DFlow: A Generative Model Combining Denoising AutoEncoder and Normalizing Flow for High Fidelity Waveform Generation

Chenfeng Miao [1] **Qingying Zhu** [1] **Minchuan Chen** [1] **Wei Hu** [1] **Zijian Li** [2] **Shaojun Wang** [1] **Jing Xiao** [1]

Abstract

In this work, we present DFlow, a novel generative framework that combines Normalizing Flow (NF) with a Denoising AutoEncoder (DAE), for high-fidelity waveform generation. With a tactfully designed structure, DFlow seamlessly integrates the capabilities of both NF and DAE, resulting in a significantly improved performance compared to the standard NF models. Experimental results showcase DFlow's superiority, achieving the highest MOS score among the existing methods on commonly used datasets and the fastest synthesis speed among all likelihood models. We further demonstrate the generalization ability of DFlow by generating high-quality out-of-distribution audio samples, such as singing and music audio. Additionally, we extend the model capacity of DFlow by scaling up both the model size and training set size. Our large-scale universal vocoder, DFlow-XL, achieves highly competitive performance against the best universal vocoder, BigVGAN.

1. Introduction

Deep Generative Models (DGMs) have gained remarkable success in the wave generation task in recent years. Starting with WaveNet (Oord et al., 2016), many DGMs have flourished in the waveform generation field, including Generative Adversarial Networks (GAN) (Kong et al., 2020; Kumar et al., 2019; You et al., 2021; Yang et al., 2020) and likelihood-based models, such as autoregressive (AR) models (Oord et al., 2016; Kalchbrenner et al., 2018; Valin & Skoglund, 2019), normalizing flows (NF) (Kim et al., 2019; Prenger et al., 2019), and diffusion models (Chen et al., 2021; Kong et al., 2021). Compared with other generative models, NFs have many theoretical advantages, including exact likelihood evaluation, efficient training and inference, and controllable latent representation, making them especially attractive. Unfortunately, on many generation tasks, the standard NFs generally do not have competitive generation performance against GAN-based models (Kong et al., 2020; Lee et al., 2023; Brock et al., 2019) or other likelihood models (Oord et al., 2016; Chen et al., 2021; Kong et al., 2021; Ho et al., 2020). These other generative models, on the other hand, also have pragmatic drawbacks over NFs. For example, AR or diffusion models require multiple steps to generate the output, while GAN models often suffer from mode collapse when scaling up the model size (Lee et al., 2023; Brock et al., 2019).

In this work, we aim to construct a powerful generative framework that inherits the advantages of NFs while overcoming their limitations, thus further holding the potential to surpass other generative models across various aspects. For this purpose, we begin this work by investigating the limitations of standard NF models. We find that NF-based models are not robust to small input variations and often encounter initial errors during inference. This inspires us to enhance the NF model with a denoising autoencoder (DAE) (Vincent et al., 2008). However, our preliminary results indicate that simple integration of NF and DAE poses significant challenges to stable training. To address this issue, we tactfully designed the model structure by having an autoregressive flow module acting as the denoising encoder, constraining the primary flow to be volume-preserving, and utilizing a feed-forward decoder. These design choices empower our model, the DFlow, with stable training and surprisingly improved performance. To summarize, DFlow has the following advantages:

- Stable and efficient training. DFlow is trained from scratch, using only one training stage. Moreover, DFlow enjoys very efficient training thanks to its fully convolutional and fully parallel structure.
- Fast synthesis. The synthesis speed of DFlow is significantly faster than the standard NF models. To the best of our knowledge, DFlow sets a new benchmark of synthesis speed among likelihood-based waveform generation models and is approaching the speed of

[1]Ping An Technology, Shanghai, China [2]Georgia Institute of Technology, GA, US. Correspondence to: Chenfeng Miao <miao_chenfeng@126.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

GAN models.

- High-quality generation. The audio quality of DFlow is significantly improved compared to the standard NF models, and even better or at least comparable to the GAN models.
- High robustness. We demonstrate the impressive generalization capabilities of DFlow, which, though trained only on speech data, performs exceptionally well in synthesizing out-of-domain non-speech waveforms, including singing and music waveforms, where standard NF models often fail.

Furthermore, the contemporary landscape of machine learning exhibits a tendency towards large-scale models, as evidenced by previous studies (Brock et al., 2019; Henighan et al., 2020) suggesting that increasing the dataset scale and model size can enhance the model performance. Taking inspiration from this, we proceeded to train DFlow-XL, an expanded version of DFlow, by increasing both the training set scale and the model size of DFlow. Experimental findings suggest that as the model size increases, DFlow exhibits noteworthy performance improvements, thereby showcasing its scalability.

Our audio samples are available on the demo website[1], and the implementation of our model is provided as open source to ensure reproducibility and facilitate future research efforts[2].

2. Background

In this section, we briefly describe the basic concepts of normalizing flows (NFs) and then discuss the limitations of NF-based generative models.

2.1. Normalizing Flows

Normalizing flows (NFs) are a category of generative models that model one distribution $p_X(x)$ as a parametric invertible transformation f_θ to another distribution $p_Z(z)$, where $p_X(x)$ is the unknown data distribution on the data space X and $p_Z(z)$ is a known distribution, such as a multivariate Gaussian, on the latent space Z. Following the *change of variable formula*, the likelihood of a data sample $x \in X$ can be exactly computed as:

$$p_X(x) = p_Z(f_\theta(x)) \left| \det \frac{\partial f_\theta}{\partial x} \right| \tag{1}$$

where $\frac{\partial f_\theta}{\partial x}$ is the Jacobian matrix of f_θ at x. An NF model is typically trained by maximizing the log-likelihood of the training data with respect to the model parameter θ. For better expressiveness, f_θ is often parameterized by stacking multiple invertible transformations, where each individual transformation has an exact computation of the Jacobian determinant and the inverse transformation. In practice, these individual transformations are often designed as triangular mappings such that the time complexity of computing the determinant of such $n \times n$ Jacobian matrix is $O(n)$ as opposed to $O(n^3)$ for an unconstrained matrix.

AR transformations vs. Non-AR transformations. A particular type of invertible transformation that is often used as building blocks of NFs is the autoregressive (AR) transformation (Germain et al., 2015; Papamakarios et al., 2017). Due to the autoregressive nature, the Jacobian of this type of transformation is triangular by design and, hence, can be easily computed. A drawback of AR transformation is recursive sampling, which is inefficient on parallel devices such as GPUs. Therefore, modern NF models tend to use non-AR transformations (Dinh et al., 2015; 2017; Kingma & Dhariwal, 2018). Unlike invertible AR transformations, non-AR transformations enable parallel computation for both training and sampling.

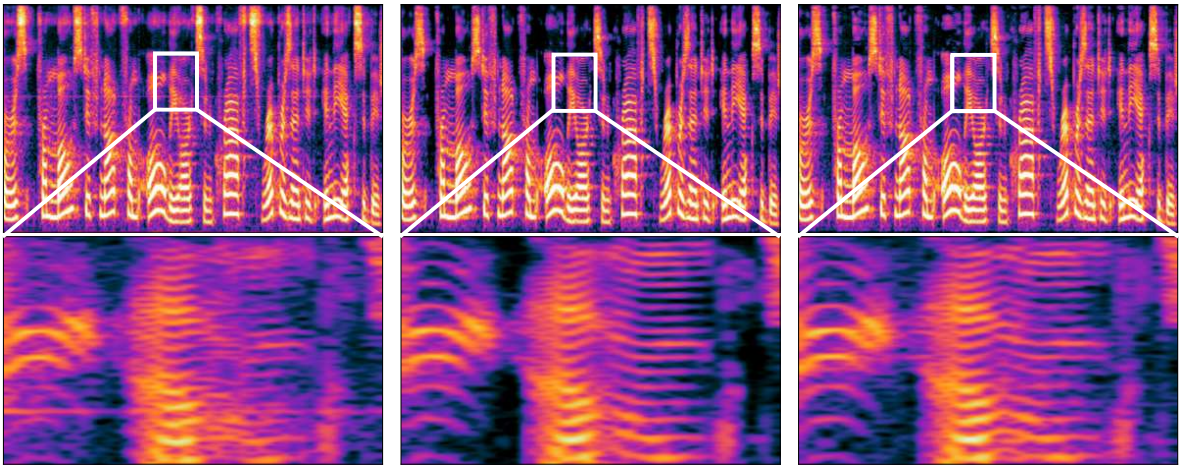
VP transformations vs. NVP transformations. Invertible transformations can be categorized into volume-preserving (VP) transformations and non-volume-preserving (NVP) transformations according to different Jacobian determinants. Specifically, a VP transformation has a unit Jacobian determinant (Dinh et al., 2015) while an NVP (Dinh et al., 2017; Kingma & Dhariwal, 2018) transformation does not.

