NNSVS: A NEURAL NETWORK-BASED SINGING VOICE SYNTHESIS TOOLKIT

Ryuichi Yamamoto[1,2]*, Reo Yoneyama*[2]*, and Tomoki Toda*[2]

[1]LINE Corp., Japan, [2]Nagoya University, Japan.

ABSTRACT

This paper describes the design of NNSVS, an open-source software for neural network-based singing voice synthesis research. NNSVS is inspired by Sinsy, an open-source pioneer in singing voice synthesis research, and provides many additional features such as multi-stream models, autoregressive fundamental frequency models, and neural vocoders. Furthermore, NNSVS provides extensive documentation and numerous scripts to build complete singing voice synthesis systems. Experimental results demonstrate that our best system significantly outperforms our reproduction of Sinsy and other baseline systems. The toolkit is available at `https://github.com/nnsvs/nnsvs`.

Index Terms— singing voice synthesis, open source software, PyTorch, multi-stream models, autoregressive models

1. INTRODUCTION

Open-source software has played a crucial role in advancing research; for example, PyTorch [1] and TensorFlow [2] for deep learning, HTK [3] and Kaldi [4] for speech recognition, and HTS [5] and Merlin [6] for speech synthesis, have been extensively used by the research and industry communities.

Sinsy is an open-source pioneer in singing voice synthesis (SVS) [7]–[9]. Sinsy has had more than 10 years of development history since its first public release. Sisny adopted statistical parametric SVS based on hidden Markov models (HMMs) in the first version and switched to deep neural networks (DNNs) to improve SVS quality. Although the SVS community greatly benefited from their efforts, the functionality of its open-source version is limited to traditional HMM-based SVS and DNN-based SVS is not publicly available.

Most recently, a new open-source toolkit for end-to-end SVS, Muskits has been proposed [10]. Although Muskits provides several DNN-based SVS models [11]–[14] with a number of reproducible recipes [4], it does not support well-designed parametric approaches that can achieve both good quality and pitch robustness, as in the latest Sinsy [9].

In this paper, we propose NNSVS, a new open-source toolkit for singing voice synthesis (SVS) written in Python and PyTorch [1]. In contrast to Muskits, NNSVS does not specialize in end-to-end SVS. Instead, we aim to provide a modular and extensible codebase that is easily applied to various SVS architectures including parametric SVS [8], [12], [15], modern SVS using neural vocoders [16], [17], and their hybrid methods [9], [18]. The important features are summarized as follows:

Modular design: Following Sinsy's structure [9], NNSVS decomposes an SVS system into four core modules: the time-lag model, duration model, acoustic model, and vocoder.

Extensible design: Together with the modular design, every module can be flexibly customized. For example, users can add new acoustic model architectures without modifying the remaining modules. Furthermore, our generic multi-stream implementation of acoustic models gives users fine-grained control over the model architecture for each feature stream separately.

Language-independent design: The above-mentioned four core modules are language-independent by design. Therefore, users can create SVS systems for custom languages by implementing a language-dependent pre-processing (e.g., extracting phonetic contexts from musical scores).

Everything is open-source: In contrast to the open-source version of Sinsy, our code is fully open-sourced. We also provide an implementation that resembles Sinsy as a baseline system to further encourage reproducible research.

Complete recipes: Following the success of Kaldi [4], ESPnet [19], and Muskits [10], we provide complete setups for building SVS systems.

Documentation: NNSVS is extensively documented. Documentation is available online[1].

Even though the modular and extensible design allows users to implement their custom models and recipes, our toolkit provides several baseline implementations. In particular, we provide varieties of acoustic models such as those of Sinsy [8], [9], multi-stream models [15], and autoregressive fundamental frequency (F_0) models [18]. Furthermore, to obtain the best performance possible of traditional parametric approaches and recent neural vocoders, we incorporate the unified source-filter generative adversarial networks (uSFGAN) [20] to achieve high-quality and pitch-robust SVS systems.

To evaluate the quality of the SVS systems, we compare NNSVS with some baseline systems including Muskits [10], Sinsy [9], and a recently proposed modern SVS system called DiffSinger [17]. Experimental results demonstrate that our best system archives a mean opinion score (MOS) of 3.86, significantly outperforming the baseline systems.

