JATTS: A Comparison-oriented Japanese Text-to-speech Open-sourced Toolkit

Wen-Chin HUANG†, Lester VIOLETA†, and Tomoki TODA†

† Nagoya University

E-mail: †wen.chinhuang@g.sp.m.is.nagoya-u.ac.jp

Abstract This paper presents JATTS, our initiative on building an open-source toolkit that implements a comprehensive set of representative, modern-day text-to-speech (TTS) methods, as well as the benchmark results using Japanese datasets. We analyze how different design choices affect synthesis quality, including alignment strategies, model architectures, and training objectives. We also explore the prevalent, in-context learning-based approaches towards large-scale TTS, and discuss practical challenges in training and evaluation. Our findings provide insights into building more expressive and robust Japanese TTS systems and highlight the need for better datasets and benchmarks for future research.

Key words text-to-speech, open-source

1. Introduction

Text-to-speech (TTS) refers to the task of generating natural speech from a given input text, optionally with zero or more conditions, such as speaker, emotion, etc. In recent years, the success of deep learning has powered the rapid development of TTS technology, making it possible to achieve human parity in terms of naturalness in certain settings [1]. However, in the literature, state-of-the-art TTS models have been mainly developed and benchmarked on high-resource languages such as English and Chinese. Japanese, on the other hand, remains underrepresented in recent open-source TTS research. Japanese presents unique challenges in terms of TTS, such as pitch accent, mora-timing, and context-sensitive pronunciation, making it a valuable target for evaluating the generalization ability of modern TTS techniques. This work aims to accelerate research on these advanced TTS models in the context of Japanese.

In the modern era of machine learning research, there are two components that are essential to promoting and facilitating development in a specific field, with the most important element being the existence of datasets. The Japanese speech processing community has been constantly building speech corpora, with the earliest attempts dating back to the development of the ATR [2] and CSJ [3] corpora. Up to now, there exist plenty of datasets with diverse characteristics, including reading-style datasets like JSUT [4] and JVS [5], conversational-style single-speaker datasets like Hi-Fi-Captain [6], audiobook datasets like J-MAC [7], non-verbal datasets like JVNV [8], empathetic datasets like STUDIES [9], voice conversion-oriented datasets like SRC4VC [10], and description-rich datasets like COCO-NUT [11]. The above-mentioned datasets are limited in scale, with the total durations ranging from tens to hundreds of hours. As modern-day TTS models can scale up to 100k hours of training data [12, 13], some have attempted to curate large-scale Japanese speech datasets using web-based data, including JTubeSpeech [14], YODAS [15], and J-Chat [16]. However, there lacks a systematic benchmark comparison on the effectiveness of using these datasets in TTS scenarios.

In addition, the development of open-source toolkits also plays an important role. It has gradually become a de facto standard for scientific papers to provide training recipes for interested readers to reproduce the results reported in the paper, and TTS is no exception. However, such recipes are often restricted to the method implemented and the dataset used in the paper. Other researchers have also tried to build TTS toolkits to support benchmarking on multiple datasets with multiple model options, from early attempts like HTS [17] and Merlin [18] to recent toolkits like ESPnet-TTS [19,20], Coqui-TTS [21], and Amphion [22]. However, the most recent major releases in ESPnet-TTS and Coqui-TTS date back to 2022, and there is a need to reflect the latest advances in TTS research.

This work presents JATTS(1), our initiative to build an open-source toolkit for Japanese TTS, with a special focus on comparison and benchmarking. JATTS inherits the design of ESPnet-TTS, providing all-in-one training and evaluation scripts for conducting TTS experiments on a collection of Japanese corpora. We present our latest experimental results of three highly cited TTS models, namely FastSpeech2 [23], VITS [24] and Matcha-TTS [25], on three commonly used Japanese speech datasets: JSUT, Hi-Fi-Captain, and JVS. The results provide valuable insights into the cause of the performance difference between the three models. In addition, we report our ongoing effort to train VALL-E [26], a representative approach that resembles the latest trend of large language model (LLM)-based TTS.

2. Features in JATTS

2.1 Toolkit design

JATTS closely follows the design of ESPnet-TTS [19], where in such a design, for each training dataset, a collection of bash or python scripts called *recipe* is prepared, which provides all-in-one instructions to complete data preprocessing, model training, inference, and evaluation. Users can choose from many configuration files, each containing a set of hyperparameters to perform training of a specific model. The core part of the toolkit is the *library*, which implements the model architectures, loss computation, and training logic. In JATTS, the main deep learning framework was PyTorch.