2.2. Limitations of NF-based generative models

The model capability of NF models relies exclusively on maximizing likelihood estimation (MLE). Unfortunately, much evidence shows that the penalties and priorities imposed by MLE are not always aligned with human perception (Theis et al., 2016; Espuña I Fontcuberta, 2022; Grover et al., 2018). As for waveform generation, the likelihood of the time-domain signal does not consistently match human evaluations. Human judgment of waveform quality relies heavily on the frequency-domain information. However, as natural signals, even minor variations in the time domain can induce perceptible changes in the frequency domain. For example, as shown in Fig. 1, from a frequency-domain perspective, the harmonic components of the waveform generated by WaveGlow are quite blurry compared to the ground truth. Thus, to ensure high-quality waveform generation, the model has to be sufficiently robust to small changes in its input data.

In addition, NF models often suffer from the initial error of mismatch between $p(f_\theta(x))$ and $p(z)$ (Ramasinghe et al., 2022). Moreover, nonlinear systems, such as NF-based models, often amplify initial errors and yield substantially distinct outputs even with minor variations in the input data

[1] https://mcf330.github.io/DFlowDemo/
[2] https://github.com/mcf330/DFlowCode



WaveGlow DFlow Ground Truth

Figure 1. Mel-spectrograms of waveforms generated by WaveGlow, DFlow, and the ground truth, with zoomed-in views of high-frequency details. Both the waveforms generated by DFlow and the ground truth exhibit clear harmonic components. However, the one generated by WaveGlow not only has blurry harmonic components but also periodic noises.

(Higham, 2002; Lanckriet et al., 2002). In the context of waveform generation, it is a common practice to transform the one-dimensional waveform into a multi-dimensional vector using a *Squeeze* (Prenger et al., 2019; Ping et al., 2020) operation. This step is necessary for the widely employed affine coupling layer (Dinh et al., 2017; Kingma & Dhariwal, 2018). However, the periodic operation introduces biases into $p(f_\theta(x))$ during training. Consequently, it leads to initial errors during the sampling process, resulting in periodic artifacts in the generated waveforms. The throughout horizontal lines peculiar to the mel-spectrogram of the waveform generated by WaveGlow in Fig. 1 are visualizations of these periodic artifacts.

3. Method

3.1. Overview

Given the standard NF often encounters errors during inference, a natural solution to address this problem is to train a refinement module that enhances the capacity of the standard NF. In considering this, we split the generation of the complex data distribution $p(\boldsymbol{x})$ into two steps: a generation step that learns a noisy data distribution $p(\tilde{\boldsymbol{x}})$ and a refinement step that learns the clean data distribution based on the noisy data distribution $p(\boldsymbol{x}|\tilde{\boldsymbol{x}})$. Here, the noisy data $\tilde{\boldsymbol{x}}$ is obtained by adding a random Gaussian noise $\boldsymbol{\epsilon}$ with a small scalar β to each clean data sample: $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \beta\boldsymbol{\epsilon}$. The generation of $p(\tilde{\boldsymbol{x}})$ is formulated as an NF model, while the refinement step $p(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ is a DAE.

As presented in Fig. 2a, the proposed framework consists of 3 modules: the Auxiliary Flow network f, the Primary Flow network g, and the Decoder network m. The Auxiliary Flow plays two fundamental roles at the same time. On the one hand, the two NF modules, the Auxiliary Flow and the Primary Flow, formulate a standard NF that serves as an invertible transformation from the noisy input $\tilde{\boldsymbol{x}}$ into a Gaussian prior $\boldsymbol{z}_p$. On the other hand, the Auxiliary Flow also acts as the DAE encoder that transforms the noisy input $\tilde{\boldsymbol{x}}$ into the latent variable $\boldsymbol{z}_l$, from which the DAE decoder m learns to reconstruct the clean signal.

The sampling is processed by first inverting the Primary Flow module: $\boldsymbol{z}_l = g^{-1}(\boldsymbol{z}_p)$ and then producing the output through the decoder: $\hat{\boldsymbol{x}} = m(\boldsymbol{z}_l)$. The Auxiliary Flow module f is excluded from the sampling phase.

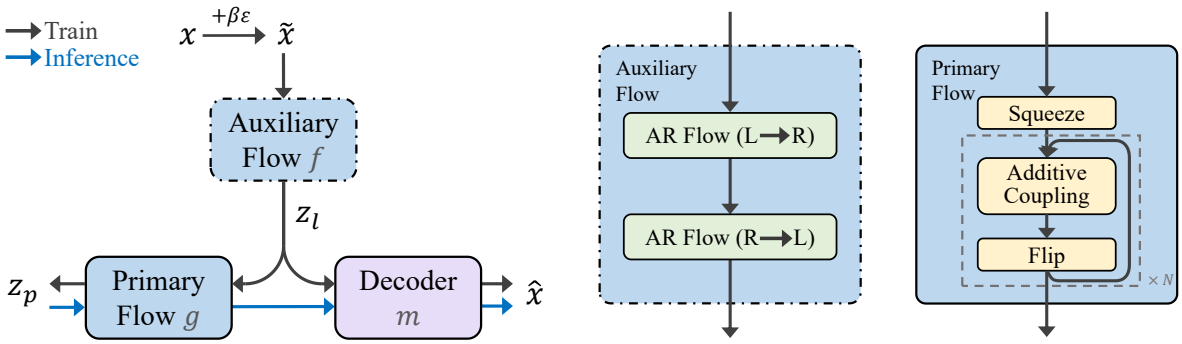
3.2. Training objective

The primary goal of probabilistic generative modeling is to maximize the marginal log-likelihood of each training sample $\boldsymbol{x} \in X$ with respect to the model parameters $\boldsymbol{\theta}$. This requires the marginalization of any latent variables in the model:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p_{\boldsymbol{\theta}}(\boldsymbol{z_l})\, p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z_l}) d\boldsymbol{z_l} \tag{2}$$

Here, $\boldsymbol{\theta}$ refers to the generative parameters from network g and m. Since the Auxiliary Flow network f is an invertible flow module, Eq. 2 can be rewritten as:

$$\begin{aligned} &\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \\ &= \log \int p_{\boldsymbol{\theta}}(\boldsymbol{z_l}) p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{x}}|\boldsymbol{z_l}) p_{\boldsymbol{\phi}}(\boldsymbol{z_l}|\tilde{\boldsymbol{x}})\, p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z_l}) d\boldsymbol{z_l}\, d\tilde{\boldsymbol{x}} \\ &= \log \int p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\tilde{\boldsymbol{x}})\, p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \end{aligned} \tag{3}$$



(a) Overall Structure (b) Auxiliary Flow (c) Primary Flow

Figure 2. Architecture of DFlow.

Here, ϕ refers to the parameters of f. This integration is still intractable, so instead, we optimize a lower bound on the marginal likelihood following the variational principle:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \log \int p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\tilde{\boldsymbol{x}})\, p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \\ &= \log \int \frac{q_\beta(\tilde{\boldsymbol{x}}|\boldsymbol{x})}{q_\beta(\tilde{\boldsymbol{x}}|\boldsymbol{x})} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\tilde{\boldsymbol{x}})\, p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \\ &\geq \mathbb{E}_{q_\beta(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \\ &\quad + \log p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\tilde{\boldsymbol{x}}) - \log q_\beta(\tilde{\boldsymbol{x}}|\boldsymbol{x})] \end{aligned} \tag{4}$$

This bound is often referred to as the evidence lower bound (ELBO). The first term corresponds to the expected negative reconstruction error of DAE and the last two terms can usually be combined as the KL divergence between the approximated posterior and the prior distribution: $D_{\mathrm{KL}}(q_\beta(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\tilde{\boldsymbol{x}}))$. Notice that β is a constant that does not depend on the model parameters, therefore the last term in Eq. 4 does not need to be optimized during training. Then the training objective can be rewritten as the combination of negative log-likelihood of $\tilde{\boldsymbol{x}}$ given by the NF and the reconstruction error $\mathcal{L}_{rec}$ given by the DAE:

$$\mathcal{L}_{all} = -\log p(\tilde{\boldsymbol{x}}) + \mathcal{L}_{rec} \tag{5}$$

Negative log-likelihood. Following Eq. 1, the negative log-likelihood (NLL) of $\tilde{\boldsymbol{x}}$ can be computed as:

$$\begin{aligned} -\log p(\tilde{\boldsymbol{x}}) &= -\log(p(\boldsymbol{z}_p)) - \log(|\det(J(f))|) \\ &\quad - \log(|\det(J(g))|) \qquad (6) \\ &= -\log(p(\boldsymbol{z}_p)) - \log(|\det(J(f))|) \qquad (7) \end{aligned}$$

where $\boldsymbol{z}_p \sim N(\mathbf{0}, \boldsymbol{I})$, $\det(J(f))$ and $\det(J(g))$ are the Jacobian determinants of f and g respectively. Here g is designed to be a VP flow, therefore $\det(J(g)) = 1$ as mentioned in section 2.1, and thus $\log(|\det(J(g))|) = 0$.

