ULF-TTS: An Uncluttered Hybrid TTS System using Language and Flow Matching Models

Jae Hyun Park, Seung Jae Choi, Young Sik Eom, Allison Shindell, Min-Gwan Seo and Gyeong-hoon Lee
NC AI Co., Ltd
E-mail: {pkjh0219, seung, yseom, shindell, mgseo, ghlee0304}@ncsoft.com

***Abstract*—Hybrid text-to-speech (TTS) systems that integrate a language model with flow matching decoders have been proposed to generate speech in voice-related applications such as dubbing. However, these systems suffer from latency and computational overhead caused by next-token prediction and denoising steps. Therefore, we propose an uncluttered hybrid TTS system that integrates a lightweight flow matching decoder and supports variable token prediction time. In addition, we leverage a training strategy from a diffusion-based approach into the language model to enhance its generative performance. The results of objective and subjective evaluations demonstrate that our proposed method is significantly more efficient than existing hybrid systems while maintaining competitive speech quality. Online demo is available at https://nc-ai.github.io/speech/publications/ulftts/index.html.**

I. Introduction

Large-scale text-to-speech (TTS) systems [1]–[3] employ the impressive performance of large language models (LLM) by utilizing a neural audio codec [4]–[6]. Previous studies [1], [2] first used the codec using vector quantization to convert a speech sequence into a discrete token sequence, which captures text-speech relationships while preserving acoustic information. The large language model then learns to predict the token sequence based on the text and a speech sequence prompt. Specifically, when the text sequence and speaker attributes are provided as the prompt, the language model predicts the token sequence autoregressively by leveraging its strong contextual understanding capabilities.

Other studies [7]–[9] have investigated an alternative approach based on flow matching. This method learns the distribution of the mel-spectrogram and samples the distribution from a simple distribution through a flow-based matching process. To achieve this, the model reconstructs randomly masked regions by conditioning on speech and text prompt representations, effectively leveraging contextual information for accurate restoration. However, unlike language model-based approaches, the flow matching approach can only learn speech sequences with fixed durations and requires the use of special tokens, such as filler tokens [10], to predict the end of the sequence.

Recently, the hybrid TTS systems [3], [11]–[13] have been introduced to effectively leverage the advantages of both approaches by combining the language model-based and continuous generative approaches. These systems use the language model to predict a discrete token sequence, while a decoder, either diffusion or flow matching based, generates a mel-spectrogram conditioned on the predicted token sequence. The discrete token sequence is extracted using tokenizers [12], [14], and the generated mel-spectrogram is then converted into a waveform by an off-the-shelf neural vocoder [15]. These TTS systems can generate high-fidelity and expressive speech, making them suitable for a wide range of applications, such as dubbing. However, they incur significant latency and computational overhead, primarily due to the next-token prediction process within the language model and the denoising steps in the flow matching decoder. Furthermore, since their language model predicts only one token at a time, reducing inference time is impractical. These limitations are critical for real-time applications.

As in [16], a natural and expressive mel-spectrogram can be generated using only a language model on its own. Likewise, we utilize the language model and the tokenizer decoder to predict the token sequence and reconstruct the mel-spectrogram respectively, while the flow matching decoder is employed to enhance the overall quality. The flow matching decoder performs a significantly simpler task than the generation task, allowing it to operate with fast inference speeds and using fewer parameters, as in [17]. We also leverage Multi-token prediction [18], which predicts multiple future tokens all at once, enabling variable token prediction time. Therefore, we can flexibly optimize the trade-off between prediction accuracy and inference speed on-the-fly. Furthermore, according to [18], this approach contributes to accurate token prediction.

In this paper, we propose an efficient system built on the hybrid TTS system (**ULF-TTS**) by constructing a multi-modal block module [19] within the flow matching decoder using simple multi-layer perceptrons (MLPs), and employing a multi-token prediction approach to enable flexible token prediction time. Note that the role of our flow matching decoder is to enhance mel-spectrogram quality. This is a simpler task than generation, so we use an MLP architecture for efficiency. In addition, we utilize a diffusion training strategy to improve the prediction performance of the language model.

