Singing voice synthesis via latent flow matching and differentiable digital signal processing

Serhii Lupenko[1,†], Maksym Klishch[2,*,†], Vasyl Yatsyshyn[2,†] and Oleh Pastukh[2,†]

[1] *Faculty of Computer Science, Opole University of Technology, 45-758 Opole, Poland*
[2] *Ternopil Ivan Puluj National Technical University, Ruska Street 56, T46001 Ternopil, Ukraine*

Abstract
Recent advances in singing voice synthesis (SVS) have achieved high perceptual quality through variational and diffusion-based generative frameworks. However, diffusion models require many iterative steps for inference, while variational approaches may suffer from a mismatch between prior and posterior distributions, affecting pitch accuracy and pronunciation. We propose a flow matching-based latent generative model that integrates a DDSP autoencoder with a latent flow predictor for efficient and expressive singing voice synthesis. By combining spectral parameter modeling with continuous latent flow transformation, the system achieves high-fidelity waveform generation with reduced computational cost. Experimental results demonstrate that the proposed model attains comparable perceptual quality to state-of-the-art baselines while using fewer parameters and achieving faster inference.

Keywords
Singing voice synthesis, flow matching, differentiable digital signal processing, acoustic modelling.

1. Introduction

Singing Voice Synthesis (SVS) aims to generate expressive singing from musical scores and lyrics. Unlike Text-to-Speech, SVS must precisely reproduce pitch, rhythm, and expressive nuances such as vibrato and legato.

Despite their conceptual similarity, SVS presents a set of challenges that are substantially more complex than those in TTS. First, the pitch contour in singing spans a much wider dynamic range and requires frame-level accuracy: even small deviations in fundamental frequency $F0$ can lead to perceptually unnatural or dissonant results. Second, the temporal structure of singing is governed by musical rhythm and note duration rather than by natural speech prosody. This introduces a strong dependency between phoneme alignment and musical timing, where mismatched durations or onsets can severely distort lyrical intelligibility. Third, expressive performance characteristics such as vibrato, portamento, and dynamic loudness variation are essential for naturalness, yet are difficult to model using standard text-to-speech architectures. Moreover, singing datasets are typically smaller and less diverse than speech corpora, limiting the robustness of purely data-driven approaches.

Over the past two decades, singing voice synthesis has evolved from concatenative to neural generative paradigms. Early systems such as Vocaloid [1] and UTAU relied on concatenative playback of recorded phonemes, offering manual control but limited expressiveness. Statistical models like Sinsy [2] introduced hidden Markov modeling for note durations and pitch, enabling automation at the cost of oversmoothed timbre. With deep learning, architectures such as XiaoiceSing [3] and DeepSinger [4] adopted transformer-based encoders for non-autoregressive prediction, improving pitch and rhythm stability.

AIT&AIS'2025: International Scientific Workshop on Applied Information Technologies and Artificial Intelligence Systems, December 18–19 2025, Chernivtsi, Ukraine
[1*] Corresponding author.
[†] These authors contributed equally.

lupenko.san@gmail.com (S. Lupenko); maksim.klishch@gmail.com (M. Klishch); vyatcyshyn@gmail.com (V. Yatsyshyn); oleg.pastuh@gmail.com (O. Pastukh)
0000-0002-6559-0721 (S. Lupenko); 0009-0003-6242-9953 (M. Klishch); 0000-0002-5517-6359 (V. Yatsyshyn); 0000-0002-0080-7053 (O. Pastukh)

Recent neural SVS systems address these issues through variational, diffusion, or flow-based generative modeling. While architectures, such as VISinger [5], achieve high quality and can generalize with limited data, they often suffer from training-inference mismatch between the posterior (audio) and prior (musical score) distributions, which can lead to inaccurate pitch and mispronunciations [6]. Diffusion models, including HiddenSinger [6] and DiffSinger [7], deliver state-of-the-art fidelity but require iterative denoising in high-dimensional acoustic space, leading to slow inference and high computational cost.

2. Related Works

2.1. Singing voice synthesis models

Recent research in SVS has progressed through the integration of deep learning, neural audio codecs, diffusion processes, flow-based methods, and digital signal processing (DSP) techniques. The evolution of SVS models reflects a shift from deterministic acoustic modeling toward probabilistic and latent generative paradigms.

One of the systems of this generation, XiaoiceSing [3], employs a FastSpeech-style architecture that combines phoneme, positional, and musical features to predict phoneme durations and the fundamental frequency F_0. The model performs non-autoregressive acoustic prediction while incorporating rhythmic and melodic conditioning.

The VISinger series [5, 8] is built upon the VITS architecture (Variational Inference with adversarial learning for end-to-end Text-to-Speech). It integrates variational latent modeling, flow transformations, and adversarial training to jointly model acoustic features and waveform reconstruction. VISinger 2 extends this framework by introducing differentiable digital signal processing (DDSP) blocks and a HiFi-GAN vocoder, explicitly separating harmonic and noise components. The system operates in an end-to-end manner, synthesizing 44.1 kHz audio, and was trained for 500 k steps on 5 hours of data.

DiffSinger [7] adopts diffusion probabilistic modeling for generating mel-spectrograms. The model transforms Gaussian noise into target spectrograms through a conditional diffusion process, and its shallow diffusion mechanism reduces the number of denoising steps to improve generation efficiency.