Reconstruction error. The reconstruction error of DAE is measured by MAE (Mean Absolute Error) between the predicted audio $\hat{\boldsymbol{x}}$ and clean audio $\boldsymbol{x}$. The MAE is multiplied by a positive scalar $1/\beta$, which is consistent with those noise learning models and diffusion models (Lee et al., 2021; Kong et al., 2021):

$$\mathcal{L}_{rec} = \frac{1}{\beta} ||\boldsymbol{x} - \hat{\boldsymbol{x}}||_1 \tag{8}$$

In the next subsections, we discuss the model in detail.

3.3. Noise injection

It is noteworthy that, in the absence of noise injection into the input $\boldsymbol{x}$, the network f and m together form a plain AutoEncoder (AE), in which the decoder serves as an estimated inverse of the encoder. However, in this scenario, the proposed $g + m$ generation process would yield inferior results compared to directly inverting the $g + f$ NF module, as the estimated inverse provided by m would inevitably be less accurate than the exact inverse of f. Therefore, from a design perspective, we aim for the decoder m to offer more than just an estimated inverse of f. To achieve this, we introduce noise through the use of a DAE, which confers several advantages to the model. Firstly, the Decoder m now learns beyond the exact input, implying the potential for superior performance compared to the standard NF model. Secondly, the incorporation of noise injection enables the model to acquire expressive latent representations that exhibit heightened sensitivity and robustness to minor input variations. Consequently, this enhances the model's ability to generate accurate high-frequency details. Thirdly, the Decoder becomes more resilient to minor errors arising from inverting the Primary Flow network during inference, thereby further improving the overall quality of the generated outputs.

3.4. Auxiliary Flow

The Auxiliary Flow f is made up of a stack of invertible AR transformations. As previously mentioned, it serves both as part of the standard NF and as the encoder of the DAE.

One could query the rationale behind designing the DAE encoder f to be invertible here. The invertible encoder in DFlow provides explicit computing of its Jacobian determinant, therefore allowing us to train a standard NF directly on $\tilde{\boldsymbol{x}}$, an observable variable with a fixed distribution. This key feature sets DFlow apart from Latent Flows Models (LFM) (Böhm & Seljak, 2022; Xiao et al., 2019), in which a standard NF is trained on the latent variables of an AE. Unfortunately, LFMs face a common limitation where, in the cause of stable training, the latent space variable must be separately learned before training the NF model. This often leads to separate training stages (Böhm & Seljak, 2022) or separate optimization using a stop gradient operation (Xiao et al., 2019). We direct readers to (Xiao et al., 2019) for detailed explanations. Unlike LFMs, DFlow computes the log-likelihood on a variable of a fixed distribution, rather than on a variable of an unfixed distribution as in LFMs, thus overcoming the challenges of training stability encountered by LFMs. Additionally, unlike traditional NF models that require explicit and efficient computation of both the Jacobian determinant and the inverse of the transformation, in our scenario, concerns about inverting f can be disregarded as f is completely excluded from the sampling process. This gains us the freedom to build f with invertible AR transformations, avoiding the drawback of slow inverse while enjoying the benefits outlined below. Firstly, invertible AR transformations are typically much more expressive than NAR transformations (Papamakarios et al., 2017; Ping et al., 2020) and often require fewer layers. Secondly, AR transformations have no *Squeeze* operation that may hurt the overall performance as discussed in Section 2.2. Thirdly, some invertible AR transformations, for instance, those utilizing causal convolutions (Oord et al., 2016), conduct parallel training and therefore offer high training efficiency.

As shown in Fig. 2b, the Auxiliary Flow comprises one left-to-right block and one right-to-left block. As a result, the model might exhibit inductive bias in both directions. Each block contains 4 single-directional AR transformations, which are specified as follows (taking the left-to-right transformation for example):

$$\log s_i, b_i, \boldsymbol{h}'_i = \mathrm{WaveNet}(\boldsymbol{x}_{1:i-1}, \boldsymbol{h}_{1:i-1}) \tag{9}$$

$$x'_i = s_i \cdot x_i + b_i \tag{10}$$

where, $\mathrm{WaveNet}(\cdot,\cdot)$ is the standard casual WaveNet block (Oord et al., 2016), $\boldsymbol{h} \in \mathcal{R}^{D_h \times T}$ is a hidden variable with a dimensionality of D_h. For each block, $\boldsymbol{h}$ is initialized as a zero vector and passed through all AR transformations within this block. The variable $\boldsymbol{h}_i$ has no impact on the Jacobian determinant because it is derived from previous samples $\boldsymbol{x}_{1:i-1}$, which means that $\frac{\partial h_i}{\partial x_i} = 0$. We purposefully set $D_h \gg 1$ so that the model could carry extensive information from one transformation to the next. This effectively addresses the bottleneck limitation of NF-based models mentioned in several earlier researches (Chen et al., 2020; Grcić et al., 2021).

3.5. Primary Flow

Aside from the considerations on stable training mentioned in Section 3.4, obtaining an expressive yet steady latent variable $\boldsymbol{z}_l$ is also essential as the final output $\hat{\boldsymbol{x}}$ is derived from $\boldsymbol{z}_l$. Here, the Primary Flow g is specifically designed as a VP flow. If both f and g were NVP flows, then the NLL of $\hat{\boldsymbol{x}}$ would become Eq. 6 rather than Eq. 7. The constraint on maximizing the Jacobian determinant of f is then weakened since there is an extra term in Eq. 6 contributing to the NLL. In this case, f is not forced to explore the latent space, resulting in $\boldsymbol{z}_l$ having a particularly small variance and therefore less expressive. The training is also less stable as $\boldsymbol{z}_l$ is under-constrained but the reconstruction process relies heavily on it. By constraining g to be volume-preserving, $\boldsymbol{z}_l$ is guaranteed to have the same variance as $\boldsymbol{z}_p$, which works as a regularization constraint on $\boldsymbol{z}_l$, making it steady. The volume-preserving constraint also encourages f to produce $\boldsymbol{z}_l$ with variance, making it more expressive, thereby enhancing the overall performance.

In detail, as shown in Fig. 2c, a *Squeeze* layer is used to rearrange the 1-dimentional latent variable $\boldsymbol{z}_l \in \mathcal{R}^{1\times T}$ into a 4-dimensional vector $\boldsymbol{z}'_l \in \mathcal{R}^{4\times T/4}$. The squeezed vector $\boldsymbol{z}'_l$ is then passed through a stack of additive coupling layers (Dinh et al., 2015). The order of the channels is reversed by a flip layer after each coupling layer. We developed a $\mathrm{UNet}(\cdot)$ structure to parameterize the additive coupling since the computational efficiency of the additive coupling layers is essential to the overall model efficiency:

$$\boldsymbol{x}_1, \boldsymbol{x}_2 = \mathrm{Split}(\boldsymbol{x}) \quad \boldsymbol{b} = \mathrm{UNet}(\boldsymbol{x}_2) \tag{11}$$

$$\boldsymbol{x}'_1 = \boldsymbol{x}_1 + \boldsymbol{b} \quad \boldsymbol{x}'_2 = \boldsymbol{x}_2 \quad \boldsymbol{x}' = \mathrm{Concat}(\boldsymbol{x}'_1, \boldsymbol{x}'_2) \tag{12}$$

The $\mathrm{UNet}(\cdot)$ structure contains a downsampling stage followed by an upsampling stage. Three strided and three transposed convolutional layers are used to perform the downsampling and upsampling respectively, with residual blocks of dilated convolutions between every single layer. The input vector is downsampled by factors of (4, 4, 4) and then upsampled with the same factors, while the channel size of each residual block gradually increases or decreases by a factor of 2 for the downsampling or upsampling process. We add as many residual connections as possible between the downsampling and upsampling stages. The detailed implementation of the residual blocks and $\mathrm{UNet}(\cdot)$ can be found in Appendix A.1.