2.2 Datasets

Table 1 lists the training datasets currently supported in JATTS.

(1): https://github.com/unilight/jatts

Table 1 Statistics of the supported training datasets. fs stands for sampling frequency.

Name	# Speakers	# Samples	Duration (hr)	fs (kHz)
JSUT	1	4500	6.0	24
HFC female	1	18856	20.2	48
JVS	86	9976	16.6	24
CSJ	3176	403096	515.6	16

JSUT [4] is a single-speaker dataset, containing 10 hours of read speech from a native Japanese female speaker. We used the `basic5000` set, which covered daily characters, and further used a 4500/250/250 train/dev/test split. JSUT was originally of 48 kHz, and we downsampled it to 24 kHz. Hi-Fi-Captain (HFC) [6] is also a single-speaker dataset, but in conversational style, and we used samples from the Japanese female speaker. We used the official dev and test sets, which both contain 250 utterances, and combined the `train_non_parallel` and `train_parallel` subsets as the training set. JVS [5] is a multi-speaker read style dataset, and we split the original 100 speakers into 86/4/10 for train/dev/test. We designed the test set such that we have the ground-truth for all ten utterances for the ten test speakers, resulting in 100 testing utterances. We also made sure there is no overlap in text between the train, dev, and test sets. The CSJ dataset [3] is a spontaneous speech dataset containing academic presentation speech and simulated public speech. Since CSJ is not of studio quality like the three datasets mentioned above, we only used CSJ for training the VALL-E model and did not include its official test set in JATTS.

2.3 Text preprocessing and vocoders

Using phonemes instead of raw characters (graphemes) in TTS can greatly ease model learning, especially in the case of modern-day TTS systems based on deep neural networks, and such a process is called grapheme-to-phoneme (G2P) conversion. For Japanese, G2P can be more difficult than other languages because the Japanese orthography consists of three unique writing systems: Hiragana, Katakana, and Kanji. In JATTS, we used the `pyopenjtalk.g2p`[(2)] function to perform G2P. While it is possible to augment the phoneme sequence with accent and prosody labels using the `pyopenjtalk.extract_fullcontext` function, we leave it as future work.

The complex nature of the speech waveform makes it challenging to directly model TTS in a fully end-to-end manner, and for the time being, the mel-spectrogram is still the most common intermediate representation. Most TTS systems can therefore be broken down into two components: a text-to-mel module (also called the acoustic model), and a mel-to-waveform module (also called the vocoder). Although there have been recent attempts towards fully end-to-end models like VITS, in this work, we still use mel-spectrogram for (1) better comparison of different acoustic models, and (2) faster experiment cycles, as end-to-end models typically take longer to train. In this work, we mainly use the HiFi-GAN vocoder [27].

2.4 Supported model category 1: non-autoregressive models

2.4.1 Overview

Non-autoregressive (NAR) TTS models have largely superseded the earliest pioneering works of the neural TTS wave, notably the Tacotron family [28, 29], which relied on autoregressive (AR) modeling. AR models suffer from slow inference and error accumulation due to their inherently sequential generation process. In contrast, NAR models address these limitations by decoupling duration prediction from acoustic feature generation, enabling parallel computation and improved robustness.

(2): https://github.com/r9y9/pyopenjtalk

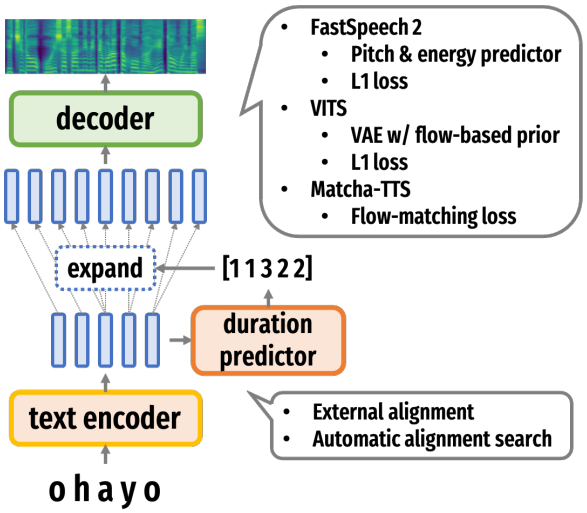

Fig. 1 Illustration of the inference process of the non-autoregressive models supported in JATTS currently. The mel-to-waveform module is omitted.