HiddenSinger [6] combines a neural audio codec with a latent diffusion model. The system encodes singing audio into a low-dimensional latent space, performs generation within this space, and reconstructs full-band audio through a neural decoder. It was trained on 150 hours of data for 3 million steps with a batch size of 32. The variant HiddenSinger-U supports semi-supervised training on unpaired singing data.

Multilingual and zero-shot adaptation is explored in TCSinger 2 [9], which enables cross-lingual and cross-singer synthesis without additional fine-tuning. Meanwhile, TechSinger [10] employs the flow matching paradigm for controllable SVS, allowing explicit manipulation of vocal techniques such as vibrato and breathiness across multiple languages.

Overall, contemporary SVS systems encompass a wide spectrum of architectures—from nonautoregressive models (XiaoiceSing) and VITS-based variational frameworks (VISinger, VISinger 2) to diffusion-based (DiffSinger), latent-diffusion (HiddenSinger), and flow-based (TechSinger) approaches — advancing toward multilinguality, controllability, and expressive singing synthesis.

2.2. Recent flow matching based TTS models

Recent advances in text-to-speech (TTS) modeling demonstrate a steady transition from autoregressive architectures to flow-based and latent generative approaches aimed at improving synthesis efficiency, prosodic coherence, and controllability. Several works explore different formulations of flow matching and diffusion processes for speech generation.

MetaTTS [11] integrates pre-training strategies into TTS and examines how large-scale flow matching can enhance generalization and naturalness. This study represents one of the first attempts to apply flow matching at the scale of foundation speech models, bridging pre-trained acoustic representations with generative modeling.

Building upon this line, F5-TTS [12] investigates long-form speech synthesis, emphasizing textspeech alignment and temporal consistency. Using the flow matching framework, it maintains coherent prosody across extended utterances, highlighting the suitability of flow-based methods for narrative and expressive speech generation.

Matcha-TTS [13] proposes a streamlined conditional flow matching architecture optimized for lowlatency synthesis. The model reduces inference time while preserving competitive audio quality, indicating its potential for real-time applications.

A complementary direction is explored in LatentSpeech [14], which applies latent diffusion modeling to TTS. Speech is generated within a compact latent space rather than directly in the spectral or temporal domain, reducing computational requirements while retaining acoustic detail. This approach demonstrates the efficiency gains achievable through latent-space diffusion.

VoiceFlow [15] extends the flow-based paradigm by employing rectified flow matching, reformulating the underlying dynamics to minimize integration steps and inference cost. The resulting model achieves comparable perceptual quality with improved computational efficiency.

Finally, ProsodyFlow [16] integrates conditional flow matching with prosody modeling informed by large speech language models. This combination allows explicit control over high-level prosodic dimensions such as intonation, rhythm, and stress, illustrating how linguistic conditioning can enhance expressiveness in neural TTS.

3. Proposed method

This section presents the proposed latent-conditioned architecture for singing voice synthesis, which combines a flow-based latent predictor with a DDSP-based autoencoder for waveform reconstruction, as illustrated in Figure 1.

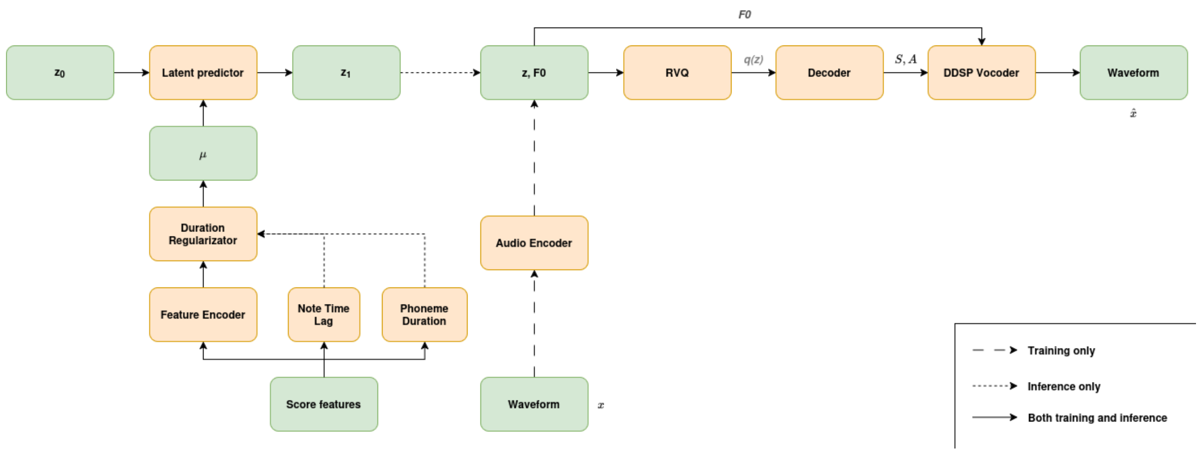

Figure 1: Overview of the proposed architecture.

3.1. Autoencoder

Architecture adopts a differential DSP based autoencoder. Following the approach of HiddenSinger [6], we integrate Residual Vector Quantization (RVQ) blocks into the bottleneck to enable compact latent representation of input features.

3.1.1. Encoder

Given an input waveform $x \in \mathbb{R}^{L}$, where N is the number of audio samples, the encoder $E(\cdot)$ extracts a latent feature sequence:

$$z = E(x) \in \mathbb{R}^{d \times T}. \tag{1}$$

where d denotes the feature dimension and T is the number of encoded frames.