3.6. Decoder

The Decoder m is formulated by a stack of feed-forward convolution layers that do not need to be invertible. Specifically,

the Decoder is constructed using a series of UNet layers mentioned in Section 3.5. These UNet layers provide the model with the required inductive bias for high-resolution and high-quality waveform generation, including a large scale of the receptive field and efficient computation. Inspired by (Kong et al., 2020), a $\tanh$ activation is employed to generate the final output. Further implementation details can be found in Appendix A.1.

To facilitate a better understanding of each module within DFlow, we provide a concise summary of their individual characteristics in Table 1.

4. Related work

To the best of our knowledge, DFlow is the first model that jointly trains NF and DAE. Prior to our work, (Böhm & Seljak, 2022) proposes a hybrid model for image generation that cascades an NF and an AE using sequential training stages. Another approach, as seen in (Xiao et al., 2019), attempts to combine NF with an AE by employing a stop gradient operation on the latent variable. To address the topological constraint problem, (Horvat & Pfister, 2021) uses a denoising training objective to train the NF model by bidirectionally running the model during training. Concurrent to our work, (Silvestri et al., 2023) proposes to train a VAE model using an NF-based encoder and a Gaussian decoder. DFlow offers several advantages over the aforementioned models, including end-to-end training and a fully deterministic generative network. Most importantly, DFlow demonstrates strong generative performance, further distinguishing it from the aforementioned models.

In the waveform generation field, GAN approaches are the dominant techniques due to their efficacy in achieving high-quality outputs and efficient synthesis (Kong et al., 2020; Kumar et al., 2019; Lee et al., 2023; Miao et al., 2022). Recently, diffusion models (Kong et al., 2021; Chen et al., 2021), which are built from a hierarchy of DAEs, have been shown to achieve high-quality results. Although diffusion models do not exhibit mode collapse or training instabilities as GAN models, they suffer from slow synthesis speed. Flow-based neural vocoders (Prenger et al., 2019; Kim et al., 2019; Ping et al., 2020; Lee et al., 2020; Luong & Tran, 2022) are an important line of research for our work. Most of these models (Ping et al., 2020; Luong & Tran, 2022; Ping et al., 2020) focus on developing powerful invertible layers, while some models (Ping et al., 2020; Lee et al., 2020; Luong & Tran, 2022) focus on reducing the model footprint. This work, on the other hand, focuses on addressing the common limitations of standard normalizing flows, allowing for better expressiveness, robustness, and model efficiency.

From a broader perspective, the proposed model can be seen as a special case of Variational AutoEncoder (VAE) (Kingma & Welling, 2014) (which we referred to as the standard VAE), where $q_\beta(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ acts as a non-trainable approximate posterior encoder, $p_{f+m}(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ and $p_{f+g}(\tilde{\boldsymbol{x}})$ serve as the decoder and the prior respectively. However, there is a key distinction between DFlow and the standard VAE: the standard VAE employs a stochastic encoder, whereas DFlow utilizes a deterministic encoder. The stochastic structure of standard VAE raises the reconstruction error between the input and the estimated output, while the reconstruction in DFlow is much easier to learn. For instance, in the high-fidelity waveform generation field, the standard VAE models are often accompanied by an adversarial training objective to minimize the reconstruction error (Kim et al., 2021; Miao et al., 2024) while DFlow is adversarial-free. Another notable distinction between DFlow and the standard VAE is that the standard VAE typically assumes a latent variable with a known distribution, often the Gaussian distribution. In contrast, DFlow does not impose such a constraint on the latent variable, allowing for more flexibility in modeling the underlying data distribution.

Inverse autoregressive flow (IAF) (Kingma et al., 2016) is also similar to DFlow as they both employ autoregressive flows to enhance the capacity of posterior distributions. However, the autoregressive flow module in DFlow serves not only as an enhancement module of the posterior distribution but also as an integral component of a general NF that collectively forms the prior distribution in conjunction with the rest of the flow modules. In addition, IAF is built on top of the standard VAE framework, whereas DFlow is quite different from standard VAEs as discussed in the last paragraph. The unique combination of DAE and NF in DFlow sets it apart from both IAF and standard VAEs, enabling improved performance and distinct modeling capabilities.

DFlow is trained by injecting noise into the training data, making it closely related to prior efforts on improving NF via dequantization (Ho et al., 2019; Yoon et al., 2020). The proposed noise injection can be viewed as a form of dequantization applied to the 16-bit discrete waveform signals. However, it is important to note that our approach differs from conventional dequantization approaches. In prior works like (Ho et al., 2019; Yoon et al., 2020), the injected noises are maintained at a very small scale to ensure the noisy data remains indistinguishable from the original clean data. In contrast, DFlow introduces unconstrained noise, resulting in noisy data that may significantly differ from the original clean data. Moreover, our design perspective goes beyond mere data dequantization. Our empirical findings suggest that training DFlow with small-scale noise (e.g., setting $\beta = 0.0001$), which is sufficient for dequantization purposes, performs only at a comparable level to the standard NF. However, we noticed a significant improvement when we intended to "inject noise" into the input data and trained the model with larger-scale noises (e.g., setting

Table 1. Summary of characteristics of each DFlow module.

Modules	VP / NVP	AR / NAR	Invertible	In sampling	Building block
Auxiliary Flow	NVP	AR	yes	no	WaveNet
Primary Flow	VP	NAR	yes	yes	UNet
Decoder	-	NAR	no	yes	UNet

$\beta = 0.01$).

5. Evaluations on commonly used datasets

5.1. Setup

To facilitate the comparison of results across different models, we first conduct experiments on two commonly used speech datasets: a single-speaker speech dataset named LJ-Speech (Ito, 2017) with a total length of approximately 24 hours and a multi-speaker speech dataset named VCTK (Yamagishi et al., 2019) with 109 speakers and a total length of about 44 hours. All audio samples were resampled to 22.05 kHz despite their original sampling rates varied. We used 80-band mel-spectrograms as the input conditions. The FFT size, window length, and hop size were set to 1024, 1024, and 256, respectively. We upsampled mel-spectrograms to different length scales ($4\times, 16\times, 64\times, 256\times$) through 4 transposed 1-D convolution layers, with the upsampling rates of $[4, 4, 4, 4]$. Each transposed convolution layer is followed by a Leaky ReLU activation. The upsampled mel-spectrograms of different length scales are fed as conditions to each causal WaveNet layer and each UNet layer of DFlow. The detailed model configurations of DFlow are shown in Appendix A.2. We trained the DFlow models on 4 Nvidia Tesla V100 GPU (16G), with a batch size of 32 per GPU. The AdamW optimizer (Loshchilov & Hutter, 2019) with $\beta_1 = 0.8, \beta_2 = 0.99$ is used to train the models. The initial learning rate is 2×10^{-4} and it decays at every training epoch with a decay rate of 0.997. We trained each model for a total of 1M iterations. The subjective evaluations are conducted through the Mean Opinion Score (MOS) test and the side-by-side Comparative Mean Opinion Score (CMOS) test. Details of these tests are discussed in Appendix A.3.

5.2. Benchmarking baseline models

We first evaluate DFlow on the mel-spectrogram inversion task by comparing it with several publicly available models on the LJ-Speech dataset, such as an AR model WaveNet (MoL) (Yamamoto, 2018), a standard NF model WaveGlow (Valle, 2018), an adversarial model HiFi-GAN (Kong, 2020), and a diffusion model DiffWave (LMNT, 2022). For all these models, we took the pretrained models on the LJ-Speech dataset from their GitHub pages for our evaluations. Inspired by (Lee et al., 2023), five objective metrics were introduced to measure various types of the distance between the ground-truth audio and the generated audio, including: 1) Multi-resolution STFT error (M-STFT); 2) Perceptual evaluation of speech quality (PESQ); 3) Mel-cepstral distortion (MCD); 4) Periodicity error and 5) F1 score of voiced/unvoiced classification (V/UV F1). Please refer to (Lee et al., 2023) for more details on these metrics. The objective evaluation results, along with the 5-scale MOS results of different models, are presented in Table 2. For the objective evaluations, DFlow outperforms baseline models on most metrics except M-STFT, on which HiFi-GAN achieves the best. One possible reason for this is that, among these models, HiFi-GAN is the only one trained with a spectrogram loss. For subjective evaluation, we find that DFlow exhibits slightly better performance than HiFi-GAN but significantly better than WaveGlow and WaveNet. In particular, DFlow achieves a MOS of 4.42, with a gap of 0.11 to the ground truth, indicating its superiority.