JATTS currently supports three representative NAR TTS models, namely FastSpeech 2, VITS, and Matcha-TTS. These three models share a common inference process, which is illustrated in Figure 1. The text encoder first takes the text as input to generate a sequence of encoder representations, which is then used by the duration predictor to generate a sequence of predicted durations. Then, an "expand" operation, also known as "length regulation" in some literature, adjusts the length of the encoder representation sequence given the predicted durations. Finally, the decoder synthesizes the mel-spectrogram from the expanded encoder representations. and a vocoder generates the final waveform.

For multi-speaker modeling, we simply use a pre-trained speaker encoder and broadcast the speaker embedding to concatenate it with the encoder representations. In JATTS, the ECAPA-TDNN model [30] from SpeechBrain[(3)] was used. Readers should note that all three models were benchmarked in the single-speaker TTS setting in their respective original papers, and a better speaker conditioning for these models is yet to be explored in the future.

2.4.2 Duration modeling

In NAR TTS, there are typically two ways to perform duration modeling. The first method is to use an **external module** to generate the "ground truth" durations for each input text. In practice, a forced aligner is often used, which takes a speech sample and its transcription as input, and outputs the duration of each text in the transcription. During the training of the TTS model, the duration predictor is directly optimized against the ground truth durations, and the expand operation directly uses the ground truth durations. In JATTS, the Julius forced aligner[(4)] is used [31].

The second approach is referred to as **implicit alignment**, where the alignment is learned automatically during model training. This method was first introduced in [32], and JATTS uses an enhanced version proposed in [33]. It applies monotonicity constraints as a strong inductive bias and uses the Viterbi algorithm with diagonal constraints to find the optimal alignment path. This path can be viewed as a hard alignment matrix, from which the duration sequence is derived. Readers are referred to the original work [33] and the JATTS implementation for further details.

(3): https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb
(4): https://github.com/julius-speech/segmentation-kit

Generally speaking, models that employ the implicit alignment method are harder to train because the alignment and acoustic model need to be learned simultaneously, making optimization less stable. Meanwhile, the model is also equipped with greater flexibility, allowing it to capture richer prosodic variation, particularly when large amounts of data are available. This creates a trade-off: on smaller datasets, external alignment tends to yield better performance, whereas on larger datasets, implicit alignment becomes advantageous.

2.4.3 Detailed descriptions of each model

The reason why the three models are chosen is that they all introduce their own inductive biases or probabilistic modeling strategies to tackle the one-to-many nature in TTS. In TTS, a given input sequence can correspond to multiple realizations, varying in prosody, pitch, and speaking style. It is therefore essential to Below, we briefly describe each model, and the modifications we made in the implementation in JATTS.

FastSpeech2 [23]. FastSpeech 2 employs a pitch and energy predictor module to explicitly predict prosodic features. The training loss is a simple L1 loss on the mel spectrogram generation, as well as the duration predictor loss. We used WORLD [34] to extract pitch, and the Conformer [35] architecture was used for both the text encoder and decoder. We also used the token-averaged pitch and energy as in ESPnet2-TTS. As a result, automatic alignment search is not applicable to FastSpeech2.

VITS [24]. VITS employs a variational autoencoder (VAE) structure with a prior enhanced with flow, which provides larger model latent variability and enhances expressiveness. The original VITS further used a generative adversarial network (GAN) objective to directly model the speech waveform, with a cost of increased training time and longer experiment cycles. Therefore, the VITS in the current JATTS outputs mel-spectrograms, such that the loss function consists of an L1 loss on the mel-spectrogram, a KL divergence loss, and the duration predictor loss.

Matcha-TTS [25]. Matcha-TTS is a representative TTS model based on conditional flow-matching (CFM) [36], a generative modeling framework that is often considered more efficient than diffusion-based (or score-based) models [37] in terms of inference speed. When applied to TTS, probabilistic models like CFM are expected to more accurately approximate the true data distribution compared to models trained solely with L1 loss (such as FastSpeech 2), offering increased modeling flexibility and expressiveness. The JATTS implementation largely followed the official Matcha-TTS codebase, except that we employed the automatic alignment search method proposed in [33] instead of the mechanism from [32] used in the original paper.

