as a more effective method of improving completeness over enlarging the codebook size. The TTS model based on Z_5, i.e. S1C4 in the MOS test of Table III, also shows that keeping sufficient completeness of the compact representation, VQR, is essential for high-quality TTS synthesis.

3) Multi-Stage Modeling: Z_7 - Z_9 are implemented based on Z_5, i.e. using four heads and more stages. Different from MHVQ, these multi-stage representations aim to disentangle the speech sequence into different stages while decreasing the reconstruction loss in training. To balance these two objectives, the reconstruction quality from these representations decreases, although the compactness is slightly lower than Z_5. Hence, for multi-stage modeling, we also need to investigate if the speech is represented in stages in these representations. First, we propose analyzing information at different stages by reconstructing the Mel-spectrogram in different modes. In decoding, the model can be given the predicted sequence $\hat{\mathbf{z}}^{(i)}$ or the ground-truth sequence $\mathbf{z}^{(i)}$ at i-th stage. For instance, to fully reconstruct speech from Z_7, ground-truth $\mathbf{z}^{(2)}$ and $\mathbf{z}^{(1)}$ at two stages should be given to corresponding decoders. But we can also generate the speech with only high-stage information by giving $\mathbf{z}^{(2)}$ and $\hat{\mathbf{z}}^{(1)}$ predicted from $\mathbf{z}^{(2)}$. In this way, we can analyze each stage through the generated audio.

We first calculate objective metrics to audio generated in different modes for Z_7 and Z_9, as shown in Table VII. In the “Stage” column, “P” and “G” denote that the sequence at the corresponding stage is predicted or ground truth. The sequence $\hat{\mathbf{z}}^{(i)}$ at any stage, except for the highest stage, can be predicted by a higher-stage sequence $\hat{\mathbf{z}}^{(i+1)}$. Hence, in Z_7-

[6]The tools to calculate PESQ, MCD are available at https://github.com/ludlows/python-pesq, https://github.com/MattShannon/mcd. The tool to extract pitch is available at https://github.com/r9y9/pysptk.