5.3. Model size and efficiency

Table 2 presents a comparison of DFlow and the baseline models in terms of model size, computation cost, and synthesis speed. Despite having a large model size, DFlow exhibits significantly low computation cost, with nearly 10 times fewer operations (39.2 BFLOPs/second) compared to WaveGlow (368.36 BFLOPs/second). As a result, DFlow achieves a much faster inference speed, nearly 3 times faster, than WaveGlow on GPU. In fact, DFlow runs significantly faster than all the above likelihood models and is approaching HiFi-GAN. These improvements can be attributed to two factors. Firstly, DFlow requires only one generation step, which significantly boosts the model efficiency compared to AR or diffusion models. Secondly, the utilization of UNet layers in DFlow contributes to its overall efficiency, as they are more efficient and faster compared to the commonly used WaveNet layers in previous works (Prenger et al., 2019; Ping et al., 2020; Kong et al., 2021). In particular, DFlow can generate 22.05 kHz audio samples at a speed 71 times faster than real-time on GPU.

5.4. Ablation Study

To verify the effectiveness of each design choice of DFlow, we conducted a 7-scale CMOS test on ablation models of DFlow on the LJ-Speech dataset. We also trained a model with a standard NF structure for comparison. The detailed descriptions of these ablation models are shown in Appendix

Table 2. Objective and subjective metrics of DFlow and the baselines, evaluated on the LJ-Speech test set.

Models	**M-STFT (↓)**	**PESQ (↑)**	**MCD (↓)**	**Periodicity (↓)**	**V/UV (↑)**	**MOS (↑)**
Ground Truth	-	-	-	-	-	4.53±0.06
WaveNet (MOL)	1.18	2.32	1.91	0.132	0.91	4.08±0.09
WaveGlow	1.29	3.01	2.41	0.141	0.92	3.92±0.07
HiFi-GAN	**0.99**	3.11	0.92	0.152	0.94	4.39±0.08
DiffWave	1.23	3.08	2.58	**0.101**	0.94	4.31±0.07
DFlow	1.21	**3.43**	**0.71**	0.104	**0.95**	**4.42±0.06**

Table 3. The model size, synthesis speed, and memory usage of different models. All models were benchmarked using the same hardware. The FLOPs represent the number of Floating Points Operations for generating a one-second waveform. One Nvidia V100 GPU was used for the GPU benchmark. The time cost of transferring data between CPU and GPU was excluded from the GPU speed evaluation. For the CPU benchmark, an Intel(R) Xeon(R) Gold 6130 CPU server with 48 CPU cores was used.

Models	**# param**	**FLOPs/second (↓)**	**GPU RTF (↓)**	**CPU RTF (↓)**
WaveNet (MoL)	24.7M	53.3B	317.92	-
WaveGlow	87.7M	368.3B	0.045	3.4
HiFi-GAN	13.9M	**30.5B**	**0.008**	**0.13**
DiffWave	2.62M	28.7B×50	1.23	-
DFlow	105.3M	39.2B	0.014	0.26

Table 4. Ablation study on LJ-Speech dataset.

Model	**CMOS (↑)**
DFlow	0.0
w/o Noise Injection	-0.25
w/o AR Flow on f	-0.19
w/o Invertible Constraint on f	not converge
w/o VP Constraint on g	-0.18
Standard NF	-0.18

A.4. All the ablation models are trained for 500k iterations and the results are shown in Table 4. The absence of the invertible constraint on Auxiliary Flow leads to unstable training, resulting in the model's failure to converge. Other ablations all achieve substantially worse results than DFlow, demonstrating the significance of our design choices. Removing noise injection causes the most significant decrease in perceptual quality, and it performs even worse than the Standard NF model. This confirms the hypothesis we discussed in Section 3.3, that the absence of noise injection would result in DFlow falling behind a standard NF model. Removing the VP constraint on g also raises the reconstruction errors and therefore causes a decrease in model performance.

We also provide a detailed comparison of the test log-likelihood between various models with different implementations of invertible transformations in Appendix B.1. This additional comparison further confirms the effectiveness of our design choices.

5.5. Generalization on OOD samples

To evaluate the generalization ability of DFlow, we run a MOS test where DFlow and the baseline models are trained on the multi-speaker dataset VCTK. The baseline models in this experiment include a pretrained HiFi-GAN model and a WaveGlow model which we trained on VCTK using its default training configuration. We perform evaluations on both in-domain (IND) and out-of-domain (OOD) audio samples. The IND samples are from the test set of VCTK while the OOD samples are collected from external datasets. Specifically, we collected two OOD test sets, one containing speech waveforms of unseen languages from the Multilingual TEDx Corpus (Salesky et al., 2021), and another containing non-speech waveforms from MUSDB18-HQ (Rafii et al., 2017) and YouTube. As shown in Table 5, DFlow outperforms HiFi-GAN by a slight margin on the IND test set while exhibiting a notable performance improvement on the OOD test sets, especially on the non-speech OOD test set. The standard NF model, WaveGlow, performs poorly on both the IND and OOD test sets.

5.6. Manipulation of latent variables

We further demonstrate that DFlow inherits the latent variable manipulation capability from the standard NF, where DFlow also exhibits excellent performance. Results on low-temperature sampling and partial editing can be found in Appendix B.2.

Table 5. The 5-scale MOS results of different models trained on VCTK dataset, evaluated on both in-domain (IND) and out-of-domain (OOD) datasets.

Models	IND	OOD speech	OOD non-speech
Ground Truth	4.54±0.08	4.21±0.06	4.34±0.09
WaveGlow	4.01±0.11	3.71±0.09	3.24±0.09
HiFi-GAN	4.34±0.06	3.94±0.08	3.84±0.08
DFlow	**4.39±0.07**	**4.05±0.09**	**4.02±0.08**

6. DFlow with Large-scale Training

In previous experiments, we have showcased the generalization ability of DFlow in synthesizing OOD audio samples, indicating its potential as a universal neural vocoder (Jang et al., 2021; Lee et al., 2023). In this section, we further explore the capacity of DFlow with a large-scale training set and a large model size. We found that the current architecture of DFlow could be easily scaled up. We built our big model, denoted as DFlow-XL, by increasing the number of invertible layers in the Primary Flow and the number of the UNet layers in the Decoder, resulting in approximately doubling the number of parameters compared to DFlow. To evaluate the performance gain from expanding the model size, we also trained a moderate-sized version of DFlow, denoted as DFlow-L. The configurations of DFlow-L and DFlow-XL can be found in Appendix A.2. We use all training data (*train-clean-100*, *train-clean-360* and *train-other-500*, with a total length of 585 hours.) of the LibriTTS (Zen et al., 2019) dataset with the original sampling rate of 24 kHz for training. The best-performing universal neural vocoder BigVGAN (Lee et al., 2023) was chosen as our baseline. For a fair comparison, we strictly follow BigVGAN's mel-spectrogram settings: 100-band log-scale mel-spectrograms with a frequency range of [0, 12] kHz were extracted from the original audio clips with 1024 FFT size, 1024 window size, and 256 hop size. Both DFlow-L and DFlow-XL were trained on 8 Nvidia V100 (16G) GPUs for 2M steps, with a batch size of 16 per GPU with FP16. The segment size was 8192 and the initial learning rate was 1×10^{-4}. The official pretrained BigVGAN model from GitHub (Corporation, 2023), which was trained with *SnakeBeta* activation on the LibriTTS dataset for 5M training steps, was used for comparison. This version of BigVGAN was claimed by the authors to have the best performance.

Results for MOS evaluations are presented in Table 6. DFlow-XL achieves almost equal MOS to BigVGAN, indicating that DFlow-XL is among the best-performing universal vocoders. We further observe that for a dataset of this scale, we see a noticeable improvement in expanding the model capacity from DFlow-L to DFlow-XL, indicating that DFlow is easily scalable. We also report the detailed quantitative results of large models in Appendix B.3, where we showcase the superiority of training DFlow-XL compared to BigVGAN.