We compare our system with other hybrid systems through objective and subjective experiments in zero-shot scenarios. These experimental results demonstrate that our system is at least twice as fast as other systems, while achieving competitive speaker similarity and generating natural speech. Additionally, through internal experiments, we show that our system can optimize both the prediction time and the quality of the generated speech.

979-8-3315-7206-8/25/$31.00

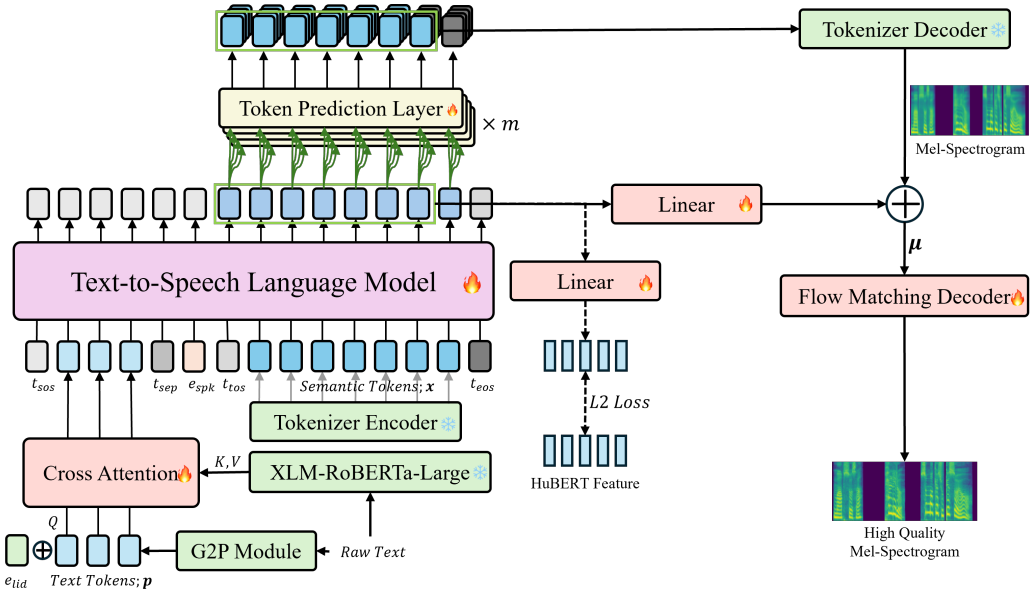

Fig. 1. The overview of our uncluttered hybrid TTS system, ULF-TTS.

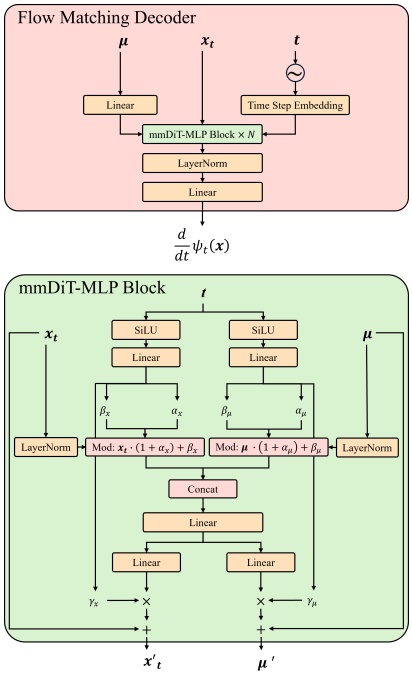

Fig. 2. Flow matching decoder and the multi-modal MLP block. The upper illustrates the architecture of the flow matching decoder, while the lower depicts the architecture of the multi-modal MLP block, which serves as the core component of the flow matching decoder.