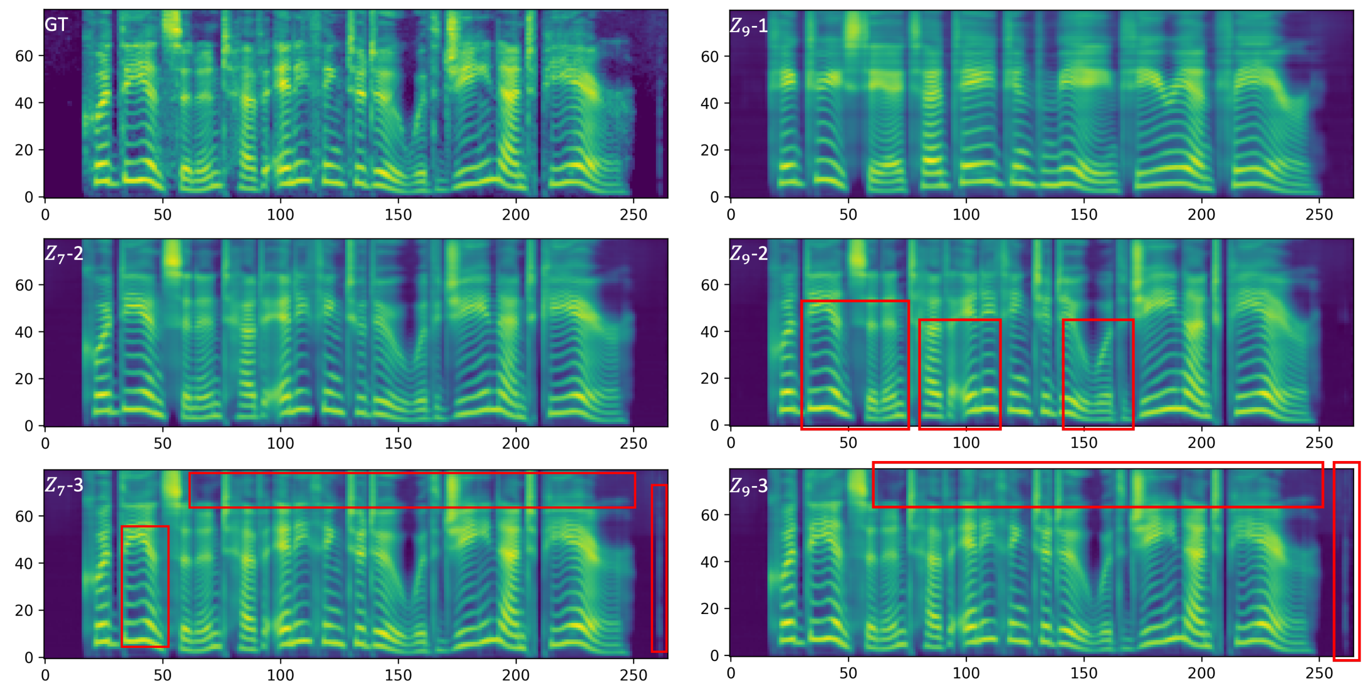

Fig. 9. Mel-spectrograms generated in different modes. The left column accordingly shows the ground-truth spectrogram, and reconstructed spectrograms from Z_7-2, Z_7-3. And the right column shows the reconstructed spectrograms from Z_9-1, Z_9-2, and Z_9-3. The red rectangles emphasize some areas with salient differences.

TABLE VII
OBJECTIVE METRICS OF MSMCRS IN DIFFERENT MODES.

Name	Stage			PESQ	MCD	F0-RMSE	F0-VUV
	1	2	3				
Z_7-1	P	P	-	0.17	15.59	59.23	44.85
Z_7-2	P	G	-	2.62	4.03	6.43	4.25
Z_7-3	G	G	-	3.72	2.22	2.42	2.33
Z_7-4	G	P	-	0.65	10.50	40.31	43.67
Z_9-1	P	P	G	0.80	8.94	21.85	11.03
Z_9-2	P	G	G	2.17	4.34	10.08	5.34
Z_9-3	G	G	G	3.61	2.46	2.69	2.63

1, $\hat{\mathbf{z}}^{(2)}$ is not predictable, we can only generate it by random sampling from the codebook. The generated speech carries nothing related but the target timbre, showing the lower bound of these metrics. Given only the ground-truth sequence $\mathbf{z}^{(2)}$, the reconstructed speech of Z_7-2 shows the much higher PESQ and lower MCD, F0-RMSE, F0-VUV. It shows that the high stage carries much information about the target speech. Then, in Z_7-3, these metrics are further improved given $\mathbf{z}^{(1)}$, and the speech is fully reconstructed. It shows that the low stage contains information that the high stage does not. Moreover, the awful reconstruction quality of Z_7-4 shows that high-stage information is also not covered in the low stage. The low-stage sequence is a supplement to the high-stage. This pattern is also reflected in Z_9 with three stages. With more ground-truth latent sequences, speech reconstruction quality is gradually improved. But the low reconstrution quality of Z_9-1 also shows the highest stage in Z_9 contains less valuable information, hence our TTS experiments are mainly based on the two-stage MSMCR.

To further investigate the information in different stages, we visualize Mel-spectrograms reconstructed from different representations, as shown in Fig. 9. First, for the two-stage representation Z_7, Z_7-2 with only the high-stage information shows a coarse version of the target speech with over-smoothed harmonics and blurry high-frequency information. It lacks sufficient details, leading to the unclear pronunciation of the reconstructed audio. Hence, the high-stage sequence tends to preserve more coarse-grained or more abstract information. Z_7-3 with low-stage information shows a clearer spectrogram with richer fine-grained information, including detailed high-frequency components, short-time pitch changes, and even some noise in silent parts. The low-stage modeling focuses on capturing low-level or local information that high-stage information lacks. And the capability of multi-stage modeling in capturing information at different levels is more salient in the three-stage representation Z_9. Due to a 16x reduction in time resolution, the spectrogram reconstructed from Z_9-1 with only the highest-stage information is much smoother with average pitch changes and no high-frequency details. With the introduction of the second-stage sequence, the spectrogram reconstructed from Z_9-2 recovers most fine-grained changes in pitch. Finally, the lowest-stage sequence with more low-level information helps recover more fine-grained details in the spectrogram reconstructed from Z_9-3.

In conclusion, MSMCR can preserve speech information at different levels via multi-stage modeling. And this approach also further enhances TTS. As shown in Table III, MSMC-TTS based on Z_7-3, i.e. S2C4, shows the highest MOS of 4.61, significantly outperforming TTS approaches based on single-stage representations.