2.5 Supported model category 2: In-context learning, prompt-based models

Parallel to the direction of increasing compuational efficiency using techniques like NAR modeling, another direction that has been drawing attention is how to scale up TTS training in terms of data and model size. Inspired by the recent success of LLMs in the natural language processing field, VALL-E was one of the pioneering works to scale up TTS training by casting TTS as a next-token prediction task. Notably, these LLM-based models demonstrate the so-called "in-context learning" (ICL) ability [38], which, in the context of TTS, refers to generating speech with the given text and desired information such as speaker identity, emotion, or even recording environment, with a short reference sample called the "prompt".

The core idea of LLM-based TTS is to model speech in a discrete representation space. In VALL-E, this was done by using EnCodec [39], a neural audio codec model, to encode a speech sample $\boldsymbol{x}$ into a *speech token* matrix of shape $\boldsymbol{C} \in \{1, \ldots, K\}^{T \times 8}$, where K is the quantizer codebook size, and T is the length. Using the residual vector quantization technique, the EnCodec model has a hierarchical architecture, thus for each time step the speech token has 8 entries. It is thereafter straightforward to model the input text and the speech tokens, which are both in the discrete space, using one unified LLM.

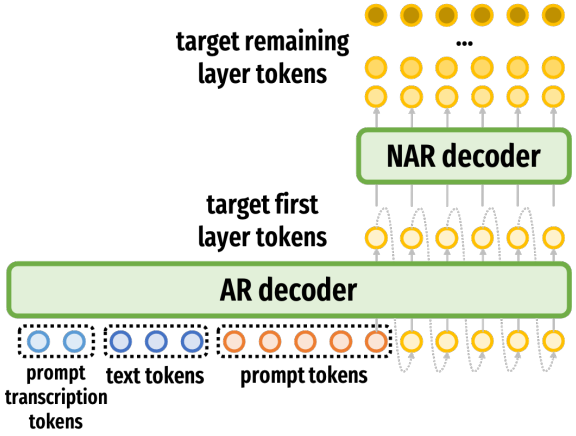

Fig. 2 Illustration of the inference process of the VALL-E model.

Figure 2 illustrates the inference process of VALl-E. To further take advantage of the hierarchical structure of EnCodec, an AR decoder, which is an usual LLM, first generates the tokens in the first layer (i.e., $\boldsymbol{C}_{t,1}$). Then, an NAR decoder generates the tokens in the rest of the layers (i.e., $\boldsymbol{C}_{t,2:8}$). The intuition is that shallow layers control global properties like speaker identity and prosody, while deep layers refine the acoustic details.

The ICL ability of VALL-E primarily stems from the AR decoder, leveraging the inherent tendency of LLMs to perform continuation. Looking at the inference design shown in Figure 2, the AR decoder takes as input a concatenation of: (1) text tokens from the prompt, (2) text tokens to synthesize, and (3) EnCodec speech tokens from the prompt. From the model's perspective, it is trained to predict the speech tokens corresponding to the entire input text. However, because the speech tokens of the prompt are already provided, they are treated as "already synthesized," and the model proceeds to generate the speech tokens for the remaining input text. This behavior can also be viewed as speech continuation. Moreover, because the prompt's speech tokens encode acoustic characteristics such as speaker identity and emotion, the decoder naturally conditions on these properties when generating new speech tokens, allowing it to produce speech that matches the style of the prompt. VALL-E thus obtains its zero-shot generation ability.

In the original paper, the training procedure was not entirely clear. In our opinion, it is essential to mirror the speech continuation scenario during training. Thus, in our implementation, given a training sample consisting of the text tokens and speech tokens, the input to the AR decoder is a concatenation of (1) the full text tokens, and (2) the speech tokens of the first 3 seconds. The model is then trained to predict the remaining speech tokens, thus learns to perform continuation which is consistent with its inference-time usage.

2.6 Objective evaluation metrics

Although subjective tests are the gold standard for TTS, objective evaluation metrics can serve as guidance to the goodness of the current model and hyperparameter set during development, and can represent the actual quality when properly chosen. In JATTS, the following metrics can be calculated:

- The **Mel cepstral distortion (MCD)** is a general objective metric in TTS that well correlates with human perception in a

Table 2 Single-speaker TTS results. Boldface indicates the best result in the same train/test set.