2. DATA REPRESENTATION

To build SVS systems with NNSVS, the following data are required: 1) the waveform, 2) musical score, and 3) phone segmentation. The latter two can be represented as an HTS label file [5], which includes timings (i.e., the start and end time of each phone) and phonetic/musical contexts.

[1]`https://nnsvs.github.io/`

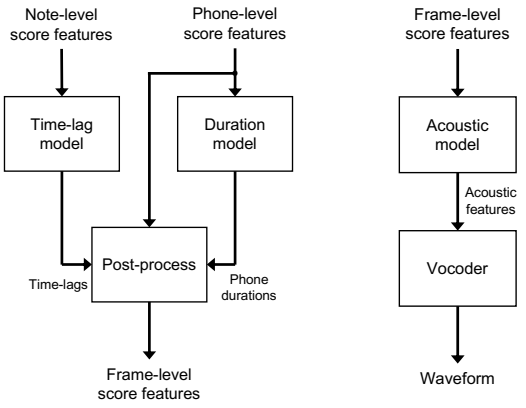

Fig. 1. Block diagram of an NNSVS SVS system: (a) phonetic timing prediction and (b) waveform synthesis.

2.1. Musical score features

Inspired by Sinsy [9], our toolkit uses HTS full-context labels as the primary musical score representation. We provide tools to obtain HTS labels from MusicXML [21] and UST[2] [24] files. We also provide a set of context definitions (i.e., phonetic and musical contexts to be extracted from HTS labels) designed for Japanese SVS. Note that they can be extended to other languages (e.g., UK, US, or Australian English [25]). Furthermore, to control the characteristics of the synthetic voice, users can add custom contexts such as falsetto flags, emotion flags, and strength of voice.

Given the context definitions, our toolkit converts HTS labels to note-level and phone-level musical score features that consist of categorical (e.g., phone identity) and numeric features (e.g., note pitch and duration). Then, those extracted features are used as inputs for the neural networks.

2.2. Acoustic features

We use WORLD [26] as the main acoustic feature extraction tool. WORLD decomposes audio signals into F_0, the spectral envelope, and band-aperiodicity (BAP). The spectral envelope is converted to mel-generalized cepstral coefficients (MGCs) to reduce dimensionality without quality deterioration [27]. Furthermore, F_0 is converted into a continuous log-scale F_0 (log-F_0) [28] and voiced/unvoiced flags (VUVs) for the ease of modeling.

The extracted acoustic features are used as the target features of the acoustic models and conditioning features for the neural vocoders. We also integrate mel-spectrogram feature extraction to support recent SVS model architectures that predict mel-spectrograms from the musical score. To support explicit vibrato modeling proposed in Sinsy [9], we provide optional vibrato parameter extraction [29].

3. CORE MODULES

Figure 1 presents an overview of an SVS system built using NNSVS. The phonetic timing prediction consists of time-lag and duration models, whereas the waveform synthesis consists of an acoustic model and a vocoder.

3.1. Time-lag model

Depending on the singing style, human vocals often deviate from the start timings of musical notes. A time-lag model predicts those note-level timing deviations given the note-level[3] musical score features [30]. We provide the same functionality as the latest Sinsy [9], which models the time-lags using mixture density networks (MDNs) [31].

3.2. Duration model

A duration model predicts phone durations given the phone-level musical score features. As in the time-lag model, we employ an MDN-based duration model [9]. By combining the predicted time-lags and phone durations, the post-processing module in Fig. 1 (a) computes normalized phone durations so that the sum of the predicted durations equals to the note durations [9]. After the post-processing, phone-level score features are converted to frame-level ones and used as the input of the acoustic models.

3.3. Acoustic model

An acoustic model predicts frame-level acoustic features from frame-level score features[4]. We describe the details of several important implementations below.