II. ULF-TTS

An overview of the system is represented in Figure 1, which illustrates its main components: an autoregressive language model and a flow matching decoder. Note that we define the language model's purpose as generating the mel-spectrogram, while the flow matching decoder's role is to enhance its quality. Thus, flow matching decoder's role differs from conventional decoders [12], [13] designed for generation. The training process is divided into two stages: first, the tokenizer is trained, and then the language model and the flow matching decoder are jointly trained. This is covered in more depth in the following sections.

A. Tokenizer

A tokenizer aims to translate self-supervised learning (SSL) features, specifically HuBERT features [20], into a semantic token sequence $\boldsymbol{x}_{T:1} = \{x_1, ..., x_T\}$ and a mel-spectrogram.

We adopt the approach from [14] as our tokenizer, while the finite-scalar quantization (FSQ) module [21] is used for vector quantization, with 512 codewords and no frameshift. Speaker embeddings are extracted from a pre-trained speaker encoder [22], and language embeddings correspond to pre-defined language indices.

Compared to [14], we have three key distinctions. First, we replace a convolutional block with a transformer block as the core components of the tokenizer to better capture intricate relationships within sequential data. Second, we isolate acoustic information from the semantic token sequence by utilizing both speaker and language embeddings, representing acoustic attributes, as conditioning inputs for the decoder. Finally, we incorporate the SSIM loss [23] into the training objective to enhance the reconstruction capability of the mel-spectrogram. The training objective $\mathcal{L}_{token}$ is as follows:

$$\mathcal{L}_{token} = \lambda_S * \mathcal{L}_S + \lambda_A * \mathcal{L}_A + \lambda_{SSIM} * \mathcal{L}_{SSIM} \quad (1)$$

, where $\mathcal{L}_S$ and $\mathcal{L}_A$, as defined in [14], denote L2 loss between HuBERT features and the reconstructed features, as well as the loss between the mel-spectrogram and the reconstructed features respectively, while $\mathcal{L}_{SSIM}$ is SSIM loss. λ_S, λ_A and λ_{SSIM} are hyper-parameters, which are set to 1.

B. Autoregressive Language Model

Given the strong generative capabilities of language models in TTS [1], [3], we employ one to translate an input sequence $\boldsymbol{c}_{N:1} = \{c_1, ..., c_N\}$ comprising the phonetic sequence $\boldsymbol{p}_{U:1} = \{p_1, ..., p_U\}$ and the speaker embedding e_{spk} into the semantic token sequence. We use the Llama-style casual transformer [24] with 12 layers and a feature dimension of 1,024, while incorporating rotary positioning encoding [25] and RMSNorm [26] as the backbone. We convert raw text into phonetic sequences using grapheme-to-phoneme (G2P) modules and add them to the language embedding. The total sequence $\boldsymbol{s}$ is comprised of the input sequence $\boldsymbol{c}$, which includes the phonetic sequence $\boldsymbol{p}$, speaker embeddings e_{spk}, semantic tokens $\boldsymbol{x}$, and special tokens:

$$\boldsymbol{s} = \{t_{sos}, \boldsymbol{p}_{U:1}, t_{sep}, e_{spk}, t_{tos}, \boldsymbol{x}_{T:1}, t_{eos}\} \tag{2}$$

, where t_{sos}, t_{sep}, t_{tos}, and t_{eos} denote the start-of-sequence, separation, turn-of-speech, and EOS tokens, respectively. During training, the model optimizes the loss for the total sequence while excluding the loss from the input sequence through in-context learning. During inference, the model predicts the next token autoregressively, corresponding to the past token sequence by employing a sampling strategy with Gumbel-Softmax [27] until the EOS token is predicted.

A limitation of previous hybrid systems that predict one next token at a time is their inability to flexibly adjust the prediction time. Therefore, we utilize Multi-token prediction [18] in the language model. Specifically, during training, the model is designed to predict up to $m \in [1, T]$ future tokens, whereas it has the flexibility to predict up to m tokens at a time during inference. This enables on-the-fly processing, optimizing the balance between token prediction time and prediction accuracy.