VIII. Conclusion

This paper proposes MSMC-TTS, a compact speech representation based TTS system. The proposed feature analyzer converts Mel-spectrograms to MSMCRs composed of sequences at different time resolutions and quantized by multiple codebooks. A multi-stage predictor is trained as the acoustic model to better predict MSMCRs by minimizing the combined loss of MSE and "triplet loss". In TTS evaluation, experimental results of TTS evaluation on the English and Chinese datasets show that MSMC-TTS can predict features with minor domain discrepancy and generate higher-quality audio over baseline systems. The MSMCR and the proposed "triplet loss" also show their effectiveness in enhancing TTS based on VQ-VAE. Moreover, in applications of lightweight and low-resource TTS, the proposed system still performs well even with much fewer model parameters or training data, showing lower requirements for the modeling complexity and the data size. Finally, experimental results of analysis-synthesis validate that MHVQ is a more practical approach in improving feature completeness, and multi-stage modeling effectively represent the speech sequence with multiple sequences at different time resolutions.

References

[1] H. Guo, S. Zhang, F. K. Soong, L. He, and L. Xie, "Conversational end-to-end TTS for voice agents," in *Proc. SLT*. IEEE, 2021, pp. 403–409.

[2] C. Hu, Q. Tian, T. Li, W. Yuping, Y. Wang, and H. Zhao, "Neural dubber: Dubbing for videos according to scripts," *Proc. NeurIPS*, vol. 34, 2021.

[3] D. Wang, S. Yang, D. Su, X. Liu, D. Yu, and H. Meng, "VCVTS: Multi-speaker video-to-speech synthesis via cross-modal knowledge transfer from voice conversion," in *Proc. ICASSP*. IEEE, 2022, pp. 7252–7256.

[4] D. Wang, J. Yu, X. Wu, S. Liu, L. Sun, X. Liu, and H. Meng, "End-to-end voice conversion via cross-modal knowledge distillation for dysarthric speech reconstruction," in *Proc. ICASSP*. IEEE, 2020, pp. 7744–7748.

[5] Y. Jia, R. J. Weiss, F. Biadsy, W. Macherey, M. Johnson, Z. Chen, and Y. Wu, "Direct speech-to-speech translation with a sequence-to-sequence model," in *Proc. Interspeech*, 2019.

[6] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," *arXiv preprint arXiv:2106.15561*, 2021.

[7] E. Azarov, M. Vashkevich, and A. Petrovsky, "Instantaneous pitch estimation based on RAPT framework," in *Proc. EUSIPCO*. IEEE, 2012, pp. 2787–2791.

[8] F. Soong and B. Juang, "Line spectrum pair (LSP) and speech data compression," in *Proc. ICASSP*, vol. 9. IEEE, 1984, pp. 37–40.

[9] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis-a unified approach to speech spectral estimation." in *Proc. ICSLP*, vol. 94. Citeseer, 1994, pp. 18–22.

[10] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio." in *Proc. SSW*, 2016, pp. 125–125.

[11] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. v. d. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proc. ICML*, 2018.

[12] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *Proc. ICASSP*. IEEE, 2019, pp. 3617–3621.

[13] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Proc. NeurIPS*, 2019, pp. 14 910–14 921.

[14] Y. Ijima, T. Asami, and H. Mizuno, "Objective evaluation using association between dimensions within spectral features for statistical parametric speech synthesis." in *Proc. Interspeech*, 2016, pp. 337–341.

[15] Y. Saito, S. Takamichi, and H. Saruwatari, "Statistical parametric speech synthesis incorporating generative adversarial networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 84–96, 2017.

[16] J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, R. Barra-Chicote, A. Moinet, and V. Aggarwal, "Towards achieving robust universal neural vocoding," in *Proc. Interspeech*, 2019.

[17] D. Paul, Y. Pantazis, and Y. Stylianou, "Speaker conditional waveRNN: Towards universal neural vocoder for unseen speaker and recording conditions," in *Proc. Interspeech*, 2020.

[18] W. Jang, D. C. Y. Lim, and J. Yoon, "Universal MelGAN: A robust neural vocoder for high-fidelity waveform generation in multiple domains," in *Proc. ISCSLP*, 2022, pp. 71–75.