3.3.1. Sinsy-based model

We provide an implementation that resembles Sinsy's acoustic model [9]. The architecture of Sinsy's acoustic model consists of three fully connected layers, three one-dimensional convolution layers with batch normalization [32] and ReLU activations [33], followed by two bi-directional long short-term memory networks [34] and a projection layer. As the target acoustic features, vocoder parameters (e.g., MGCs and F_0) and additional vibrato parameters (i.e., binary vibrato flags, amplitude, and speed [29]) are used. For robust F_0 prediction, Sinsy adopts a pitch normalization technique: predicting the residual log-F_0 based on the input note pitch.

To avoid out-of-tune pitch issues, several pitch correction algorithms have been proposed [9]. Our toolkit provides one of Sinsy's pitch correction algorithms: a pitch correction method that imposes a prior distribution of pitch based on the note pitch information.

3.3.2. Multi-stream models

DNN-based acoustic models such as the one in Sinsy jointly predict F_0 and other spectral features. However, it has been found that a DNN tends to prioritize higher dimensional spectral features over F_0 [35]. To address this problem, we provide multi-stream architectures that model each feature stream separately. This multi-stream design enables flexible and fine-grained control for modeling different features. Furthermore, to encourage predictions of

[2]Files created by vocal synthesizer UTAU [22] and its open-source successor OpenUTAU [23].

[3]It is possible to use phone-level features, but we currently use note-level features for simplicity.

[4]We could support phone-level features as input for joint optimization of the duration and acoustic models, but we have not implemented it yet.

multiple networks to be coherent, we implemented functionality that conditions the predictions of one network to the input of another, similar to the neural parametric singing synthesizer [15]. In particular, we found that conditioning log-F_0 to the spectral feature prediction models was beneficial.

Our toolkit provides several generic implementations that can be used with the multi-stream architecture: convolutional neural networks (CNNs), recurrent neural networks (RNNs), and advanced architectures such as that of Sinsy and a duration-informed autoregressive model based on the Tacotron [36]. Furthermore, we provide implementations specifically designed for F_0 prediction.

3.3.3. Autoregresssive F_0 models

Modeling F_0 is the key to achieving expressive and natural SVS. Although Sinsy models the dynamic characteristics of the F_0 contour by an explicit vibrato modeling, we provide alternative implementations based on autoregressive models, of which the effectiveness has been confirmed in text-to-speech [37] and SVS [15], [18]. As demonstrated in Section 4.3, autoregressive F_0 models can generate a more natural voice without explicitly modeling vibrato.

Inspired by Sinsy [9] and XiaoiceSing [12], we incorporate residual log-F_0 modeling within the autoregressive models. The choice of detailed architecture may be arbitrary, but we found that a Tacotron-based RNN works well [36].

3.4. Vocoder

NNSVS supports WORLD [26] as a signal processing-based vocoder and uSFGAN [20] as a neural vocoder. WORLD can be used to achieve reasonably good-quality SVS, whereas neural vocoders are generally preferred to achieve better sound quality.

Note that we also support various neural vocoders based on generative adversarial networks (GANs) [38] such as Parallel WaveGAN [39] and HiFi-GAN [40]. However, we found that uSFGAN archived a better tradeoff between quality and pitch robustness.

4. EXPERIMENTAL EVALUATIONS

4.1. Database

To evaluate the performance of NNSVS, we used Namine Ritsu's publicly available database [41]. The database contains 110 songs recorded by a single Japanese singer. Specifically, it includes 4.35 hours of singing data with timings, phonetic, and musical context annotations. We split the 110 songs using a ratio of 100/5/5 for the training, validation, and test sets. respectively. We also split each song into small segments based on the rest notes in the musical scores. Note that we selected test songs to cover a wide range of note pitches: the lowerest and highest notes of the test songs were D#4 (155.6 Hz) and A5 (880 Hz), whereas those of the training data were D#3 (146.8 Hz) and B5 (987.8 Hz). The audio signals were sampled at 44.1 kHz and downsampled to 24 kHz. Each audio was normalized to -26 dB.

At the pre-processing stage, we extracted 82-dimensional score features (e.g., phone identity, note pitch, and note duration) for the time-lag and duration models. Additional four-dimensional coarsely coded positional features [42] were used for the acoustic models.