The encoder transforms the raw waveform into a time-aligned latent representation that summarizes its relevant acoustic structure. This representation retains information about spectral shape, energy, and temporal evolution while discarding fine-grained sample-level details that are unnecessary for higher-level modeling.

3.1.2. Residual Vector Quantization

RVQ approximates a signal $z \in \mathbb{R}^d$ as the sum of L quantized vectors, each selected from a codebook $C^{(i)}$, $i = 1, 2, \dots, L$, of fixed size M_i:

$$q(z) = \sum_{i=1}^{L} q_i(r_i), \tag{2}$$

where $q_i : \mathbb{R}^d \longrightarrow C^{(i)}$ is the quantization function for the i-th codebook $C^{(i)}$, and r_i is the residual vector at the i-th quantization step:

$$r_1 = z, \quad r_i = z - \sum_{j=1}^{i-1} q_j(r_j). \tag{3}$$

This structure enables high-fidelity reconstruction with compact codebooks, establishes a discrete prior over the latent space, and imposes structural constraints on the latent manifold.

3.1.3. Decoder and vocoder

The decoder is adapted from the architecture proposed in [17], adopting a Emformer-blocks [18]. The vocoder follows the design principles of [17, 19], combining differentiable signal synthesis with spectral parameter decoding. It operates on three feature components: the fundamental frequency F_0, the spectral envelope S, and the aperiodicity A.

The excitation signal for the harmonic component is defined as the sum of sawtooth-like signals [20]:

$$e_h(t) = \sum_{k=1}^{K} \frac{1}{k} \sin(\varphi_k(t)), \tag{4}$$

where K is the number of harmonics and $\varphi_k(t)$ is the instantaneous phase of the k-th harmonic, given by:

$$\varphi_k(t) = 2\pi k \int_0^t F_0(\tau)\, d\tau. \tag{5}$$

The excitation signal is then passed through a frequency-domain filter to generate the harmonic signal:

$$x_h = \mathscr{F}^{-1}((1 - A) S \mathscr{F}(e_h)), \tag{6}$$

where $\mathscr{F}$ and $\mathscr{F}^{-1}$ denote the forward and inverse short-time Fourier transforms (STFT) with a window length of 2048 samples and 75% overlap, A is the aperiodicity coefficient, and S is the spectral shaping filter.

The initial excitation for the noise component is defined as Gaussian white noise:

$$e_n(t) \sim N(0, 1). \tag{7}$$

The noise signal is then generated by applying the same spectral filter with the aperiodicity weighting:

$$x_n = \mathscr{F}^{-1}(AS\mathscr{F}(e_n)). \tag{8}$$

Finally, the synthesized singing voice signal is obtained as a weighted sum of the harmonic and noise components:

$$\hat{x}=\lambda_h x_h+\lambda_n x_n, \tag{9}$$

where λ_h and λ_n are coefficients that control the relative contribution and amplitude of the harmonic and noise components, respectively.

Following [19] we adopt compression/decompression for S and A.

The linear spectral envelope S is mapped to the log-mel domain using a mel filterbank M:

$$S_c=\log_{10}(MS+\varepsilon), \tag{10}$$

and reconstructed during synthesis as:

$$S=M_0 10^{S_c}-\varepsilon, \tag{11}$$

where $M_0 = \max(M^{-1}, 0)$ is a non-negative pseudo-inverse of M applied elementwise to avoid negative reconstruction artifacts.

For aperiodicity, the decoder predicts a compressed representation A_c with 16 channels, which is linearly interpolated along the frequency axis to obtain the full 513-dimensional aperiodicity A used by the differentiable signal synthesis module.

3.1.4. Training criteria

The autoencoder model is optimized using a combination of spectral reconstruction, vector quantization, and adversarial objectives. Each term contributes to different aspects of perceptual and structural fidelity in the generated waveform.

Reconstruction loss. To ensure accurate waveform reconstruction, we adopt a multi-resolution STFT loss:

$$\mathscr{L}_{STFT}=E\sum_{i=1}^{6}\left\|S_i(x)-S_i(\hat{x}_i)\right\|_2^2+\left\|\log S_i(x)-\log S_i(\hat{x}_i)\right\|_2^2, \tag{12}$$

where S_i represents normalized STFT with size of FFT and window size 2^{5+i} and hop lenght 25%.

Vector quantization commitment loss. For residual vector quantization, the commitment objective encourages stable codebook utilization [21]:

$$\mathscr{L}_{RVQ}=E\sum_{i=1}^{L}\left\|r_i-q_i(r_i)\right\|_2^2. \tag{13}$$

Discriminative loss. Following [22], we adopt both a multi-period discriminator (MPD) and a multiscale discriminator (MSD) to enhance the perceptual quality of the generated waveform. The MPD is designed to capture pitch-synchronous periodic structures across multiple temporal periods, while the MSD operates on differently scaled versions of the waveform to assess spectral consistency and long-term coherence. Both discriminators are jointly trained with the generator using least-squares adversarial objectives and a feature-matching term.

The generator G and discriminators $\left\{D_{MPD}^{(i)}\right\}_{i=1}^{N_p}$, $\left\{D_{MSD}^{(j)}\right\}_{j=1}^{N_s}$, are optimized using least-squares adversarial losses [23]. For a target waveform x and its generated counterpart $\hat{x}=G(\cdot)$, the objectives are defined as follows.