Furthermore, we apply classifier-free guidance (CFG) [28], [29] to the language model. In [30], they demonstrated improved prediction performance not only in diffusion models but also in autoregressive-based generation models. During training, the input sequence is replaced with a null embedding e_{null} with a probability of 0.2. During inference, the logits ℓ of the language model are set to $\ell_c + \alpha(\ell_c - \ell_u)$, where ℓ_c represents the logits with condition, ℓ_u denotes the logits without condition, and α is the scaling factor of 1.0. We leverage the pre-trained XLM-RoBERTa [1] as in [31] to capture contextual information from the utterance. In addition, the model is trained on HuBERT features, which contain rich phonetic information, to reflect semantic information. The training objective of the language model $\mathcal{L}_{lm}$ is defined as follows:

$$\mathcal{L}_{lm} = ||f_L - f_H||_2 - \log \Sigma^T P(\boldsymbol{x}_{t+m:t+1} | x_{t:1}, \tilde{\boldsymbol{c}}_{n:1}; \theta_{lm}) \tag{3}$$

, where $||\cdot||_2$ represents L2 loss, f_L and f_H denote the last hidden features of the language model and HuBERT features respectively. The second term of $\mathcal{L}_{lm}$ represents the multi-token prediction cross-entropy loss mentioned in [18], and $\tilde{\boldsymbol{c}}$ represents the condition where $\tilde{\boldsymbol{c}}$ is randomly dropped and replaced with e_{null}.

[1] https://huggingface.co/FacebookAI/xlm-roberta-large

C. Flow Matching Decoder with Simple MLP

A flow matching decoder has strong generative capabilities in TTS [7], [8]. It defines a time-dependent probability path p_t that transforms a Gaussian noise distribution into the target mel-spectrogram distribution. Specifically, the probability distribution p_t transitions continuously over time t through a mechanism known as a flow $\phi_t(\boldsymbol{x})$, which is mathematically described by the ordinary differential equation (ODE):

$$\frac{d}{dt}\phi_t(\boldsymbol{x}) = v_t(\phi_t(\boldsymbol{x})); \quad \phi_0(\boldsymbol{x}) = \boldsymbol{x} \tag{4}$$

, where v_t denotes a vector field. Note that we follow the notation established in [32]. In [32], a neural network $v_t(\boldsymbol{x})$ parameterized with θ is trained to estimate u_t, which is a known vector field. This approach enables the generation of p_t from p_0 to $p_1 \approx q$. Conditional flow matching (CFM) aims to avoid the intractability of the distribution. We use optimal-transport conditional flow matching (OT-CFM) [32] to train the neural network to enable straightforward gradient updates and fast training.

$$L_{fm}(\theta) = E_{t,q,p_0(\boldsymbol{x}_0)} ||v_t^{OT}(\psi_t(\boldsymbol{x}|\boldsymbol{\mu})) - u_t^{OT}(\psi_t(\boldsymbol{x}_0)|\boldsymbol{x}_1)||^2 \tag{5}$$

, where ψ_t denotes an affine transform that maps to the normally distributed random variable, $\psi_t(\boldsymbol{x}) = (1 - (1 - \sigma_{min})t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$ and $u_t^{OT}(\psi_t(\boldsymbol{x}_0)|\boldsymbol{x}_1) = \boldsymbol{x}_1 - (1 - \sigma_{min})\boldsymbol{x}_0$. Note that unlike in [33], $\boldsymbol{\mu}$ denotes the mel-spectrogram condition for the flow matching decoder. We use σ_{min} of $1e-5$. The mel-spectrogram generated by the tokenizer and step t are fed into the neural network. In addition, the CFG is employed to effectively incorporate the given condition properties and to mitigate overfitting. We set the probability of the drop and the scale equal to Sec II-C.