ID	Train/test set	Model	Alignment	MCD↓	F0RMSE↓	F0CORR↑	DDUR↓	SHEET↑	CER↓
1	JSUT/JSUT	FastSpeech 2	External	**7.28**	49.510	0.571	0.705	4.072	**2.6**
2		Matcha-TTS	External	7.72	50.428	0.580	0.745	4.098	3.2
3		Matcha-TTS	Implicit	7.90	51.556	0.565	0.345	**4.189**	4.6
4		VITS	Implicit	7.82	**49.008**	**0.595**	**0.255**	4.019	22.1
5	HFC/HFC	FastSpeech 2	External	**5.65**	48.020	0.682	0.645	4.175	1.8
6		Matcha-TTS	External	6.12	50.840	0.658	0.686	4.169	2.1
7		Matcha-TTS	Implicit	6.13	50.182	0.655	0.163	**4.236**	**1.5**
8		VITS	Implicit	5.99	**46.123**	**0.709**	**0.144**	4.129	10.3

homogeneous setting. The lower the distortion, the better the performance.

- The **F0 root mean square error (F0RMSE)** and **F0 linear correlation (F0CORR)** assess the ability to properly model prosody. For F0RMSE, the lower the better, and for F0CORR, the higher the better. The f0 extractor in WORLD is again used here.
- The **duration difference (DDUR)** assess the duration modeling. The lower the difference, the better the performance.
- The **character error rates (CER)** from an off-the-shelf ASR model assess speech intelligibility. The lower the CER, the better the intelligibility. In JATTS, an open-source model is used[5].
- The **SHEET** score is from a perceptual rating predictor[6] trained with human ratings. The higher the score, the higher the performance.
- The **speaker embedding cosine similarity (SIM)** is used only in multi-speaker training, and it measures whether the speaker identity is properly modeled. The higher the cosine similarity, the better the speaker identity is modeled. Here, the x-vector described in Section 2.4.1 is used.

3. Experimental results

In this section, we present our current experimental results, which are broken down into single-speaker and multi-speaker settings. Audio samples can be found online[7].

3.1 Single-speaker TTS

Table 2 shows the single-speaker TTS results on the JSUT and HFC datasets. First, FastSpeech 2 achieved the best MCD scores on both datasets, and our manual inspection found that its samples were consistently more stable, though somewhat lacking in expressiveness. Next, we observed that Matcha-TTS with external alignment performed worse than FastSpeech 2 across most metrics (see rows 1 vs. 2, and 5 vs. 6). However, when using implicit alignment – as in the original Matcha-TTS paper – the model showed notable improvements in DDUR and SHEET scores (rows 2 vs. 3, and 6 vs. 7), and on HFC, using implicit alignment even led to a lower CER (rows 6 and 7). These results suggest that switching from an L1 loss to a CFM-based objective alone is not sufficient to improve performance: the benefit emerges only when combined with implicit alignment. Moreover, the gains from implicit alignment are more pronounced on the larger HFC dataset (20 hours) compared to the smaller JSUT dataset (6 hours). Finally, VITS showed the best performance in prosody-related metrics, including F0RMSE, F0CORR, and DDUR, highlighting its ability to generate more expressive speech. However, this came at the cost of significantly reduced intelligibility, a drawback that appears to diminish as dataset size increases (rows 4 and 8).

3.2 Multi-speaker TTS

We first look at the multi-speaker TTS results using the JVS training set, as shown in Table 3. We found that the observations in the single-speaker TTS experiments do not hold in the multi-speaker setting. Notably, for Matcha-TTS, switching from explicit alignment to implicit alignment led to worse scores in most metrics. As for VITS, our current implementation fails to generate any intelligible sample. By manually inspecting the samples, we found that FastSpeech 2 is currently the best-performing model on JVS.

Following our discussion in Section 2.4.2, we suspect that the JVS dataset is too limited to fully exploit the potential of expressive, probabilistic models like Matcha-TTS and VITS, especially when using implicit alignment. First, the scale of JVS (16.6 hours, 86 speakers) is much smaller than that of typical multi-speaker TTS datasets, such as LibriTTS (585 hours, 2,456 speakers) [40]. Additionally, JVS is known for its restricted linguistic diversity, as there are many parallel sentences across speakers. These limitations highlight the need for a larger and more linguistically rich Japanese multi-speaker TTS dataset.

Finally, row 13 in Table 3 shows our current VALL-E results. While the evaluation metrics appear discouraging, manual inspection reveals that the generated samples are promising and show potential. of the samples shows positive results. The primary limitation lies in the scale of data and computational resources. The original VALL-E model was trained on LibriLight [41], a 60k-hour dataset, using 16 NVIDIA TESLA V100 32GB GPUs with a batch size of 6k tokens per GPU for 800k steps. In contrast, our current setup used The CSJ dataset, which is rougly 100x smaller, and due to the limited budget and on-demand compute in the academics, was trained on only 4 V100 GPUs. This not only significantly slowed down the training process but also constrained the total number of training steps. With larger-scale data and more computational resources, we expect that our implementation could achieve comparable performance to that of the original.