The generator loss is expressed as:

$$\mathscr{L}_G=E_{\hat{x}}\left[\sum_{i=1}^{N_p}\left(D_{MPD}^{(i)}(\hat{x})-1\right)^2+\sum_{j=1}^{N_s}\left(D_{MSD}^{(j)}(\hat{x})-1\right)^2\right]. \tag{14}$$

The discriminator loss is defined symmetrically as:

$$\mathscr{L}_D=E_{x,\hat{x}}\left[\sum_{i=1}^{N_p}\left(\left(D_{MPD}^{(i)}(x)-1\right)^2+\left(D_{MPD}^{(i)}(\hat{x})\right)^2\right)+\sum_{j=1}^{N_s}\left(\left(D_{MSD}^{(j)}(x)-1\right)^2+\left(D_{MSD}^{(j)}(\hat{x})\right)^2\right)\right]. \tag{15}$$

3.2. Flow matching

Flow matching [24] is an approach for learning a direct deterministic mapping from a simple prior distribution to a complex target data distribution using a vector field. Unlike diffusion models, where the transformation process is governed by stochastic differential equations (SDEs), Flow matching relies on a deterministic ordinary differential equation (ODE) that defines a continuous trajectory between distributions.

We adopts flow matching to predict latent embeddings z by modeling a time-dependent vector field $v_t(z|\mu)$, which describes the transformation of the base distribution $p_0 = \mathcal{N}(0, 1)$ into a target distribution p_1 that approximates the latent data manifold:

$$\frac{d\,\psi_t(z|\mu)}{dt} = v_t(\psi_t(z|\mu)), \tag{16}$$

where $\psi : [0, 1] \times \mathbb{R}^{d \times T} \longrightarrow \mathbb{R}^{d \times T}$ is a time-dependent flow function defined as:

$$\psi_t(z|\mu) = (1-t)z + tz_1. \tag{17}$$

Here, z_1 denotes a sample from the data distribution in latent space, and $t \in [0, 1]$ parameterizes the transformation path from the prior to the target distribution.

Model is training using reparametrized optimal-transport conditional flow matching [25]:

$$\mathcal{L}_{CFM} = E_{t \sim U[0,1], z_1 \sim q} \frac{1}{1-t} \left\| z_1 - v_t(z_t|\mu) \right\|_2^2. \tag{18}$$

To couple the conditioning pathway with the latent target, we also align encoder features μ to the target latent z_1 via an auxiliary MSE:

$$\mathcal{L}_{feature} = E \left\| \mu - z_1 \right\|_2^2. \tag{19}$$

This auxiliary coupling encourages the encoder features to carry predictive information about the target latent while preserving the flow-matching dynamics.

3.3. Time lag and note duration

In singing voice synthesis, the temporal alignment between musical notes and phonemes plays a critical role in maintaining the naturalness and intelligibility of the generated performance. Following the approach proposed in [26], we model the time-lag and note duration as two separate prediction tasks.

3.3.1. Time-lag model

The time-lag represents the offset between the onset of a musical note in the score and the actual beginning of the corresponding phoneme in the singing performance. This offset primarily arises because consonants often precede the note onset, while vowels align more closely with the note boundary. Let g_n denote the reference time-lag for the n-th note and $\hat{g}_n$ the predicted value. The model is trained to minimize the mean squared error between predicted and reference lags:

$$\mathcal{L}_{lag} = E \left\| g - \hat{g} \right\|_2^2. \tag{20}$$

During synthesis, the predicted lag shifts the onset of each note, providing a corrected effective duration $\hat{L}_n$ for subsequent phoneme allocation.

3.3.2. Duration model

The duration model predicts the length of each phoneme within a note, given musical and phonetic features. For the n-th note containing K_n phonemes, the model outputs the expected phoneme durations $\hat{d}_{nk}$ under the constraint that their total sum equals the adjusted note length $\hat{L}_n$:

$$\sum_{k=1}^{K_n} \hat{d}_{nk} = \hat{L}_n. \tag{21}$$

The duration network is trained with a mean squared error criterion between predicted and reference phoneme durations:

$$\mathcal{L}_{dur} = E\left\|d - \hat{d}\right\|_2^2. \tag{22}$$

This objective encourages accurate phoneme-level timing, ensuring that predicted durations remain consistent with the musical score.

4. Experiment

4.1. Datasets

For training, we used the Tohoku Kiritan dataset [27], which contains 50 songs with a total duration of approximately 3.5 hours. The first three songs were used for testing, the next three for validation, and the remaining 44 for training.

All audio signals were downsampled to 24 kHz and normalized to -26 dB. The dataset was segmented into 4 second samples, with segmentation boundaries aligned to word-level timing and natural pauses to preserve linguistic and prosodic coherence.

4.2. Training

We trained the model using the Adam optimizer [28] with a learning rate of 10−5, β_1 = 0.8, and β_2 = 0.999 for 250k steps. All experiments were conducted on a single NVIDIA RTX 4060 GPU with a batch size of 16 and fp16.

All components of the proposed system are optimized jointly to achieve accurate temporal alignment, spectral reconstruction, and perceptual fidelity. The overall training objective integrates the note- and phoneme-level temporal models with the latent and waveform reconstruction modules. To ensure stable convergence, the discriminators are introduced after 20k steps of autoencoder pretraining, allowing the generator to learn coarse spectral structures before adversarial feedback is applied. Full joint optimization of the encoder, decoder, flow predictor, and discriminators begins after 200k steps, enabling coordinated fine-tuning of all modules under both reconstruction and adversarial losses.