[19] Y. Jiao, A. Gabrys, G. Tinchev, B. Putrycz, D. Korzekwa, and V. Klimkov, "Universal neural vocoding with parallel wavenet," in *Proc. ICASSP*, 2021, pp. 6044–6048.

[20] H. Zen, "Acoustic modeling in statistical parametric speech synthesis-from HMM to LSTM-RNN," in *Proc. MLSLP*, 2015.

[21] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. Interspeech*, 2014.

[22] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, "FastSpeech: Fast, robust and controllable text to speech," in *Proc. NeurIPS*, 2019, pp. 3165–3174.

[23] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, "FastSpeech 2: Fast and high-quality end-to-end text to speech," in *Proc. ICLR*, 2021.

[24] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proc. AAAI*, vol. 33, no. 01, 2019, pp. 6706–6713.

[25] H. Guo, F. K. Soong, L. He, and L. Xie, "A New GAN-Based End-to-End TTS Training Algorithm," *Proc. Interspeech*, pp. 1288–1292, 2019.

[26] J. Yang, J.-S. Bae, T. Bak, Y.-I. Kim, and H.-Y. Cho, "GANSpeech: Adversarial training for high-fidelity multi-speaker speech synthesis," in *Proc. Interspeech*, 2021, pp. 2202–2206.

[27] H. Guo, H. Lu, X. Wu, and H. Meng, "A multi-scale time-frequency spectrogram discriminator for GAN-based non-autoregressive TTS," in *Proc. Interspeech*, 2022.

[28] C. Miao, S. Liang, M. Chen, J. Ma, S. Wang, and J. Xiao, "Flow-TTS: A non-autoregressive network for text to speech based on flow," in *Proc. ICASSP*. IEEE, 2020, pp. 7209–7213.

[29] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-TTS: A Generative flow for text-to-speech via monotonic alignment search," in *Proc. NeurIPS*, 2020.

[30] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural TTS synthesis by conditioning wavenet on Mel spectrogram predictions," in *Proc. ICASSP*. IEEE, 2018, pp. 4779–4783.

[31] P. Neekhara, J. Li, and B. Ginsburg, "Adapting TTS models For New Speakers using Transfer Learning," *ArXiv*, vol. abs/2110.05798, 2021.

[32] S. Ö. Arık, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman *et al.*, "Deep voice: Real-time neural text-to-speech," in *Proc. ICML*. PMLR, 2017, pp. 195–204.

[33] R. J. Weiss, R. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis," in *Proc. ICASSP*. IEEE, 2021, pp. 5679–5683.

[34] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *Proc. ICML*. PMLR, 2021, pp. 5530–5540.

[35] M.-J. Hwang, R. Yamamoto, E. Song, and J.-M. Kim, "TTS-by-TTS: TTS-driven data augmentation for fast and high-quality speech synthesis," in *Proc. ICASSP*. IEEE, 2021, pp. 6598–6602.

[36] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *International work-conference on artificial neural networks*. Springer, 2005, pp. 758–770.

[37] R. Liu and D. F. Gillies, "Overfitting in linear feature extraction for classification of high-dimensional image data," *Pattern Recognition*, vol. 53, pp. 73–86, 2016.

[38] B. Labbé, R. Hérault, and C. Chatelain, "Learning deep neural networks for high dimensional output problems," in *Proc. ICMLA*. IEEE, 2009, pp. 63–68.

[39] W. B. Kleijn and D. Talkin, "Compact speech representations for speech synthesis," in *Proc. SSW*. IEEE, 2002, pp. 35–38.

[40] J. Chorowski, R. J. Weiss, S. Bengio, and A. Van Den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 12, pp. 2041–2053, 2019.