In previous hybrid systems [12], [13], the roles of the two generative models in generating speech are redundant, resulting in unnecessary overhead e.g., the high latency from both the denoising and the next token prediction steps. To handle this, we clearly define the role of this generative model. Specifically, the flow matching decoder aims to improve the low-quality mel-spectrogram reconstructed from the tokenizer, enhancing it to a high-quality one. The task of improving quality requires significantly less capability than generation, making it feasible to implement using a simple MLP architecture. Therefore, following [17], we replace the flow matching decoder with a simple MLP architecture. We propose modifying the multi-modal-styled block (mmDiT) [19] into the simple MLP-based variant (mmDiT-MLP). Our system replaces the QKV-Attention module in the original mmDiT with the simple MLP architecture, and predicts both the condition $\boldsymbol{\mu}$ and the input $\boldsymbol{x}_t$ at each flow matching timestep t. Its prior distribution is initialized from Gaussian noise. As it is designed to refine the generated mel-spectrogram rather than to generate it from scratch, the mmDiT-MLP allows

TABLE I
TOTAL DURATION OF EACH LANGUAGE FOR TRAINING.

	KO	EN	TW	JA
Duration(hours)	1,221	185	56	26

for a more lightweight design compared to the U-Net-based architecture of the flow matching decoder used in previous hybrid systems. The proposed decoder is illustrated on the bottom side of Figure 2.

Similar to [11], which refines the language model-generated features through a diffusion model, we construct the local condition of the decoder by combining the total of the last hidden features of the language model with the mel-spectrogram, which is generated by the tokenizer's decoder. Since the decoder leverages the last learnable hidden feature, it allows for more flexible refinement than simply using the mel-spectrogram as a condition.

III. EXPERIMENTS

A. Experimental Settings

1) Datasets: The training dataset consists of four languages: English, Korean, Taiwanese, and Japanese. We used the open datasets LibriTTS(train-clean-100)[34], LJSpeech[35], and our internal datasets as our English dataset. The Korean dataset includes the open datasets AI-Hub with emotion and style [2] and our internal datasets. The Taiwanese and Japanese datasets are our internal datasets. All speech was resampled to 22.05kHz, with mel-spectrograms extracted using 80 frequency bins, a hop size of 256, and a window size of 1024. The total duration of each language for training is shown in Table I.

2) Training configuration: We used a batch size of 44, trained for 600M iterations with the AdamW optimizer [36] with an initial learning rate of 1e-4 for both the tokenizer and the model training. We used 4 A100 NVIDIA GPUs to train our model. We utilized G2P tools on the speech transcripts to obtain the phoneme sequence for four languages. We used the open-source G2P tool g2pe[3]; for English, our custom G2P tool for Korean, g2pw[4] for Taiwanese, and OpenJTalk[5] for Japanese. For additional configurations of the language model, the number of multi-token predictions (m) was set to 6 for training. During inference, we set m to 2, resulting in a prediction rate of 43Hz per step, which matches the conditions used in other hybrid systems. The number of mmDiT-MLP blocks (N) was set to 2 for the flow matching decoder We employed the pre-trained vocoder Avocodo [37] to convert the mel-spectrogram into a waveform.

3) Evaluation metrics: We conducted both objective and subjective evaluations for our model. We used a A100 NVIDIA GPU for all evaluations. For objective evaluation, we assessed the model's efficiency, intelligibility, and speaker similarity. Efficiency was measured using the average real-time factor (RTF) and the model's parameter count. We assessed intelligibility using 100 sampled utterances, measuring word error rate (WER), character error rate (CER), substitutions (Sub), and insertions/deletions (Ins.&Del.) with the pre-trained automatic speech recognition (ASR) model, Whisper [6]. Speaker similarity (SS) was measured as the raw cosine similarity between a prompt and its synthesized speech using a pre-trained speech encoder [7]. For subjective evaluation, we assessed the capabilities of speech naturalness and similarity. We conducted the mean opinion score (MOS) test via Amazon Mechanical Turk. To measure naturalness MOS (NMOS), including pronunciation and intonation, 20 native participants rated 30 speech samples on a scale of 1 to 5. Similarly, 20 native participants rated 60 speech samples for similarity MOS (SMOS).