The total loss function is defined as

$$\mathcal{L}_{total} = \lambda_{lag}\mathcal{L}_{lag} + \lambda_{dur}\mathcal{L}_{dur} + \lambda_{STFT}\mathcal{L}_{STFT} + \lambda_{RVQ}\mathcal{L}_{RVQ} + \\ + \lambda_{GAN}\mathcal{L}_{GAN} + \lambda_{FM}\mathcal{L}_{FM} + \lambda_{feature}\mathcal{L}_{feature}, \tag{23}$$

where each λ denotes a weighting coefficient that balances the contribution of the corresponding term.

The loss weights are empirically set to λ_{lag} = 0.02, λ_{dur} = 0.02, λ_{STFT} = 1.0, λ_{RVQ} = 0.5, λ_{GAN} = 0.5, λ_{FM} = 1.0, and $\lambda_{\mathrm{feature}}$ = 0.1 with the values selected based on validation performance across objective metrics (detailed in Section 5).

4.3. Implementation details

4.3.1. Autoencoder

Following the design of the DDSP architecture [29], we reuse their F_0- and z-encoder components, while omitting the loudness encoder, as our model does not rely on loudness features. In our formulation, overall signal amplitude and perceived loudness are expected to be implicitly represented within the latent variable z. The F_0 encoder employs a pretrained CREPE pitch estimator [30]. For the z-encoder, we adopt the same feature extraction and network structure as DDSP, which maps MFCC-based representations to a compact latent embedding $z(t)$ through a GRU layer.

The RVQ module is implemented with 8 quantizers, each comprising a codebook of 1024 entries with dimension 128.

4.3.2. Latent Generator

Following the architecture of Matcha-TTS [13], the latent generator predicts the denoised latent sequence conditioned on time t and features μ. It consists of six convolutional-attention blocks with residual connections and explicit time conditioning at each layer.

Each block combines a residual 1D convolution (kernel size 7) for local context modeling and an Emformer [18] layer for efficient long-range temporal attention. In contrast to Matcha-TTS, which adopts Transformer blocks from BigVGAN [31], we replace them with Emformer layers to improve efficiency and support streaming inference. A sinusoidal time embedding with dimension 64 is concatenated to the input of each block, allowing the model to represent the continuous diffusion or flow-matching timestep. Skip connections between early and deep layers facilitate information flow. The output features are projected by a multilayer perceptron to match the target latent dimensionality of 129 channels per frame, where 128 dimensions correspond to the latent representation z and one additional channel represents the fundamental frequency F_0.

Following Matcha-TTS [13] the conditioning features are first processed by an encoder that maps linguistic and prosodic inputs into a continuous representation μ. This representation serves as the conditioning variable for the latent generator, providing frame-level context about phonetic content, duration, and pitch.

For inference, the final latent trajectory is obtained by numerically integrating $v_t(x_t)$ using the first-order Euler method.

4.4. Baselines

We used two singing voice synthesis systems for comparison: HiddenSinger and ViSinger2. For ViSinger2, we adopted the official implementation provided in the ESPNet toolkit [32], following the default configuration released by the authors. The HiddenSinger model was reimplemented according to the architecture and training setup described in the original paper to ensure consistent preprocessing and evaluation conditions. For inference, 50 denoising steps were employed to generate the final outputs.

Table 1
Experimental results on the test dataset. Mean Opinion Score (MOS) and Real-Time Factor (RTF) are reported with 95% confidence intervals, while other metrics are presented as point estimates. Objective metrics include Mel-Cepstral Distortion (MCD), F_0 Root Mean Square Error (F_0-RMSE), voicing/unvoicing accuracy (V/UV F_1), and computational efficiency indicators such as model size and peak memory usage

Model	Params.	Peak memory usage (GB)	RTF	MOS	MCD (dB)	F_0-RMSE	V/UV F_1
Ground Truth	—	—	—	4,51 ± 0,07	0,00	0,00	1,000
Autoencoder (recon.)	3,2M	0,29	0,010 ± 0,002	4,25 ± 0,10	1,47	15,03	0,968
ViSinger2	25,7M	4,07	0,021 ± 0,004	3,81 ± 0,08	2,71	19,74	0,951
HiddenSinger	27,2M	3,82	0,784 ± 0,017	3,49 ± 0,11	2,81	23,14	0,948
Ours (1 Step)	18,6M	1,94	0,016 ± 0,003	3,56 ± 0,07	2,92	23,52	0,937
Ours (8 Steps)	18,6M	1,94	0,063 ± 0,003	3,76 ± 0,13	2,78	22,89	0,949

Both baselines were trained and evaluated on the same dataset splits as our proposed model for a fair comparison.

5. Results

To evaluate the proposed model, we conducted both subjective and objective assessments on a held out test set of the singing voice dataset. The comparison includes two baseline systems, ViSinger2 and HiddenSinger, along with an autoencoder reconstruction reference and the ground-truth recordings. For our model, we report performance under two inference configurations: a single flow-matching step (1 Step) and with eight steps (8 Steps).

5.1. Subjective Evaluation

Subjective quality was evaluated on a subset of 80 synthesized samples using five independent listeners. Each participant rated the perceptual quality on a five-point Mean Opinion Score (MOS) scale, where higher scores indicate more natural and pleasant sound.