4. Conclusions and Future Work

In this paper, we introduced JATTS, our initiative to build an open-source Japanese TTS toolkit that supports a range of modern models, including NAR architectures such as FastSpeech 2, Matcha-TTS, and VITS, as well as in-context learning-based approaches like VALL-E. JATTS also supports multiple datasets, from single-speaker corpora like JSUT and HFC to multi-speaker corpora like JVS and CSJ. Experimental results revealed the trade-off between different modeling choices, including external versus implicit alignment and model expressivity versus intelligibility. We

(5): https://huggingface.co/rinna/nue-asr
(6): https://github.com/unilight/sheet
(7): https://unilight.github.io/Publication-Demos/publications/jatts/index.html

Table 3 Multi-speaker TTS results. Boldface indicates the best result in the same train/test set.

ID	Train/test set	Model	Alignment	MCD↓	F0RMSE↓	F0CORR↑	DDUR↓	SHEET↑	CER↓	SIM↑
9	JVS/JVS	FastSpeech 2	External	**8.06**	44.935	**0.655**	1.081	4.491	**3.0**	**0.709**
10		Matcha-TTS	External	8.62	**44.190**	0.638	1.408	**4.536**	3.5	0.701
11		Matcha-TTS	Implicit	8.96	44.763	0.634	**1.039**	4.422	11.1	0.687
12		VITS	Implicit	(Fail)						
13	CSJ/JVS	VALL-E	–	13.05	62.097	0.205	1.413	2.611	64.8	0.577

also confirmed the importance of matching model complexity with the dataset size. Finally, we suspect that there exists a limitation of current Japanese multi-speaker corpora for training advanced models.

Looking ahead, we plan to enrich the features in JATTS along the following directions:

Better vocoders. The current setup in JATTS trained HiFi-GAN vocoders for each dataset, while recent advanced vocoders such as BigVGAN [42] are trained on large-scale, multi-lingual datasets. It is worthwhile to use such a kind of universal vocoder to further eliminate the influence of the vocoder to better focus on the comparison of other components.

Other in-context learning, prompt-based models. The VALL-E model implemented in JATTS is just the very first work that led to emergent of the family of in-context learning and prompt-driven TTS models. Subsequent works include those based on flow-matching with an audio-infilling objective like E2-TTS [43], and models that mix both, like MaskGCT [44]. We plan to include these models in the future.

Improved frontend preprocessing. As discussed in Section 2.3, the unique characteristics of the Japanese writing system highlights the importance of text preprocessing in Japanese TTS. In addition to exploring the influence of accent and pitch labels in different models, we also plan to incorporate recent neural frontends like Japanese PnG BERT [45] and BETT [46].

Diverse and challenging test sets. As most popular TTS training datasets consist of short, neutral, read-style speech samples, there has been an increasing need to benchmark modern-day TTS models on harder and more diverse conditions. Looking at the current available resources, we plan to include emotional speech [8], conversational and long-form dialogue [47], and dialect speech [48]. Such benchmarks are critical for developing TTS systems that go beyond standard read speech.

Overall, we hope that this toolkit and study can serve as a foundation for future research in Japanese TTS and contribute to the growing open-source ecosystem.

Acknowledgements This work was partly supported by JST AIP Acceleration Research JPMJCR25U5, Japan.

References

[1] X. Tan, Neural text-to-speech synthesis, Springer, 2023.

[2] A. Kurematsu, K. Takeda, Y. Sagisaka, S. Katagiri, H. Kuwabara, and K. Shikano, "ATR Japanese speech database as a tool of speech recognition and synthesis," Speech communication, vol.9, no.4, pp.357–363, 1990.

[3] K. Maekawa, "Corpus of spontaneous Japanese: its design and evaluation," Proc. ISCA/IEEE Workshop on Spontaneous Speech Processing and Recognition, 2003.

[4] R. Sonobe, S. Takamichi, and H. Saruwatari, "JSUT corpus: free large-scale Japanese speech corpus for end-to-end speech synthesis," arXiv preprint arXiv:1711.00354, 2017.