4.2. Model details

4.2.1. Baselines

As baseline systems, we used our implementations of Sinsy [9], Muskits's RNN-based SVS [14], and the recently proposed DiffSinger [17]. We used the same MDN-based time-lag and duration models in all systems except for Muskits and DiffSinger. In Muskits and DiffSinger, time-lag models were not used, and the duration models were jointly trained with the acoustic models.

We implemented three variants of Sinsy's acoustic model: 1) a simplified version of Sinsy without vibrato modeling, 2) Sinsy with a pitch correction algorithm, and 3) Sinsy with pitch correction and vibrato modeling. The detailed model architecture follows that of Sinsy [9]. As the acoustic features, the systems use WORLD-based 65-dimensional acoustic features containing 60-dimensional MGCs, continuous log-F_0 (LF0), VUVs, and three-dimensional BAP. The frame shift was set to 5 ms. For explicit vibrato modeling, three-dimensional vibrato parameters (i.e., binary flags, amplitude, and speed; denoted as VIB) were additionally used. We did not use dynamic features as we found them to be less useful. During synthesis, a global variance-based post-filter was applied to the MGCs to alleviate over-smoothing issues [43].

For Muskits, we used the officially provided recipe to train the RNN-based SVS [14]. The system uses syllable-level score features and predicts durations together with an 80-dimensional mel-spectrogram. A HiFi-GAN vocoder was used to generate waveforms from the mel-spectrogram [40].

As for DiffSinger, we used the MIDI B-version of the official source code [44]. The system consists of three components: 1) a mel-spectrogram prediction based on a denoising diffusion probabilistic model [45], 2) F_0/VUV prediction from the mel-spectrogram, and 3) a HiFi-GAN vocoder with harmonic-plus-noise mechanism [46] (referred to as hn-HiFi-GAN). We trained each model on Namine Ritsu's database for a fair comparison with other SVS systems.

4.2.2. NNSVS

For the NNSVS systems, we used two types of SVS systems that use mel-spectrogram and WORLD-based features, respectively. The WORLD and mel-spectrogram features contain four (i.e., [MGC, LF0, VUV, BAP]) and three (i.e., [Mel-spectrogram, LF0, VUV]) feature streams, respectively. Note that the details of WORLD features, time-lag model, and duration model are the same as those described in Section 4.2.1. We trained several multi-stream acoustic models with non-autoregressive and autoregressive models for each feature type, as listed in Table 1. For the architecture of the multi-stream models, we adopted the Sinsy architecture for non-autoregressive modeling and the duration-informed Tacotron [36] for autoregressive modeling. We modeled the VUV feature stream by the Sinsy's non-autoregressive architecture in all systems. For the multi-stream models, LF0 was conditioned on the input to the spectral feature prediction models (i.e., models for predicting the MGCs, BAP, and mel-spectrogram) and the VUV prediction model. In addition, NNSVS-WORLD v4 used MGCs as conditioning for predicting BAP. Note that we did not use pitch

Table 1. Naturalness MOS test results with 95% confidence intervals. MEL denotes mel-spectrogram. A/S in the system column represents that the speech samples were generated by the extracted acoustic features. Otherwise, samples were generated by the input musical score. Bold font represents the best score in all SVS systems.