5.2. Objective Evaluation

Objective evaluation was performed to assess spectral and prosodic accuracy. Mel-cepstral distortion (MCD, in dB) quantifies spectral deviation between the synthesized and reference signals. The F_0 root mean square error (F_0-RMSE) measures the pitch contour deviation, and the voiced/unvoiced (V/UV) F_1-score reflects the accuracy of voicing decisions. As reference F_0 and V/UV features, we used those extracted from the target recordings with the WORLD vocoder [33].

5.3. Computational Efficiency

To evaluate computational efficiency, we also report peak memory usage during inference and the real-time factor (RTF). Peak memory usage (in GB) indicates the maximum GPU memory consumption per sample generation, while RTF measures the ratio of synthesis time to audio duration.

Table 1 summarizes the quantitative and perceptual results across all compared systems.

Figure 2 presents a direct comparison between the target and predicted mel-spectrograms. This visualization highlights howclosely the synthesized output follows the temporal and harmonic structures of the reference recording. We include this figure to qualitatively illustrate the model's ability to reconstruct pitch contours, note onsets, and harmonic formant structures that are not fully captured by objective metrics alone.

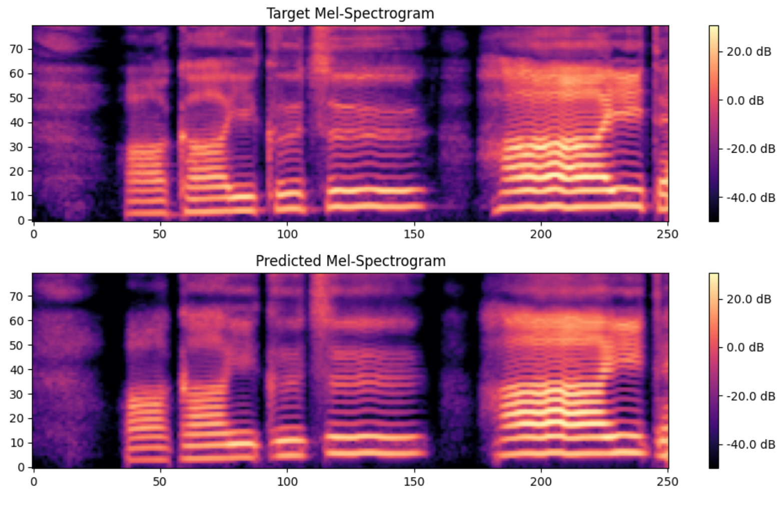

Figure 2: Comparison between target and predicted mel-spectrograms. The top panel shows the ground-truth mel-spectrogram extracted from the reference audio, while the bottom panel depicts the mel-spectrogram generated by the proposed model.

Table 2
Ablation results based on MOS with 95% confidence intervals

Model	MOS
Flow Matching (8 Step)	3,76 ± 0,13
Diffusion (50 Steps)	3,32 ± 0,16
DDSP Decoder	3,76 ± 0,13
SiFiGAN	3,73 ± 0,15
HiFiGAN	3,57 ± 0,09

5.4. Ablation study

To evaluate the contribution of the main architectural components, we design two ablation comparisons aligned with widely used alternatives in neural singing voice synthesis. The first compares our 8-step flow matching generator with a diffusion-based decoder trained under identical data and conditioning settings. The second replaces our DDSP-based acoustic decoder with commonly used GAN vocoders, including SiFiGAN [34] and HiFiGAN [22].

All ablation systems are trained using the same dataset, optimization schedule, and text–pitch conditioning to ensure comparability. Evaluation is performed on the same held-out test subset as the main experiments.

Table 2 reports subjective evaluation results for the ablation settings.

6. Discussion

The results in Table 1 demonstrate that the proposed model achieves a favorable trade-off between quality and efficiency. Compared to ViSinger2 and HiddenSinger, our approach requires 28% fewer parameters and less than half of the peak GPU memory during inference, while maintaining comparable perceptual quality. In particular, the 8-step configuration attains a MOS of 3.76, approaching ViSinger2 (3.81), yet operates with nearly twice the speed, as indicated by the lower RTF. The 1-step version achieves real-time generation (RTF < 0.02), confirming that the flow-matching formulation enables efficient synthesis with only a few iterative updates of the velocity field v_t, rather than hundreds of stochastic denoising steps required by diffusion models.

Objective metrics further show consistent spectral and prosodic accuracy. The slight increase in F_0-RMSE and V/UV F_1 compared to the baselines suggests minor pitch and voicing deviations, which may stem from the reduced latent dimensionality. However, the lower MCD for the 8-step configuration indicates improved spectral coherence and harmonic balance, particularly under multi-step refinement that progressively aligns the predicted latent with the target manifold.

While diffusion and flow-based generative models both rely on stochastic sampling, excessive stochasticity can sometimes introduce pitch or timing variability across runs, slightly degrading consistency. Our method mitigates this by learning smoother velocity fields through flow matching, requiring far fewer steps than diffusion-based denoising while retaining stochastic flexibility for expressive control. It is also important to note that the original HiddenSinger [6] was trained on a substantially larger dataset (approximately 150 hours of recording), whereas our evaluation targets smaller datasets to assess generalization in low-resource scenarios. This difference in data scale likely contributes to the observed gap in subjective quality.