TABLE II
ZERO-SHOT TTS EVALUATION RESULTS IN INTRA- AND CROSS-LINGUAL SCENARIOS. MOS RESULTS ARE WITH 95% CONFIDENCE INTERVALS.

	NMOS	SMOS	SS
	Intra-lingual		
GT	4.16±0.12	4.05±0.21	0.851
CosyVoice [12]	3.94±0.19	3.42±0.23	0.806
FireRedTTS [13]	3.81± 0.17	3.46±0.21	0.822
ULF-TTS ($m = 2$)	**4.01±0.17**	**3.58±0.21**	**0.824**
	Cross-lingual		
CosyVoice [12]	3.92±0.19	3.31±0.13	0.816
FireRedTTS [13]	3.75±0.19	3.38±0.21	0.822
ULF-TTS ($m = 2$)	**3.99±0.18**	**3.46±0.22**	**0.826**

IV. EXPERIMENTAL RESULTS

A. Zero-shot TTS

We evaluated the hybrid system's performance using a zero-shot TTS task, generating speech based on an unseen prompt. We compared our system with other pre-trained hybrid systems [12], [13]. Note that these models were trained on internal data; we only had access to inference code. Thus, we were not able to test under the same conditions. Since English is the only language supported by all systems, we conducted evaluations in English. We categorized speech prompts into intra- and cross-lingual scenarios. For the intra-lingual prompt, samples were randomly selected from LibriTTS-test-clean, whereas for the cross-lingual prompts, Chinese, Korean, and Japanese speech were collected from open datasets AI-Hub. Table II shows the evaluation results for its performance. In both intra- and cross-lingual scenarios, our system demonstrates the ability to generate speech that is more natural and closely aligned with the given prompt compared to other large-scale hybrid TTS systems. This indicates that the CFG and multi-token prediction, which enhance token prediction, help to generate natural speech related to the given prompt.

[2]https://aihub.or.kr
[3]https://github.com/Kyubyong/g2p
[4]https://github.com/GitYCC/g2pW
[5]https://github.com/r9y9/pyopenjtalk
[6]https://github.com/openai/whisper
[7]https://github.com/resemble-ai/Resemblyzer

TABLE III
THE EVALUATION RESULTS FOR MODEL EFFICIENCY.

	# Params	RTF(s)	
		LM	flow matching
CosyVoice [12]	415M	0.203	0.031
FireRedTTS [13]	760M	0.554	0.114
ULF-TTS ($m = 2$)	**302M**	**0.099**	**0.012**

TABLE IV
THE OBJECTIVE EVALUATION RESULTS FOR INTELLIGIBILITY.

	WER	CER	#SUB	#INS+DEL
GT	0.03	0.01	50	9
CosyVoice [12]	0.05	0.02	**56**	**16**
FireRedTTS [13]	0.1	0.07	67	108
ULF-TTS ($m = 2$)	**0.04**	**0.02**	62	26

B. Efficiency for Real-time processing

We measured the number of parameters of each model and the inference speed by randomly sampling 100 sentences from LibriTTS-test-clean. For a fair comparison, RTF is measured separately for the language model (LM) and the flow matching decoder, excluding the vocoder processing time. Table III represents that our system is the fastest, with significantly fewer parameters compared to other systems. Specifically, the flow matching decoder with the mmDiT-MLP block is approximately 10 times faster than [13]. This means that our system supports real-time processing using the flow matching decoder.