System	Acoustic Features	Mult-stream Architecture	Autoregressive Streams	Vocoder	MOS↑
Sinsy [9]	MGC, LF0, VUV, BAP	No	-	hn-uSFGAN	2.64 ± 0.12
Sinsy (w/ pitch correction) [9]	MGC, LF0, VUV, BAP	No	-	hn-uSFGAN	2.84 ± 0.11
Sinsy (w/ vibrato modeling) [9]	MGC, LF0, VUV, BAP, VIB	No	-	hn-uSFGAN	2.99 ± 0.11
Muskits RNN [10]	MEL	No	-	HiFi-GAN	2.22 ± 0.11
DiffSinger [17]	MEL, LF0, VUV	Yes	-	hn-HiFi-GAN	2.90 ± 0.11
NNSVS-Mel v1	MEL, LF0, VUV	Yes	-	hn-uSFGAN	3.51 ± 0.11
NNSVS-Mel v2	MEL, LF0, VUV	Yes	LF0	hn-uSFGAN	3.58 ± 0.11
NNSVS-Mel v3	MEL, LF0, VUV	Yes	MEL, LF0	hn-uSFGAN	2.58 ± 0.11
NNSVS-WORLD v0 [41]	MGC, LF0, VUV, BAP	No	-	WORLD	3.28 ± 0.10
NNSVS-WORLD v1	MGC, LF0, VUV, BAP	Yes	-	hn-uSFGAN	3.21 ± 0.12
NNSVS-WORLD v2	MGC, LF0, VUV, BAP	Yes	LF0	hn-uSFGAN	3.35 ± 0.11
NNSVS-WORLD v3	MGC, LF0, VUV, BAP	Yes	MGC, LF0	hn-uSFGAN	3.60 ± 0.11
NNSVS-WORLD v4	MGC, LF0, VUV, BAP	Yes	MGC, LF0, BAP	hn-uSFGAN	$\mathbf{3.86 \pm 0.10}$
hn-HiFi-GAN (A/S)	MEL, LF0, VUV	-	-	hn-HiFi-GAN	3.72 ± 0.11
hn-uSFGAN-Mel (A/S)	MEL, LF0, VUV	-	-	hn-uSFGAN	4.19 ± 0.09
hn-uSFGAN-WORLD (A/S)	MGC, LF0, VUV, BAP	-	-	hn-uSFGAN	4.19 ± 0.09
Recordings	-	-	-	-	4.39 ± 0.08

correction methods for NNSVS. During synthesis, global variance post-filters were used for the MGCs and mel-spectrogram. For the neural vocoder architecture, we used the harmonic-plus-noise uSFGAN (hn-uSFGAN) [20]. We trained two hn-uSFGAN vocoders for the mel-spectrogram and WORLD features. Then, the vocoders were used for NNSVS and Sinsy. All the acoustic models and vocoders of NNSVS were trained for 100 epochs and 600 K steps, respectively. More details of the model architecture, training setups, and hyperparameters can be found in our GitHub repository[5].

To evaluate the recent improvements of NNSVS, we included a publicly available pre-trained SVS model (NNSVS-WORLD v0) that was created with an earlier version of NNSVS (November 2021) [41]. The acoustic model was based on the single-stream architecture, and it consisted of stacks of six one-dimensional CNNs with residual connections, followed by an MDN layer. The time-lag and duration models were also based on MDNs.

4.3. Subjective evaluation

We performed MOS tests using a five-point scale for evaluation. Eighteen native Japanese speakers were asked to judge the quality of the singing voice samples. For the tests, five short segments of 3 to 20 seconds were randomly selected for each of the five test songs. In total, 25 samples for each SVS system were evaluated. We also evaluated the synthesized samples generated by the extracted acoustic features using the hn-HiFi-GAN and hn-uSFGAN vocoders.

Table 1 shows the MOS test results, of which the trends can be summarized as follows: (1) Sinsy with explicit vibrato modeling achieved the best score among the three Sinsy variants, which confirms the importance of modeling the dynamic characteristics of F_0. (2) All NNSVS systems except for the NNSVS-Mel v3 outperformed the Sinsy, Muskits, and DiffSinger baseline systems. (3) The source-filter-based hn-uSFGAN performed significantlly better than hn-HiFi-GAN. (4) The autoregressive architecture for the WORLD-based multi-stream models improved the naturalness of the SVS. In particular, NNSVS-WORLD v4 achieved the best score of all the SVS systems (3.86), demonstrating the effectiveness of multi-stream and autoregressive architectures for modeling F_0 and spectral features. (5) Comparing NNSVS-WORLD v4 and NNSVS-WORLD v0, we confirmed that we have obtained substantial performance improvements over the earlier version of NNSVS.

We observed that NNSVS-Mel v3 tended to generate unstable output for low- and high-pitched voices. We hypothesize that NNSVS-Mel v3 obtained a lower score than other NNSVS systems because of the instability and the exposure bias issues. We also note that even though DiffSinger generated a singing voice with higher fidelity than other systems, it often generated an unstable pitch, especially for dynamic voices. We encourage readers to listen to the singing voice samples provided on our demo page[6].