The faster and competitive performance of our system stems from three main design factors. First, the DDSP-based decoder operates directly on compact spectral parameters – the envelope S_c and aperiodicity A_c – rather than full-resolution spectrograms, reducing computational cost while preserving perceptual detail. Second, modeling in the latent space substantially lowers the prediction dimensionality, allowing the flow-matching network to learn smoother dynamics with fewer parameters. Third, the flow-matching objective enables efficient iterative generation by estimating continuous velocity fields instead of performing high-step stochastic denoising. Although the individual contribution of each factor has not been explicitly analyzed, their combined effect enables the proposed system to achieve near state-of-the-art quality with faster synthesis and significantly reduced memory usage.

In addition, Figure 2 qualitatively supports these findings by showing that the predicted melspectrogram closely matches the reference in both temporal and spectral structure. The alignment of formants and harmonic trajectories indicates that the proposed model effectively captures the finegrained frequency evolution of the singing voice, complementing the quantitative metrics in Table 1.

The ablation results show that the flow matching generator achieves higher perceptual quality than the diffusion-based alternative under identical training conditions, while avoiding the latency overhead associated with multi-step sampling. Furthermore, the DDSP-based autoencoder demonstrates perceptual performance comparable to SiFiGAN and superior to HiFiGAN.

7. Conclusions

In this paper, we presented a singing voice synthesis model architecture that integrates a DDSP-based autoencoder with a flow-matching latent predictor. The proposed model effectively combines explicit signal decomposition with continuous latent flow modeling, enabling expressive and high-fidelity singing voice generation. Experimental results demonstrate that our system achieves comparable perceptual quality to state-of-the-art approaches while requiring significantly fewer parameters. These results suggest that incorporating flow-based latent modeling within a DDSP framework provides an efficient and interpretable alternative for neural singing synthesis. Future work will focus on extending the proposed method toward multi-singer and cross-lingual scenarios, as well as exploring finer control over expressive parameters such as vibrato and dynamics.

Declaration on Generative AI

During the preparation of this work, the authors used GPT and Gemini in order to: Grammar and spelling check. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

[1] H. Kenmochi, H. Ohshita, Vocaloid – Commercial singing synthesizer based on sample concatenation, in: Proceedings of the Interspeech 2007: 8th Annual Conference of the International Speech Communication Association, International Speech Communication Association, Lausanne, Switzerland, 2007, pp. 4009–4010.

[2] K. Oura, A. Mase, T. Yamada, S. Muto, Y. Nankaku, K. Tokuda, Recent development of the HMM-based singing voice synthesis system – Sinsy, in: Proceedings of the 7th ISCA Workshop on Speech Synthesis, SSW-7 ’2010, International Speech Communication Association, Lausanne, Switzerland, 2010, pp. 211–216.

[3] P. Lu, J. Wu, J. Luan, X. Tan, L. Zhou, XiaoiceSing: A high-quality and integrated singing voice synthesis system, arXiv preprint arXiv:2006.06261 (2020). doi:10.48550/arXiv.2006.06261.

[4] Y. Ren, X. Tan, T. Qin, J. Luan, Z. Zhao, T.-Y. Liu, DeepSinger: Singing voice synthesis with data mined from the web, arXiv preprint arXiv:2007.04590 (2020). doi:10.48550/arXiv.2007.04590.

[5] Y. Zhang, J. Cong, H. Xue, L. Xie, P. Zhu, M. Bi, ViSinger: Variational inference with adversarial learning for end-to-end singing voice synthesis, arXiv preprint arXiv:2110.08813 (2022). doi:10.48550/arXiv.2110.08813.

[6] J.-S. Hwang, S.-H. Lee, S.-W. Lee, HiddenSinger: High-quality singing voice synthesis via neural audio codec and latent diffusion models, Neural Netw. 181 (2025). doi:10.1016/j.neunet.2024.106762.

[7] J. Liu, C. Li, Y. Ren, F. Chen, Z. Zhao, DiffSinger: Singing voice synthesis via shallow diffusion mechanism, arXiv preprint arXiv:2105.02446 (2022). doi:10.48550/arXiv.2105.02446.

[8] Y. Zhang, H. Xue, H. Li, L. Xie, T. Guo, R. Zhang, C. Gong, ViSinger 2: High-fidelity end-to-end singing voice synthesis enhanced by digital signal processing synthesizer, arXiv preprint arXiv:2211.02903 (2022). doi:10.48550/arXiv.2211.02903.

[9] Y. Zhang, W. Guo, C. Pan, D. Yao, Z. Zhu, Z. Jiang, Y. Wang, T. Jin, Z. Zhao, TCSinger 2: Customizable multilingual zero-shot singing voice synthesis, arXiv preprint arXiv:2505.14910 (2025). doi:10.48550/arXiv.2505.14910.

[10] W. Guo, Y. Zhang, C. Pan, R. Huang, L. Tang, R. Li, Z. Hong, Y. Wang, Z. Zhao, TechSinger: Technique-controllable multilingual singing voice synthesis via flow matching, arXiv preprint arXiv:2502.12572 (2025). doi:10.48550/arXiv.2502.12572.

[11] A. H. Liu, M. Le, A. Vyas, B. Shi, A. Tjandra, W.-N. Hsu, Generative pre-training for speech with flow matching, arXiv preprint arXiv:2310.16338 (2024). doi:10.48550/arXiv.2310.16338.

[12] Y. Chen, Z. Niu, Z. Ma, K. Deng, C. Wang, J. Zhao, K. Yu, X. Chen, F5-TTS: A fairytaler that fakes fluent and faithful speech with flow matching, arXiv preprint arXiv:2410.06885 (2025). doi:10.48550/arXiv.2410.06885.