In addition to the above evaluations, we also compared DFlow-XL with a large version of HiFi-GAN with 114M model parameters (Lee et al., 2023), denoted as Big-HiFi-GAN. We directly collect the audio samples of Big-HiFi-GAN from BigVGAN's demo page for comparison (Sang-gil Lee, 2023). The results in Appendix B.4 indicate that although both DFlow-XL and Big-HiFi-GAN do not employ any periodic activate function as BigVGAN does, DFlow-XL outperforms Big-HiFi-GAN by a considerable margin. Note that the periodic activate functions such as *SnakeBeta* proposed by (Lee et al., 2023) can be easily adapted to DFlow as well, we leave this for future work.

Table 6. The 5-scale MOS results of different models trained on LibriTTS dataset, evaluated on both in-domain (IND) and out-of-domain (OOD) datasets.

Models	IND	OOD speech	OOD non-speech
Ground Truth	4.41±0.08	4.12±0.07	4.24±0.14
BigVGAN	**4.30±0.07**	3.99±0.09	4.04±0.11
DFlow-L	4.23±0.07	3.89±0.08	3.91±0.08
DFlow-XL	**4.30±0.08**	**4.01±0.08**	**4.06±0.11**

7. Conclusion

We introduce DFlow, a new generative model for high-fidelity waveform synthesis. Relative to existing models, DFlow has many advantages, including stable training, fast generation speed, a high level of robustness, and the ability to generate audio samples of high quality. We evaluated DFlow on two widely used speech datasets. The results demonstrate that DFlow establishes a new benchmark of both the model performance and efficiency among likelihood models. It even achieves better or at least comparative performance than GAN models. Moreover, we further tested DFlow's performance by increasing both the model size and dataset scale. The evaluations on OOD datasets demonstrate that our larger model, DFlow-XL, achieves highly competitive performance against the best universal vocoder, BigVGAN.

The combination of our results makes DFlow a very attractive approach for generative modeling, since it simultaneously has the properties of fast sampling, strong mode coverage, and sample quality, thereby addressing the well-known *generative learning trilemma* (Xiao et al., 2022) that exists in conventional generative approaches. Our findings suggest that DFlow holds great promise as a prospective avenue for future research in various generative tasks.

Impact Statement

Neural vocoders including DFlow, are widely adopted for tasks that require naturally synthesized audio, such as speech synthesis, music generation, and audio processing. These techniques have practical applications in fields like entertainment, virtual reality, and human-computer interaction. Therefore, utilizing neural vocoders with the potential to generate realistic and high-fidelity waveform generation can contribute to enhancing user experiences in these domains. Moreover, the fast synthesis speed of the proposed model would also be beneficial for service providers that provide real-time audio synthesis. However, because of the ability to synthesize realistic audio, especially speech, neural vocoders could be exploited for malicious purposes, such as generating realistic fake audio or impersonating individuals. It is crucial to consider the ethical implications and potential safeguards when developing and deploying such models.

References

Böhm, V. and Seljak, U. Probabilistic autoencoder. *Transactions on Machine Learning Research*, 2022.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

Chen, J., Lu, C., Chenli, B., Zhu, J., and Tian, T. VFlow: More expressive generative flows with variational data augmentation. In *ICML*, 2020.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. WaveGrad: Estimating gradients for waveform generation. In *ICLR*, 2021.

Corporation, N. NVIDIA/BigVGAN. https://github.com/NVIDIA/BigVGAN, 2023.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. In *ICLR*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *ICLR*, 2017.

Espuña I Fontcuberta, A. Analyzing the negative log-likelihood loss in generative modeling. Master's thesis, 2022.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked autoencoder for distribution estimation. In *ICML*, 2015.

Grcić, M., Grubišić, I., and Šegvić, S. Densely connected normalizing flows. *NeurIPS*, 2021.

Grover, A., Dhar, M., and Ermon, S. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D. M., Schulman, J., Amodei, D., and McCandlish, S. Scaling laws for autoregressive generative modeling. *CoRR*, 2020.

Higham, N. J. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning*, pp. 2722–2730. PMLR, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Horvat, C. and Pfister, J.-P. Denoising normalizing flow. *NeurIPS*, 2021.

Ito, K. The LJ speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

Jang, W., Lim, D., and Yoon, J. Universal MelGAN: A robust neural vocoder for high-fidelity waveform generation in multiple domains. In *ICASSP*, 2021.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., and Kavukcuoglu, K. Efficient neural audio synthesis. In *ICML*, 2018.

Kim, J., Kong, J., and Son, J. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, 2021.

Kim, S., Lee, S., Song, J., Kim, J., and Yoon, S. FloWaveNet: A generative flow for raw audio. In *ICML*, 2019.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 2016.

Kong, J. hifi-gan. https://github.com/jik876/hifi-gan, 2020.

Kong, J., Kim, J., and Bae, J. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In *NeurIPS*, 2020.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. DiffWave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brebisson, A., Bengio, Y., and Courville., A. MelGAN: Generative adversarial networks for conditional waveform synthesis. In *NeurIPS*, 2019.

Lanckriet, G. R., Ghaoui, L. E., Bhattacharyya, C., and Jordan, M. I. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3(Dec):555–582, 2002.

Lee, S., Kim, S., and Yoon, S. Nanoflow: Scalable normalizing flows with sublinear parameter complexity. *NeurIPS*, 2020.

Lee, S., Ping, W., Ginsburg, B., Catanzaro, B., and Yoon, S. BigVGAN: A universal neural vocoder with large-scale training. In *ICLR*, 2023.

Lee, W.-H., Ozger, M., Challita, U., and Sung, K. W. Noise learning-based denoising autoencoder. *IEEE Communications Letters*, 25(9):2983–2987, 2021.

LMNT. DiffWave. https://github.com/lmnt-com/diffwave, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

Luong, M. and Tran, V. Flowvocoder: A small footprint neural vocoder based normalizing flow for speech synthesis. In Ko, H. and Hansen, J. H. L. (eds.), *Interspeech*, 2022.

Miao, C., Chen, T., Chen, M., Ma, J., Wang, S., and Xiao, J. A compact transformer-based GAN vocoder. In *Interspeech*. ISCA, 2022.

Miao, C., Zhu, Q., Chen, M., Ma, J., Wang, S., and Xiao, J. EfficientTTS 2: Variational end-to-end text-to-speech synthesis and voice conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:1650–1661, 2024.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *NeurIPS*, 30, 2017.

Ping, W., Peng, K., Zhao, K., and Song, Z. Waveflow: A compact flow-based model for raw audio. In *ICML*, 2020.

Prenger, R., Valle, R., and Catanzaro, B. WaveGlow: A flow-based generative network for speech synthesis. In *ICASSP*, 2019.

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., and Bittner, R. The MUSDB18 corpus for music separation, 2017.

Ramasinghe, S., Fernando, K., Khan, S., and Barnes, N. Robust normalizing flows using Bernstein-type polynomials. In *BMVC*, 2022.

Salesky, E., Wiesner, M., Bremerman, J., Cattoni, R., Negri, M., Turchi, M., Oard, D. W., and Post, M. The multilingual tedx corpus for speech recognition and translation. In *Interspeech*. ISCA, 2021.

Sang-gil Lee. BigVGAN demo page. https://bigvgan-demo.github.io/, 2023.

Silvestri, G., Roos, D., and Ambrogioni, L. Deterministic training of generative autoencoders using invertible layers. In *ICLR*, 2023.

Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. 2016.

Valin, J.-M. and Skoglund, J. LPCNet: Improving neural speech synthesis through linear prediction. In *ICASSP*. IEEE, 2019.

Valle, R. NVIDIA/waveglow. https://github.com/NVIDIA/waveglow, 2018.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

Xiao, Z., Yan, Q., and Amit, Y. Generative latent flow. *arXiv preprint arXiv:1905.10485*, 2019.

Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion GANs. In *ICLR*, 2022.

Yamagishi, J., Veaux, C., and MacDonald, K. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92). University of Edinburgh. The Centre for Speech Technology Research (CSTR), 2019.

Yamamoto, R. wavenet_vocoder. https://github.com/r9y9/wavenet_vocoder/, 2018.

Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., and Xie, L. Multi-band MelGAN: Faster waveform generation for high-quality text-to-speech. In *Interspeech*, 2020.

Yoon, H.-W., Lee, S.-H., Noh, H.-R., and Lee, S.-W. Audio dequantization for high fidelity audio generation in flow-based neural vocoder. *InterSpeech*, 2020.