5. CONCLUSION

This paper described the design of NNSVS, an open-source toolkit for SVS research. Our toolkit provides implementations of Sinsy and many new features such as multi-stream models, autoregressive F_0 models, and neural vocoders based on uSFGAN. Experimental results demonstrated that our best system significantlly outperformed the Sinsy, Muskits, and DiffSinger baseline systems. Future work includes adding more advanced architectures based on variational auto-encoders, diffusion models, and GANs. We also plan to work on end-to-end models.

Acknowledgments: This work was partly supported by JST CREST Grant Number JPMJCR19A3.

[5] https://github.com/nnsvs/nnsvs

[6] https://r9y9.github.io/projects/nnsvs/

6. REFERENCES

[1] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, vol. 32, 2019.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[3] S. Young, G. Evermann, M. Gales, *et al.*, "The HTK book," *Cambridge university engineering department*, vol. 3, no. 175, p. 12, 2002.

[4] D. Povey, A. Ghoshal, G. Boulianne, *et al.*, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, IEEE Signal Processing Society, 2011.

[5] H. Zen, T. Nose, J. Yamagishi, *et al.*, "The HMM-based speech synthesis system (HTS) version 2.0," in *Proc. SSW*, vol. 6, 2007, pp. 294–299.

[6] Z. Wu, O. Watts, and S. King, "Merlin: An open source neural network speech synthesis system.," in *Proc. SSW*, 2016, pp. 202–207.

[7] K. Oura, A. Mase, T. Yamada, *et al.*, "Recent development of the HMM-based singing voice synthesis system—Sinsy," in *Seventh ISCA Workshop on Speech Synthesis*, 2010.

[8] Y. Hono, S. Murata, K. Nakamura, *et al.*, "Recent development of the DNN-based singing voice synthesis system—sinsy," in *Proc. APSIPA*, 2018, pp. 1003–1009.

[9] Y. Hono, K. Hashimoto, K. Oura, *et al.*, "Sinsy: A deep neural network-based singing voice synthesis system," *IEEE/ACM Trans. on Audio, Speech, and Lang. Process.*, vol. 29, pp. 2803–2815, 2021.

[10] J. Shi, S. Guo, T. Qian, *et al.*, "Muskits: an End-to-end Music Processing Toolkit for Singing Voice Synthesis," in *Proc. Interspeech*, 2022, pp. 4277–4281.

[11] J. Kim, H. Choi, J. Park, *et al.*, "Korean singing voice synthesis system based on an lstm recurrent neural network," in *Proc. Interspeech*, 2018, pp. 1551–1555.

[12] P. Lu, J. Wu, J. Luan, *et al.*, "XiaoiceSing: A High-Quality and Integrated Singing Voice Synthesis System," in *Proc. Interspeech*, 2020, pp. 1306–1310.

[13] M. Blaauw and J. Bonada, "Sequence-to-sequence singing synthesis using the feed-forward transformer," in *Proc. ICASSP*, 2020, pp. 7229–7233.

[14] J. Shi, S. Guo, N. Huo, *et al.*, "Sequence-to-sequence singing voice synthesis with perceptual entropy loss," in *Proc. ICASSP*, 2021, pp. 76–80.

[15] M. Blaauw and J. Bonada, "A neural parametric singing synthesizer modeling timbre and expression from natural songs," *Applied Sciences*, vol. 7, no. 12, p. 1313, 2017.

[16] Y. Gu, X. Yin, Y. Rao, *et al.*, "ByteSing: A chinese singing voice synthesis system using duration allocated encoder-decoder acoustic models and WaveRNN vocoders," in *Proc. ISCSLP*, 2021, pp. 1–5.

[17] J. Liu, C. Li, Y. Ren, *et al.*, "DiffSinger: Singing voice synthesis via shallow diffusion mechanism," *AAAI*, vol. 36, no. 10, pp. 11 020–11 028, 2022.