[13] S. Mehta, R. Tu, J. Beskow, É. Székely, G. E. Henter, Matcha-TTS: A fast TTS architecture with conditional flow matching, arXiv preprint arXiv:2309.03199 (2024). doi:10.48550/arXiv.2309.03199.

[14] H. Lou, H. Paik, P. D. Haghighi, W. Hu, L. Yao, LatentSpeech: Latent diffusion for text-to-speech generation, arXiv preprint arXiv:2412.08117 (2024). doi:10.48550/arXiv.2412.08117.

[15] Y. Guo, C. Du, Z. Ma, X. Chen, K. Yu, VoiceFlow: Efficient text-to-speech with rectified flow matching, arXiv preprint arXiv:2309.05027 (2024). doi:10.48550/arXiv.2309.05027.

[16] H. Wang, S. Shan, Y. Guo, Y. Wang, ProsodyFlow: High-fidelity text-to-speech through conditional flow matching and prosody modeling with large speech language models, in: Proceedigs of the The 31st International Conference on Computational Linguistics, COLING '2025, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2025, pp. 7748–7753.

[17] P. Agrawal, T. Koehler, Z. Xiu, P. Serai, Q. He, Ultra-lightweight neural differential DSP vocoder for high-quality speech synthesis, arXiv preprint arXiv:2401.10460 (2024). doi:10.48550/arXiv.2401.10460.

[18] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, M. Seltzer, Emformer: Efficient memory transformer-based acoustic model for low latency streaming speech recognition, arXiv preprint arXiv:2010.10759 (2020). doi:10.48550/arXiv.2010.10759.

[19] S. Nercessian, Differentiable World synthesizer-based neural vocoder with application to end-to-end audio style transfer, arXiv preprint arXiv:2208.07282 (2923). doi:10.48550/arXiv.2208.07282.

[20] D.-Y. Wu, W.-Y. Hsiao, F.-R. Yang, O. Friedman, W. Jackson, S. Bruzenak, Y.-W. Liu, Y.-H. Yang, DDSP-based singing vocoders: A new subtractive-based synthesizer and a comprehensive evaluation, arXiv preprint arXiv:2208.04756 (2022). doi:10.48550/arXiv.2208.04756.

[21] A. Défossez, J. Copet, G. Synnaeve, Y. Adi, High-fidelity neural audio compression, arXiv preprint arXiv:2210.13438 (2022). doi:10.48550/arXiv.2210.13438.

[22] J. Kong, J. Kim, J. Bae, HiFi-GAN: Generative adversarial networks for efficient and high-fidelity speech synthesis, arXiv preprint arXiv:2010.05646 (2020). doi:10.48550/arXiv.2010.05646.

[23] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, S. P. Smolley, Least squares generative adversarial networks, arXiv preprint arXiv:1611.04076 (2017). doi:10.48550/arXiv.1611.04076.

[24] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, M. Le, Flow matching for generative modeling, arXiv preprint arXiv:2210.02747 (2023). doi:10.48550/arXiv.2210.02747.

[25] T. Luo, X. Miao, W. Duan, WaveFM: A high-fidelity and efficient vocoder based on flow matching, arXiv preprint arXiv:2503.16689 (2025). doi:10.48550/arXiv.2503.16689.

[26] Y. Hono, K. Hashimoto, K. Oura, Y. Nankaku, K. Tokuda, Sinsy: A deep neural network-based singing voice synthesis system, IEEE/ACM Trans. Audio Speech Lang. Process. 29 (2021) 2803–2815. doi:10.1109/TASLP.2021.3104165.

[27] I. Ogawa, M. Morise, Tohoku Kiritan singing database: A singing database for statistical parametric singing synthesis using Japanese pop songs, Acoust. Sci. Technol. 42 (2021) 140–145. doi:10.1250/ast.42.140.

[28] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2017). doi:10.48550/arXiv.1412.6980.

[29] J. Engel, L. Hantrakul, C. Gu, A. Roberts, DDSP: Differentiable digital signal processing, arXiv preprint arXiv:2001.04643 (2020). doi:10.48550/arXiv.2001.04643.

[30] J. W. Kim, J. Salamon, P. Li, J. P. Bello, CREPE: A convolutional representation for pitch estimation, arXiv preprint arXiv:1802.06182 (2018). doi:10.48550/arXiv.1802.06182.

[31] S. Lee, W. Ping, B. Ginsburg, B. Catanzaro, S. Yoon, BigVGAN: A universal neural vocoder with large-scale training, arXiv preprint arXiv:2206.04658 (2023). doi:10.48550/arXiv.2206.04658.

[32] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, T. Ochiai, ESPnet: End-to-end speech processing toolkit, arXiv preprint arXiv:1804.00015 (2018). doi:10.48550/arXiv.1804.00015.

[33] M. Morise, F. Yokomori, K. Ozawa, WORLD: A vocoder-based high-quality speech synthesis system for real-time applications, IEICE Trans. Inf. Syst. 99 (2016) 1877–1884. doi:10.1587/transinf.2015EDP7457.

[34] R. Yoneyama, Y.-C. Wu, T. Toda, Source-filter hifi-gan: Fast and pitch controllable high-fidelity neural vocoder, arXiv preprint arXiv:2210.15533 (2023). doi:10.48550/arXiv.2210.15533.