C. Intelligibility

We evaluated intelligibility using the 100 utterance samples used in Sec IV-C. These samples are challenging with long sentences and repeated words. Table IV shows that our hybrid system has performance comparable to [12], which models the ASR-based token sequence. This indicates that our system generates speech that accurately reflects the text prompt. Compared to [12], our system has a slight increase in substitution, insertion, and deletion errors. This difference is likely due to the difference in training times: [12] system was trained with 30,000 hours, while our system was trained with only 186 hours. We consider scaling up the dataset in future work.

TABLE V
EVALUATION RESULTS FOR THE NUMBER OF FUTURE TOKENS AND DECODER TYPES.

Mode	RTF(s)	NMOS	QMOS	WER
GT		4.75±0.07	4.45±0.09	0.24
$m = 1$, Tokenizer	0.082	2.95±0.14	3.06±0.16	0.49
$m = 6$, flow matching	**0.161**	3.61±0.14	4.07±0.12	0.27
$m = 4$, flow matching	0.181	3.67±0.16	4.02±0.13	0.24
$m = 2$, flow matching	0.255	3.90±0.14	4.19±0.11	**0.21**
$m = 1$, flow matching	0.405	**4.10±0.13**	**4.20±0.10**	0.26

TABLE VI
THE ABLATION RESULTS FOR EFFECTS OF CFG.

	WER	CER	#SUB	#INS+DEL
GT	0.24	0.08	50	5
ULF-TTS	0.21	0.08	44	8
-CFG	0.24	0.09	47	8

D. Impact on Multi-token prediction and Decoder types

We conducted internal experiments on multi-token prediction and different decoder types. The quality MOS (QMOS) uses the same scale as NMOS. Ten native speakers evaluated 20 expressive sentences derived from a Korean open dataset. Note that RTF measures the time required to generate the speech waveform, unlike Table III. Regarding multi-token prediction, Table V represents that RTF and NMOS exhibit an inverse relationship with respect to the number of future tokens, m. This means that our system supports the optimal balance between speech quality and inference speed depending on the service environment. The QMOS results show that the flow matching decoder outperforms the tokenizer by a substantial margin of 1. We demonstrate that flow matching enhances speech quality. The noise present in the mel-spectrogram generated by the tokenizer adversely degrades pronunciation clarity, contributing to the lower NMOS score, as evident by the low WER score.

E. Effect on classifier-free guidance (CFG)

We also conducted an internal experiment to evaluate the effect of classifier-free guidance (CFG) on the language model. The evaluation set consists of 20 expressive samples, identical to those used in Sec IV-D. To evaluate token prediction performance, we measured intelligibility as represented in Table VI. The results demonstrate that applying CFG to the language model enhances intelligibility, indicating that CFG improves token prediction not only in natural language domains but also in speech generation.

V. CONCLUSION

This paper proposes an uncluttered hybrid TTS system. We propose a fully MLP module as the flow matching decoder module and support flexible token prediction time by modifying the token prediction strategy of the language model. To enhance generative performance, we utilize the CFG training strategy employed in diffusion in language model training. Through objective and subjective evaluations, we demonstrate that our hybrid system not only generates more natural speech based on a given prompt but also achieves greater efficiency compared to other hybrid systems.

ACKNOWLEDGMENT

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2025-25441313, Professional AI Talent Development Program for Multimodal AI Agents).

REFERENCES

[1] C. Wang, S. Chen, Y. Wu, *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," *arXiv preprint arXiv:2301.02111*, 2023.

[2] Z. Borsos, R. Marinier, D. Vincent, *et al.*, "Audiolm: A language modeling approach to audio generation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 31, pp. 2523–2533, 2023.

[3] P. Anastassiou, J. Chen, J. Chen, *et al.*, "Seed-tts: A family of high-quality versatile speech generation models," *arXiv preprint arXiv:2406.02430*, 2024.

[4] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[5] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[6] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, "High-fidelity audio compression with improved rvqgan," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[7] M. Le, A. Vyas, B. Shi, *et al.*, "Voicebox: Text-guided multilingual universal speech generation at scale," *Advances in neural information processing systems*, vol. 36, 2024.