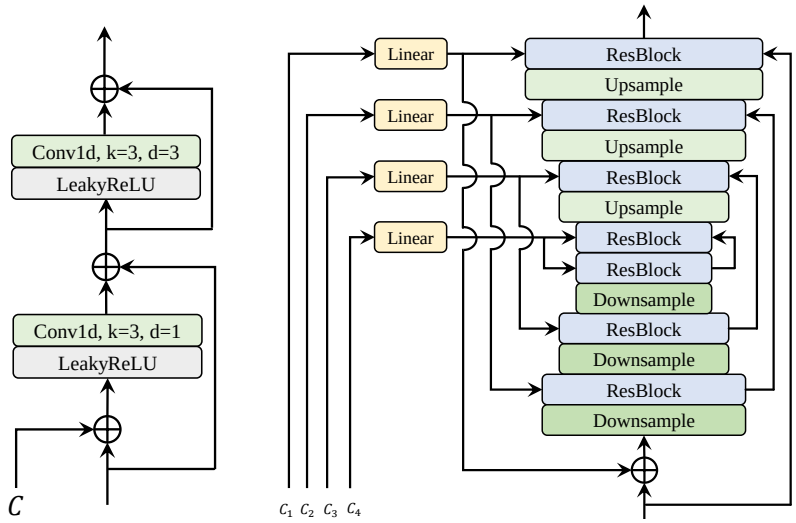
You, J., Kim, D., Nam, G., Hwang, G., and Chae, G. GAN vocoder: Multi-Resolution discriminator is all you need. In *Interspeech*, 2021.

Zen, H., Dang, V., Clark, R., Zhang, Y., Weiss, R. J., Jia, Y., Chen, Z., and Wu, Y. LibriTTS: A corpus derived from librispeech for text-to-speech. In *Interspeech*. ISCA, 2019.

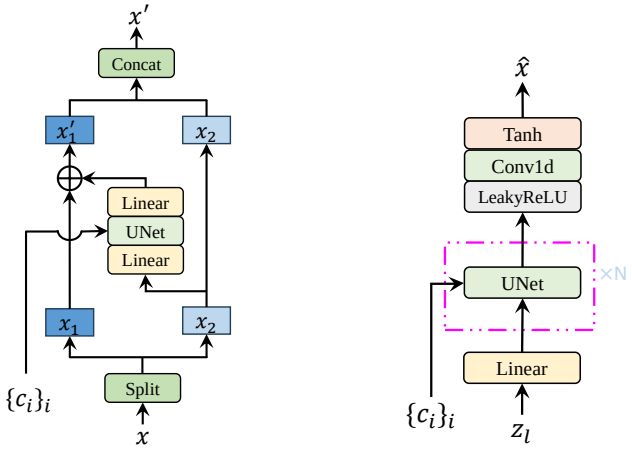
A. Model details

A.1. Module details

The detailed implementation of Residual Block, $\mathrm{UNet}(\cdot)$, Additive Coupling, and Decoder is shown in Fig. 3.



(a) Residual Block (b) UNet



(c) Additive Coupling (d) Decoder

Figure 3. Model details.

A.2. Model configurations and hyper-parameters

The detailed model configurations and hyper-parameters of DFlow, along with DFlow-L and DFlow -XL, are presented in Table 7.

A.3. MOS and CMOS evaluation details

To evaluate the audio quality of waveforms generated by different models, we use the 5-scaled Mean Opinion Score (MOS) as the subjective metric in Section 5.2, Section 5.5, Section 6, and Appendix B.4. There were 20 raters participating in the MOS test. Audio samples generated by each comparative model are randomly selected and provided to raters. Raters are

Table 7. Hyperparameters.

Modules	**Hyper-parameter**
Auxiliary Flow	InvertibleLayers = 8 (4 for left-to-right flow and 4 for right-to-left flow), WaveNetConvLayers = 8, WaveNetDilations = [1,2,4,8,16,32,64,128], WaveNetResidualchannels = 32, WaveNetGatechannels = 64,
Primary Flow	AddtiveCouplingLayers = 12 (**DFlow**), or 16 (**DFlow-L**), or 20 (**DFlow-XL**), SqueezeSize = 4, UNetChannels = [64,128,256,512,512,256,128,64], UNetUpsamplingScales = [4,4,4], UNetDownsamplingScales = [4,4,4],
Decoder	UNetLayers = 3 (**DFlow**), or 4 (**DFlow-L**), or 6 (**DFlow-XL**), UNetChannels = [32,64,128,256,512,512,256,128,64,32], UNetUpsamplingScales = [4,4,4,4], UNetDownsamplingScales = [4,4,4,4], LastConvKernelSize = 11,
Training parameters	β = 0.01 if TrainingStep <300k else 0.002

asked to rate the *audio quality* of audio samples on a 5-scale score (1 = Bad; 2 = Poor; 3 = Fair; 4 = Good; 5 = Excellent) with rating increments of 0.5. For each model, about 25 audio samples were selected for comparison, resulting in around 500 unique ratings per model. We filtered out scores from raters who scored the ground-truth samples as 3 or lower.

In Section 5.4, we conducted the 7-scale side-by-side Comparative Mean Opinion Score (CMOS) to examine our design choices. Twenty-five audio groups were provided to the same raters as in the MOS test. Each audio group contains the same audio sample generated by all different ablation models, together with the one generated by standard DFlow as the control item. Raters were asked to compare the *audio quality* of audio samples generated by ablation models to the control item on a 7-scaled score (-3 as significantly worse to 3 as significantly better) with a rating increments of 0.5.

A.4. Detail setting of ablation models

The CMOS evaluation reported in Table 4 is conducted on the ablation models of the following settings:

- *w/o Noise injection:* The model is trained on clean audio signal $\boldsymbol{x}$ rather the noisy one $\tilde{\boldsymbol{x}}$.
- *w/o AR flow on f:* The invertible AR transformations of Auxiliary Flow f are replaced with the same number of WaveGlow (Prenger et al., 2019) blocks, while the affine coupling layers of these blocks are formulated using the proposed UNet structure.
- *w/o Invertible constraint on f:* The Auxiliary Flow f is formulated using the same number of UNet layers, without the invertible constraint.
- *w/o VP constraint on g:* The additive coupling layers in the Primary Flow g are replaced with the non-volume-preserving affine coupling layers (Prenger et al., 2019)
- *Standard NF:* A standard NF model with WaveGlow blocks, with the affine coupling layers of these blocks formulated using the proposed UNet structure.

B. More results

B.1. Test log-likelihood

To further verify the effectiveness of our design details, we report comparisons of test log-likelihood between DFlow and its variants. For all these models, we remove the noisy injection and decoder m from the training and only maximize the log-likelihood of input $\boldsymbol{x}$. We consider the following models for comparison:

- *w/o UNet:* Replacing the UNet architecture with WaveNet layers with the same number of convolution layers.
- *w/o AR:* Replacing the AR transformations in the Auxiliary Flow f with the same number of WaveGlow blocks.
- *w/o bidirectional:* Replacing the bidirectional AR transformations in the Auxiliary Flow f with single directional transformations.
- *w/o $\boldsymbol{h}$:* Removing $\boldsymbol{h}$ in Equation 9.

In Table 8, we compare the likelihood of these relative models of DFlow at 500k training steps, which is not enough for these models to converge, but far enough to observe their performance differences. Switching from the UNet layer to the WaveNet layers decreases the performance most. Replacing the AR transformations in Auxillary Flow with NAR transformations also causes a significant decrease in log-likelihood.

Table 8. Comparison of likelihood under different settings.

Model	**Likelihood** (↑)
Base	5.14
w/o UNet	4.89
w/o AR	4.92
w/o bidirectional	5.08
w/o $\boldsymbol{h}$	5.07

B.2. Manipulating on latent variables

Low-temperature sampling. One interesting property of NF models is the ability to perform low-temperature sampling by decreasing the variance of the known prior $\boldsymbol{z}_p$ during the sampling phase. This decreases the diversity of the samples, however, increases the quality of each individual sample. Similar to standard NF models, DFlow inherits the ability of low-temperature sampling from NF and even performs better. In Fig 4, we present the mel-spectrogram visualizations of waveforms generated by WaveGlow and DFlow under different temperature factors t. We have the following observations:

- For both DFlow and WaveGlow, the harmonic components of the waveforms become more clear as the value of t decreases.
- At each value of t, DFlow consistently achieves significantly better harmonic quality compared to WaveGlow.
- WaveGlow exhibits a lack of robustness to different values of t. It encounters the periodic noise as the value of t decreases. In contrast, DFlow remains no periodic noise for various values of t.