[41] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
[42] M. Tschannen, O. Bachem, and M. Lucic, “Recent advances in autoencoder-based representation learning,” *arXiv preprint arXiv:1812.05069*, 2018.
[43] Y. Wang, H. Yao, and S. Zhao, “Auto-encoder based dimensionality reduction,” *Neurocomputing*, vol. 184, pp. 232–242, 2016.
[44] X. Luo, Y. Xu, W. Wang, M. Yuan, X. Ban, Y. Zhu, and W. Zhao, “Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy,” *Journal of The Franklin Institute*, vol. 355, no. 4, pp. 1945–1966, 2018.
[45] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in *Proc. NeurIPS*, 2017.
[46] C. Gârbacea, A. van den Oord, Y. Li, F. S. Lim, A. Luebs, O. Vinyals, and T. C. Walters, “Low bit-rate speech coding with VQ-VAE and a WaveNet decoder,” in *Proc. ICASSP*. IEEE, 2019, pp. 735–739.
[47] Y. Chen, S. Yang, N. Hu, L. Xie, and D. Su, “TeNC: Low bit-rate speech coding with VQ-VAE and GAN,” in *ICMI*, 2021, pp. 126–130.
[48] E. A. AlBadawy, A. Gibiansky, Q. He, J. Wu, M.-C. Chang, and S. Lyu, “VocBench: A neural vocoder benchmark for speech synthesis,” *arXiv preprint arXiv:2112.03099*, 2021.
[49] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with vq-vae-2,” in *Proc. NeurIPS*, 2019, pp. 14 866–14 876.
[50] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.
[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. NeurIPS*, 2017, pp. 5998–6008.
[52] W.-N. Hsu, Y. Zhang, R. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, P. Nguyen, and R. Pang, “Hierarchical generative modeling for controllable speech synthesis,” in *Proc. ICLR*, 2019. [Online]. Available: https://arxiv.org/pdf/1810.07217
[53] G. Sun, Y. Zhang, R. J. Weiss, Y. Cao, H. Zen, and Y. Wu, “Fully-hierarchical fine-grained prosody modeling for interpretable speech synthesis,” in *Proc. ICASSP*, 2020, pp. 6264–6268.
[54] Y. Liu, Z. Xu, G. Wang, K. Chen, B. Li, X. Tan, J. Li, L. He, and S. Zhao, “DelightfulTTS: The Microsoft speech synthesis system for Blizzard challenge 2021,” *arXiv preprint arXiv:2110.12612*, 2021.
[55] R. Child, “Very deep VAEs generalize autoregressive models and can outperform them on images,” in *Proc. ICLR*, 2021.
[56] X. Dong and J. Shen, “Triplet loss in siamese network for object tracking,” in *Proc. ECCV*, 2018, pp. 459–474.
[57] S. King and V. Karaiskos, “The Blizzard Challenge 2011,” in *Blizzard challenge workshop*, 2011.
[58] M. He, Y. Deng, and L. He, “Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural TTS,” *Proc. Interspeech*, pp. 1293–1297, 2019.
[59] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.
[60] J. Kong, J. Kim, and J. Bae, “HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis,” in *Proc. NeurIPS*, 2020.
[61] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, “UnivNet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation,” in *Proc. Interspeech*, 2021.
[62] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proc. ICLR*, 2018.
[63] J. Kim, S. Kim, J. Kong, and S. Yoon, “Glow-TTS: A generative flow for text-to-speech via monotonic alignment search,” *Proc. NeurIPS*, vol. 33, pp. 8067–8077, 2020.
[64] T. Hayashi and S. Watanabe, “Discretalk: Text-to-speech as a machine translation problem,” *arXiv preprint arXiv:2005.05525*, 2020.
[65] Y. Yasuda, X. Wang, and J. Yamagishd, “End-to-end text-to-speech using latent duration based on vq-vae,” in *Proc. ICASSP*. IEEE, 2021, pp. 5694–5698.
[66] C. Du, Y. Guo, X. Chen, and K. Yu, “VQTTS: High-fidelity text-to-speech synthesis with self-supervised VQ acoustic feature,” in *Proc. Interspeech*, 2022.
[67] S. Achanta, A. Antony, L. Golipour, J. Li, T. Raitio, R. Rasipuram, F. Rossi, J. Shi, J. Upadhyay, D. Winarsky, and H. Zhang, “On-device neural speech synthesis,” in *Proc. ASRU*, 2021, pp. 1155–1161.
[68] A. Pine, D. Wells, N. Brinklow, P. Littell, and K. Richmond, “Requirements and motivations of low-resource speech synthesis for language revitalization,” in *Proc. ACL*, 2022, pp. 7346–7359.
[69] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, “Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs,” in *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.
[70] R. Kubichek, “Mel-cepstral distance measure for objective speech quality assessment,” in *Proceedings of IEEE pacific rim conference on communications computers and signal processing*, vol. 1. IEEE, 1993, pp. 125–128.
[71] Q. Zhang, F. Soong, Y. Qian, Z. Yan, J. Pan, and Y. Yan, “Improved modeling for F0 generation and V/U decision in HMM-based TTS,” in *Proc. ICASSP*, 2010, pp. 4606–4609.
[72] H. Ze, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *Proc. ICASSP*, 2013, pp. 7962–7966.