[18] Y.-H. Yi, Y. Ai, Z.-H. Ling, *et al.*, "Singing Voice Synthesis Using Deep Autoregressive Neural Networks for Acoustic Modeling," in *Proc. Interspeech*, 2019, pp. 2593–2597.

[19] S. Watanabe, T. Hori, S. Karita, *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[20] R. Yoneyama, Y.-C. Wu, and T. Toda, "Unified Source-Filter GAN with Harmonic-plus-Noise Source Excitation Generation," in *Proc. Interspeech*, 2022, pp. 848–852.

[21] M. Good, "MusicXML for notation and analysis," *The virtual score: representation, retrieval, restoration*, vol. 12, no. 113-124, p. 160, 2001.

[22] Ameya/Ayame, *UTAU*, `http://utau2008.xrea.jp/`, Accessed: 2022.10.06.

[23] StTakira, *Open singing synthesis platform / open source UTAU successor*, `https://github.com/stakira/OpenUtau`.

[24] H.-C. Shen, "Linguistic Extension of UTAU Singing Voice Synthesis and Its Application from Japanese to Mandarin," in *Proc. ECEI*, 2022, pp. 189–192.

[25] Intunist, *The original support for English NNSVS dataset creation*, `https://github.com/intunist/nnsvs-english-support`.

[26] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[27] M. Morise, G. Miyashita, and K. Ozawa, "Low-Dimensional Representation of Spectral Envelope Without Deterioration for Full-Band Speech Analysis/Synthesis System," in *Proc. Interspeech*, 2017, pp. 409–413.

[28] K. Yu and S. Young, "Continuous F0 modeling for HMM based statistical parametric speech synthesis," *IEEE Trans. on Audio, Speech, and Lang. Process.*, vol. 19, no. 5, pp. 1071–1079, 2010.

[29] T. Nakano, M. Goto, and Y. Hiraga, "An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features," in *Proc. Interspeech*, 2006, pp. 1706–1709.

[30] K. Saino, H. Zen, Y. Nankaku, *et al.*, "An HMM-based singing voice synthesis system," in *Ninth International Conference on Spoken Language Processing*, 2006.

[31] C. M. Bishop, "Mixture density networks," Tech. Rep., 1994.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.

[33] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] X. Wang, S. Takaki, and J. Yamagishi, "Investigating very deep highway networks for parametric speech synthesis," *Speech Communication*, vol. 96, pp. 1–9, 2018.

[36] T. Okamoto, T. Toda, Y. Shiga, *et al.*, "Tacotron-based acoustic model using phoneme alignment for practical neural text-to-speech systems," in *Proc. ASRU*, 2019, pp. 214–221.

[37] X. Wang, S. Takaki, and J. Yamagishi, "Autoregressive neural F0 model for statistical parametric speech synthesis," *IEEE/ACM Trans. on Audio, Speech, and Lang. Process.*, vol. 26, no. 8, pp. 1406–1419, 2018.

[38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Proc. NeurIPS*, vol. 27, 2014, pp. 2672–2680.

[39] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *Proc. ICASSP*, 2020, pp. 6199–6203.

[40] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. NeurIPS*, vol. 33, 2020, pp. 17 022–17 033.

[41] Canon, *[NamineRitsu] Blue (YOASOBI) [ENUNU model Ver.2, Singing DBVer.2 release]*, `https://www.youtube.com/watch?v=pKeo9IE_L1I`, Accessed: 2022.10.06.

[42] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Proc. ICASSP*, 2015, pp. 4470–4474.

[43] T. Toda and K. Tokuda, "A speech parameter generation algorithm considering global variance for HMM-based speech synthesis," *IEICE Trans. on Information and Systems*, vol. 90, no. 5, pp. 816–824, 2007.

[44] L. Jinglin, *DiffSinger: Singing voice synthesis via shallow diffusion mechanism*, `https://github.com/MoonInTheRiver/DiffSinger`.

[45] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Proc. NeurIPS*, vol. 33, pp. 6840–6851, 2020.

[46] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *IEEE/ACM Trans. on Audio, Speech, and Lang. Process.*, vol. 28, pp. 402–415, 2019.