These results indicate that DFlow could be a better model for low-temperature sampling, compared to standard NF models.

Partial editing. Although DFlow does not have an exact inverse as the standard NF models do, DFlow inherits the ability of partial editing from the standard NF models. In Fig. 5, we provide the wave plots of an audio sample (first row) along with its edited version (second row), while the unchanged part is colored in blue and the edited part is highlighted in red. We simultaneously performed duration change and resampling on the edited part. The manipulation is performed in the following steps:

- The original waveform $\boldsymbol{x}$ is inputted into the Auxillary Flow and then passed through the Primary Flow to obtain the latent variable $\boldsymbol{z}_p = g(f(\boldsymbol{x}))$.
- We randomly picked a length range as the target for editing, which, in this case, is approximately the last half of the original waveform. We denote the fixed part as $\boldsymbol{z}_{p1}$ (blue part in the first plot) and the target part as $\boldsymbol{z}_{p2}$ (red part in the first plot).
- We randomly sampled a vector $\boldsymbol{z}_{p2}^{'}$from the Gaussian distribution with a length of 1.2 times the length of $\boldsymbol{z}_{p2}$ and then replaced $\boldsymbol{z}_{p2}$ with $\boldsymbol{z}_{p2}^{'}$. The new latent variable $\boldsymbol{z}_{p}^{'}$ is obtained by concatenating $\boldsymbol{z}_{p1}$ and $\boldsymbol{z}_{p2}^{'}$.

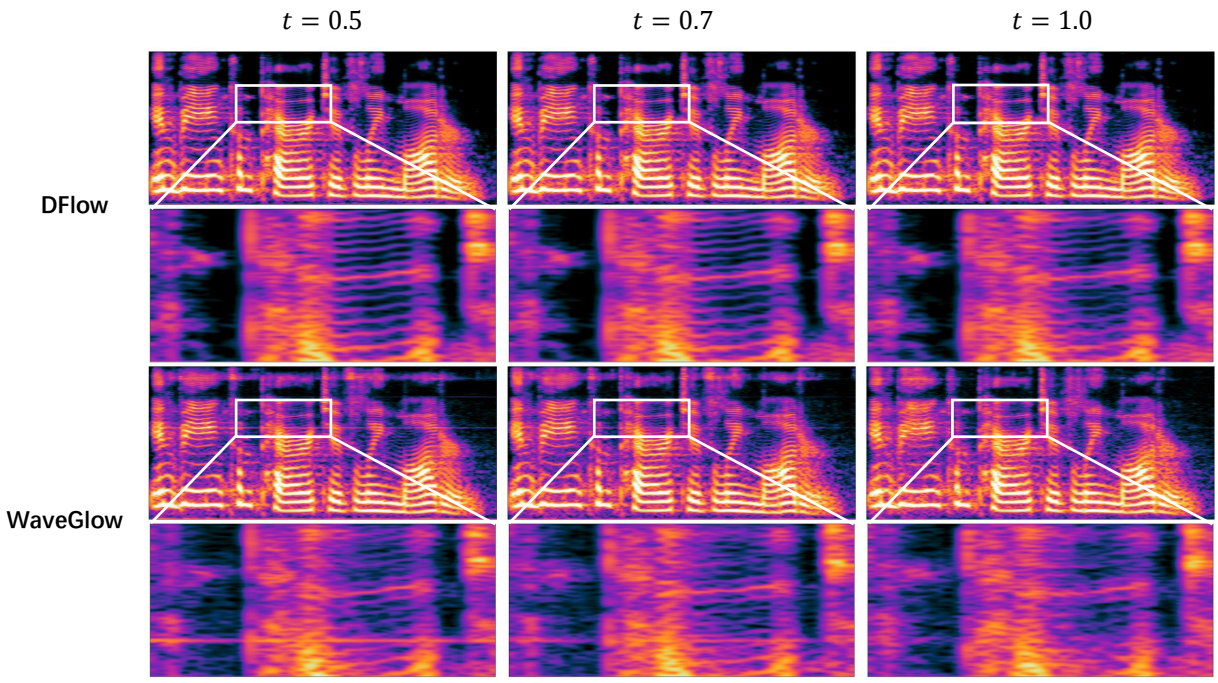


Figure 4. Mel-spectrogram visualizations of waveforms generated by WaveGlow and DFlow of different temperature factors, with a zoomed-in view of high-frequency details. The mel-spectrograms presented in the top row are generated by DFlow, whereas the mel-spectrograms in the bottom row are from WaveGlow.

- To match the length of $\boldsymbol{z}_p^{'}$, we also stretched the corresponding part of the mel-spectrogram of this audio by nearest neighbor interpolation.
- The new latent variable $\boldsymbol{z}_p^{'}$ is then sequentially inputted into the inverse of Primary Flow and the Decoder to produce the manipulated waveform $\boldsymbol{x}^{'} = m(g^{-1}(\boldsymbol{z}_p^{'}))$, with the stretched mel-spectrogram as a condition.

As can be seen in Fig. 5, though DFlow does not provide a theoretical inverse, the fixed part remains identical after reconstruction, with the phase information being highly preserved. For the target part, both duration and phase information are sufficiently changed after editing, but the edited part still conjuncts with the fixed part smoothly, indicating continuous phase changes across the two segments.

B.3. Quantitative results of large models

In Table 9, we list some quantitative results of our big models and BigVGAN. These results were obtained on a single GPU without using FP16 training. Here the batch size was set to 4 for each model. Although having a larger footprint, DFlow models have faster training speed and more efficient memory usage compared to BigVGAN.

Table 9. Model size, training speed, training memory usage, and synthesis speed of big models.

Models	**# param**	**Training Speed** (ms/step) (↓)	**Training Memory Use** (GB) (↓)	**GPU RTF (↓)**
BigVGAN	112M	932	10.6	0.036
DFlow-L	140M	654	6.5	0.024
DFlow-XL	183M	852	7.3	0.032

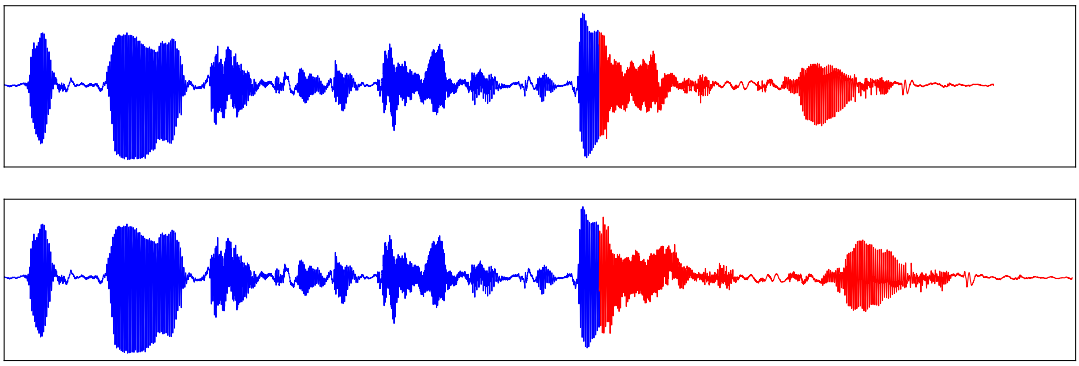


Figure 5. Waveform plots of an audio sample before and after partial editing.

B.4. Comparison with Big-HiFi-GAN

We also compared our big models with a 114M HiFi-GAN provided by BigVGAN, denoted as Big-HiFi-GAN, through a subjective test. Specifically, we downloaded both the ground truth audio and the sample audio of Big-HiFi-GAN from the demo page of BigVGAN (Sang-gil Lee, 2023). Then we synthesized the waveforms using our big models by conditioning on the same ground-truth mel-spectrograms Big-HiFi-GAN used. As can be seen, DFlow-L and DFlow-XL achieve significantly better quality than Big-HiFi-GAN.

Table 10. The MOS results of different big models.

Model	**MOS** (↑)
Ground Truth	4.51 ± 0.12
Big-HiFi-GAN	4.11 ± 0.11
DFlow-L	4.34 ± 0.13
DFlow-XL	$\mathbf{4.41 \pm 0.11}$