Haohan Guo (Student Member, IEEE) received the B.S. degree and the M.S. degree from the Northwestern Polytechnical University (NWPU), Xi’an, Shannxi Province, China, in 2017 and 2020, respectively. He has been working toward a Ph.D. degree at the Chinese University of Hong Kong (CUHK) since August 2021. Before that, he also worked at Sogou Inc., China, as an associate researcher. His research interests include speech synthesis, voice conversion, speech and language processing, and machine learning.

Fenglong Xie (Member, IEEE) received the B.S. and M.S. degrees in Computer Science and Technology from Harbin Institute of Technoloy in 2012, 2014, respectively. He received the Ph.D. degree in a joint Ph.D. program between Harbin Institute of Technology and Microsoft Research Asia in 2019. Currently He is the audio intelligence team leader of Xiaohongshu where he engages in research and product development in speech, music and audio intelligence. From 2018 to 2021, he was a senior researcher responsible for transferring speech synthesis technology from research to several products at WeChat, Tencent. Prior to that he worked at Microsoft Research Asia from 2012 to 2018 conduting fundamental research on speech synthesis and recognition. His research interests include speech synthesis, music information retrieval, speech signal processing and speech recognition.

Xixin Wu (Member, IEEE) received the B.S. degree from Beihang University, Beijing, China, the M.S. degree from Tsinghua University, Beijing, China, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong. He had been a Research Associate with the Machine Intelligence Laboratory, Cambridge University Engineering Department, and from 2021 has been a Research Assistant Professor with the Stanley Ho Big Data Decision Analytics Research Centre, The Chinese University of Hong Kong. His research interests include speech synthesis and recognition, speaker verification, and neural network uncertainty. He is a Member of ISCA.

Frank K. Soong (Fellow, IEEE) (Fellow, IEEE) received the B.S. degree from National Taiwan University, Taipei, Taiwan, the M.S. degree from the University of Rhode Island, South Kingstown, RI, USA, and the Ph.D. degree from Stanford University, Stanford, CA, USA, all in electrical engineering. He retired from Microsoft Research Asia, Beijing, China, in 2022. He has worked on fundamental research on speech and its practical applications, including Automatic Speech Recognition and Speaker Recognition,Text-to-Speech synthesis and Computer Assisted Language Learning. He has been a Visiting Professor with the Chinese University of Hong Kong, Hong Kong, and a few other top-rated universities in China. He has authored or coauthored more than 300 papers and co-edited a widely used reference book, Automatic Speech and Speaker Recognition-Advanced Topics, Kluwer, 1996. His professional research career spans four decades, first with Bell Labs, Acoustics and Speech Research, Murray Hill, NJ, USA, then with ATR Interpreting Telephony Lab, Kyoto, Japan, before joining MSRA in 2004. He was a Member of the Speech and Language Technical Committee, IEEE Signal Processing Society and other IEEE functions, including an Associate Editor for the IEEE Speech and Audio Transactions. He is an IEEE Fellow for contributions to digital processing of speech.

Helen Meng (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA. In 1998, she joined the Chinese University of Hong Kong, Hong Kong, where she is currently a Patrick Huen Wing Ming Professor with the Department of Systems Engineering & Engineering Management. She was the former Department Chairman and the Associate Dean of Research with the faculty of Engineering. Her research interests include human-computer interaction via multimodal and multilingual spoken language systems, spoken dialog systems, computer-aided pronunciation training, speech processing in assistive technologies, health-related applications, and big data decision analytics. She was the Editor-in-Chief of the IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING between 2009 and 2011. She was the recipient of the IEEE Signal Processing Society Leo L. Beranek Meritorious Service Award in 2019. She was also an Elected Board Member of the International Speech Communication Association (ISCA) and an ISCA International Advisory Board Member. She is a Fellow of ISCA, HKCS, HKIE and IEEE.