[8] S. E. Eskimez, X. Wang, M. Thakker, *et al.*, "E2 tts: Embarrassingly easy fully non-autoregressive zero-shot tts," *arXiv preprint arXiv:2406.18009*, 2024.

[9] S. Mehta, R. Tu, J. Beskow, É. Székely, and G. E. Henter, "Matcha-tts: A fast tts architecture with conditional flow matching," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2024, pp. 11 341–11 345.

[10] Y. Chen, Z. Niu, Z. Ma, *et al.*, "F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching," *arXiv preprint arXiv:2410.06885*, 2024.

[11] J. Betker, "Better speech synthesis through scaling," *arXiv preprint arXiv:2305.07243*, 2023.

[12] Z. Du, Q. Chen, S. Zhang, *et al.*, "Cosyvoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens," *arXiv preprint arXiv:2407.05407*, 2024.

[13] H.-H. Guo, K. Liu, F.-Y. Shen, *et al.*, "Fireredtts: A foundation text-to-speech framework for industry-level generative speech applications," *arXiv preprint arXiv:2409.03283*, 2024.

[14] H. Guo, F. Xie, K. Xie, *et al.*, "Socodec: A semantic-ordered multi-stream speech codec for efficient language model based text-to-speech synthesis," *arXiv preprint arXiv:2409.00933*, 2024.

[15] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in neural information processing systems*, vol. 33, pp. 17 022–17 033, 2020.

[16] L. Meng, L. Zhou, S. Liu, *et al.*, "Autoregressive speech synthesis without vector quantization," *arXiv preprint arXiv:2407.08551*, 2024.

[17] H. Tang, Y. Wu, S. Yang, *et al.*, "Hart: Efficient visual generation with hybrid autoregressive transformer," *arXiv preprint arXiv:2410.10812*, 2024.

[18] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve, "Better & faster large language models via multi-token prediction," *Proceedings of the 41st International Conference on Machine Learning*, 2025.

[19] P. Esser, S. Kulal, A. Blattmann, *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first International Conference on Machine Learning*, 2024.

[20] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 29, pp. 3451–3460, 2021.

[21] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen, "Finite scalar quantization: Vq-vae made simple," *arXiv preprint arXiv:2309.15505*, 2023.

[22] J. Chung, J. Huh, S. Mun, *et al.*, "In defence of metric learning for speaker recognition," *arXiv preprint arXiv:2003.11982*, 2020.

[23] J. Nilsson and T. Akenine-Möller, "Understanding ssim," *arXiv preprint arXiv:2006.13846*, 2020.

[24] H. Touvron, L. Martin, K. Stone, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[25] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127 063, 2024.

[26] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[28] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[29] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[30] P. Sun, Y. Jiang, S. Chen, *et al.*, "Autoregressive model beats diffusion: Llama for scalable image generation," *arXiv preprint arXiv:2406.06525*, 2024.

[31] J. Yang, J. Lee, H.-S. Choi, S. Ji, H. Kim, and J. Lee, "Dualspeech: Enhancing speaker-fidelity and text-intelligibility through dual classifier-free guidance," in *Proc. Interspeech 2024*, 2024, pp. 4423–4427.

[32] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations*, 2023.

[33] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-tts: A diffusion probabilistic model for text-to-speech," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8599–8608.

[34] Y. Koizumi, H. Zen, S. Karita, *et al.*, "Libritts-r: A restored multi-speaker text-to-speech corpus," *arXiv preprint arXiv:2305.18802*, 2023.

[35] K. Ito and L. Johnson, *The lj speech dataset*, https://keithito.com/LJ-Speech-Dataset/, 2017.

[36] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2017.

[37] T. Bak, J. Lee, H. Bae, J. Yang, J.-S. Bae, and Y.-S. Joo, "Avocodo: Generative adversarial network for artifact-free vocoder," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 12 562–12 570.