[5] S. Takamichi, K. Mitsui, Y. Saito, T. Koriyama, N. Tanji, and H. Saruwatari, "JVS corpus: free Japanese multi-speaker voice corpus," arXiv preprint arXiv:1908.06248, 2019.

[6] T. Okamoto, Y. Shiga, and H. Kawai, "Hi-Fi-CAPTAIN: High-fidelity and high-capacity conversational speech synthesis corpus developed by NICT." https://ast-astrec.nict.go.jp/en/release/hi-fi-captain/, 2023.

[7] S. Takamichi, W. Nakata, N. Tanji, and H. Saruwatari, "J-MAC: Japanese multi-speaker audiobook corpus for speech synthesis," arXiv preprint arXiv:2201.10896, 2022.

[8] D. Xin, J. Jiang, S. Takamichi, Y. Saito, A. Aizawa, and H. Saruwatari, "JVNV: A Corpus of Japanese Emotional Speech With Verbal Content and Nonverbal Expressions," IEEE Access, vol.12, pp.19752–19764, 2024.

[9] Y. Saito, Y. Nishimura, S. Takamichi, K. Tachibana, and H. Saruwatari, "STUDIES: Corpus of Japanese Empathetic Dialogue Speech Towards Friendly Voice Agent," Proc. Interspeech, pp.5155–5159, 2022.

[10] Y. Saito, T. Igarashi, K. Seki, S. Takamichi, R. Yamamoto, K. Tachibana, and H. Saruwatari, "SRC4VC: Smartphone-Recorded Corpus for Voice Conversion Benchmark," Proc. Interspeech, pp.1825–1829, 2024.

[11] A. Watanabe, S. Takamichi, Y. Saito, W. Nakata, D. Xin, and H. Saruwatari, "COCO-NUT: Corpus of Japanese Utterance and Voice Characteristics Description for Prompt-Based Control," Proc. ASRU, 2023.

[12] M. Lajszczak, G. Cámbara, Y. Li, F. Beyhan, A. van Korlaar, F. Yang, A. Joly, Á. Martín-Cortinas, A. Abbas, A. Michalski, *et al.*, "Base tts: Lessons from building a billion-parameter text-to-speech model on 100k hours of data," arXiv preprint arXiv:2402.08093, 2024.

[13] H.H. Guo, Y. Hu, K. Liu, F.Y. Shen, X. Tang, Y.C. Wu, F.L. Xie, K. Xie, and K.T. Xu, "Fireredtts: A foundation text-to-speech framework for industry-level generative speech applications," arXiv preprint arXiv:2409.03283, 2024.

[14] S. Takamichi, L. Kürzinger, T. Saeki, S. Shiota, and S. Watanabe, "JTubeSpeech: corpus of Japanese speech collected from YouTube for speech recognition and speaker verification," arXiv preprint arXiv:2112.09323, 2021.

[15] X. Li, S. Takamichi, T. Saeki, W. Chen, S. Shiota, and S. Watanabe, "YODAS: Youtube-Oriented Dataset for Audio and Speech," Proc. ASRU, 2023.

[16] W. Nakata, K. Seki, H. Yanaka, Y. Saito, S. Takamichi, and H. Saruwatari, "J-CHAT: Japanese Large-scale Spoken Dialogue Corpus for Spoken Dialogue Language Modeling," arXiv preprint arXiv:2407.15828, 2024.

[17] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A.W. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," Proc. 6th ISCA Workshop on Speech Synthesis (SSW 6), pp.294–299, 2007.

[18] Z. Wu, O. Watts, and S. King, "Merlin: An Open Source Neural Network Speech Synthesis System," Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9), pp.202–207, 2016.

[19] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, "Espnet-TTS: Unified, Reproducible, and Integratable Open Source End-to-End Text-to-Speech Toolkit," Proc. ICASSP, pp.7654–7658, 2020.

[20] T. Hayashi, R. Yamamoto, T. Yoshimura, P. Wu, J. Shi, T. Saeki, Y. Ju, Y. Yasuda, S. Takamichi, and S. Watanabe, "ESPnet2-TTS: Extending the edge of TTS research," arXiv preprint arXiv:2110.07840, 2021.

[21] G. Eren and The Coqui TTS Team, "Coqui TTS," Jan. 2021.
[22] X. Zhang, L. Xue, Y. Gu, Y. Wang, J. Li, H. He, C. Wang, T. Song, X. Chen, Z. Fang, H. Chen, J. Zhang, T.Y. Tang, L. Zou, M. Wang, J. Han, K. Chen, H. Li, and Z. Wu, "Amphion: An Open-Source Audio, Music and Speech Generation Toolkit," Proc. SLT, 2024.
[23] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.Y. Liu, "FastSpeech 2: Fast and High-Quality End-to-End Text to Speech," Proc. ICLR, 2021.
[24] J. Kim, J. Kong, and J. Son, "Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech," Proc. ICML, pp.5530–5540, 2021.
[25] S. Mehta, R. Tu, J. Beskow, É. Székely, and G.E. Henter, "Matcha-TTS: A fast TTS architecture with conditional flow matching," Proc. ICASSP, 2024.
[26] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," arXiv preprint arXiv:2301.02111, 2023.
[27] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis," Proc. NeurIPS, pp.17022–17033, 2020.
[28] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R.J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R.A. Saurous, "Tacotron: Towards end-to-end speech synthesis," Proc. Interspeech, pp.4006–4010, 2017.
[29] J. Shen, R. Pang, R.J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R.A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning WaveNet on MEL Spectrogram Predictions," Proc. ICASSP, pp.4779–4783, 2018.
[30] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," Proc. Interspeech, pp.3830–3834, 2020.
[31] A. Lee, T. Kawahara, *et al.*, "Recent development of open-source speech recognition engine julius," Proc. APSIPA ASC, pp.131–137, 2009.
[32] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search," Proc. NeruIPS, pp.8067–8077, 2020.
[33] K.J. Shih, R. Valle, R. Badlani, A. Lancucki, W. Ping, and B. Catanzaro, "RAD-TTS: Parallel flow-based TTS with robust alignment learning and diverse synthesis," Proc. ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, 2021. 8 pages.
[34] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," IEICE Transactions on Information and Systems, vol.99, no.7, pp.1877–1884, 2016.
[35] A. Gulati, J. Qin, C.C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," Proc. Interspeech, pp.5036–5040, 2020.
[36] Y. Lipman, R.T.Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow Matching for Generative Modeling," Proc. ICLR, 2023.
[37] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," Proc. ICLR, 2021.
[38] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," Proc. NeurIPS, pp.1877–1901, 2020.
[39] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," TMLR, 2023.
[40] H. Zen, V. Dang, R. Clark, Y. Zhang, R.J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech," Proc. Interspeech, pp.1526–1530, 2019.
[41] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, *et al.*, "Libri-light: A benchmark for asr with limited or no supervision," Proc. ICASSP, pp.7669–7673, 2020.
[42] S. gil Lee, W. Ping, B. Ginsburg, B. Catanzaro, and S. Yoon, "BigVGAN: A Universal Neural Vocoder with Large-Scale Training," Proc. ICLR, 2023.
[43] S.E. Eskimez, X. Wang, M. Thakker, C. Li, C.H. Tsai, Z. Xiao, H. Yang, Z. Zhu, M. Tang, X. Tan, Y. Liu, S. Zhao, and N. Kanda, "E2 TTS: Embarrassingly Easy Fully Non-Autoregressive Zero-Shot TTS," Proc. SLT, pp.682–689, 2024.
[44] Y. Wang, H. Zhan, L. Liu, R. Zeng, H. Guo, J. Zheng, Q. Zhang, X. Zhang, S. Zhang, and Z. Wu, "MaskGCT: Zero-Shot Text-to-Speech with Masked Generative Codec Transformer," Proc. ICLR, 2025.
[45] Y. Yasuda and T. Toda, "Investigation of Japanese PnG BERT Language Model in Text-to-Speech Synthesis for Pitch Accent Language," IEEE Journal of Selected Topics in Signal Processing, vol.16, no.6, pp.1319–1328, 2022.
[46] T. Ogura, T. Okamoto, Y. Ohtani, E. Cooper, T. Toda, and H. Kawai, "Mora-Level Prosody Prediction for Text-to-Speech Using Japanese BERT Without Accentual Labels," Proc. ICASSP, 2025.
[47] Y. Nishimura, T. Hirose, M. Ohi, H. Nakayama, and N. Inoue, "HALL-E: Hierarchical Neural Codec Language Model for Minute-Long Zero-Shot Text-to-Speech Synthesis," Proc. ICLR, 2025.
[48] S. Takamichi and H. Saruwatari, "CPJD corpus: Crowdsourced parallel speech corpus of Japanese dialects," Proc. LREC 2018, 2018.