WaveODE: An Efficient Vocoder Based on Probability Flow Equation

Anonymous ACL submission

Abstract

Probability flow based models for image and audio synthesis, such as denoising diffusion probabilistic models and Poisson flow generative models, can be interpreted as modeling any ground truth distribution through the non-compressible fluid partial differential equation, where the initial and final fluid density are the chosen prior distribution and the ground truth distribution correspondingly. In this research, we analyse various previous models under the unified perspective of probability flow equation, and propose WaveODE model for mel-spectrogram conditioned speech synthesis task, which learns a velocity field under a dynamically changing probability flow equation instead of estimating the groud truth with a fixed evolution equation such as VP-SDE and sub-VP-SDE in previous works. Since mel-spectrogram is a relatively strong condition which limits the possible audios to a small range, waveODE models the ground truth distribution with a mel-conditioned prior distribution rather than the standard Gaussian distribution, and adopts a distillation method to accelerate the inference process. Experimental results show that our model is more competitive with previous vocoders in sample quality, and could generate waveform with a single inference step.[1]

1 Introduction

Recent advancements in training algorithms and network architectures have facilitated the production of high-fidelity audio by deep generative models in the realm of speech synthesis (Kumar et al., 2019; Kong et al., 2020b; Lam et al., 2022; Huang et al., 2023; Ye et al., 2023). The pioneering implementation of a deep generative model involved the autoregressive generation of waveforms from mel-spectrograms (Oord et al., 2016; Kalchbrenner et al., 2018), which yielded high-fidelity audio but was hindered by a significantly slow inference speed. To overcome this limitation and achieve real-time high-fidelity speech synthesis, a multitude of non-autoregressive models have been proposed recently. These models can be broadly categorized into three types: flow-based models, generative adversarial networks, and diffusion probabilistic models.

Flow-based models generate waveforms from a chosen prior distribution, such as the Gaussian distribution,utilizing invertible neural networks to estimate the log-likelihood for training. (Ping et al., 2020; Prenger et al., 2019). These models require the preservation of invertibility and the evaluation of the determinant for training, which is accomplished through the employment of intricately designed neural networks. However, this design constrains the model's flexibility and restricts the quality of the audio output.Chen et al. (2018) proposed continuous normalizing flow(CRF) models, which use instantaneous change of variable formula to estimate the variation of log probability density, and generate data through a neural ordinary differential equation(ODE). Due to the Picard existence-uniqueness theorem, the neural network which provides the velocity function in continuous normalizing flow models could be non-invertible, and this improves the models' performance(Grathwohl et al., 2018). For mel-conditioned speech synthesis task, CRF models were hindered by the slow training process and the large variance of trace estimator (Wu and Ling, 2020; Kim et al., 2020).

Generative Adversarial Networks (GANs) provide greater flexibility than flow-based models and can efficiently generate waveforms of superior fidelity (Kumar et al., 2019; Kong et al., 2020a; Kim et al., 2021; Jang et al., 2021; Lee et al., 2022). The success of these models can be attributed to the large receptive fields of the generators and the

[1]Audio samples and codes are available at a newly registered anonymous repository: https://github.com/JBJWZZHCDS/WaveODE

discriminators' capacity to identify noises of varying scales and periods. Specifically, Kumar et al. (2019) proposed multi-scale discriminators, while Kong et al. (2020a) introduced a multi-receptive field (MRF) generator and multi-period discriminators, significantly enhancing the model's performance.

Denoising diffusion probabilistic models (DDPMs), which employ a Markov chain to transform a known prior distribution into a complex ground truth distribution, have became very popular choices recently. (Kong et al., 2020b; Lam et al., 2022; Huang et al., 2023).

These models utilize a noise-adding diffusion process without learnable parameters to obtain the training data for the denoising generator, eliminating the need for additional networks such as discriminators or autodecoders during training. However, the inference process using diffusion models is typically time-consuming. To address this, Kong et al. (2020b), Lam et al. (2022), and Huang et al. (2023) have proposed several different approximate fast-sampling algorithms that can generate waveforms efficiently, albeit with a slight reduction in sample quality.

In this study, we initially conduct a review of continuous normalizing flow models,diffusion probabilistic models, Poisson flow generative models (Xu et al., 2022) and rectified flow models (Liu et al., 2022) under a unified perspective of a non-compressible passive fluid partial differential equation. This equation has boundary conditions at $t = 0$ and $t = 1$, which are equivalent to a known prior distribution distribution and the ground truth distribution respectively. These models employ different methods to parameterize the probability flow equation, which has significant influence on their different speech synthesis performance.

Additionally, we propose the WaveODE model for the mel-spectrogram conditioned speech synthesis task, which could learn a velocity field directly and efficiently. It is noteworthy that the mel-spectrogram is a relatively strong condition, choosing a advisable conditioned distribution could speed up the inference process to a great extent. With our mel-condition prior distribution,WaveODE could generate waveform in only one inference step.The similarity mean opinion score (SMOS) test results indicate that WaveODE is on par with previous flow-based models and diffusion models in terms of sample quality and efficiency.

2 Backgrounds on Mathematics and Related Works

2.1 Non-Compressible Fluid Equation

Equation (1) shows the non-compressible fluid equation in physics, where $n(\boldsymbol{x}, t)$ is the fluid density function, $v(\boldsymbol{x}, t)$ is the velocity field, and $s(\boldsymbol{x}, t)$ is the source function.Without loss of generality, we assume the time range is $[0, 1]$.

$$\frac{\partial n}{\partial t}(\boldsymbol{x}, t) + \nabla_{\boldsymbol{x}}(n(\boldsymbol{x}, t)\boldsymbol{v}(\boldsymbol{x}, t)) - s(\boldsymbol{x}, t) = 0 \quad (1)$$

This equation discribes how the fluid's density at situation $\boldsymbol{x}$ varies from initial time $t = 0$ to finishing time $t = 1$ according to a velocity field $\boldsymbol{v}(\boldsymbol{x}, t)$ and a source function $s(\boldsymbol{x}, t)$. It only requires the fluid be non-compressile and move without friction, thus there exist infinite many specific functions $n(\boldsymbol{x}, t)$, $\boldsymbol{v}(\boldsymbol{x}, t)$, $s(\boldsymbol{x}, t)$ that could solve it.

For speech synthesis task, we primarily concern the case where the fluid density function $n(\boldsymbol{x}, t)$ is a probability density function, and we denoted it as $p(\boldsymbol{x}, t)$. Now the source term $s(\boldsymbol{x}, t)$ could be regarded as the birth (or death) probability of fluid particles at situation $\boldsymbol{x}$ and time t. If we can sample from the initial probability distribution, and both the velocity and source functions are given, we can numerically simulate the fluid with birth-death process(Lu et al., 2019).However, we don't want the costly birth-death simulation in the inference process, i.e. we focus on the probability flow equation(2), where $t \in [0, 1]$.

$$\frac{\partial p}{\partial t}(\boldsymbol{x}, t) + \nabla_{\boldsymbol{x}}(p(\boldsymbol{x}, t)\boldsymbol{v}(\boldsymbol{x}, t)) = 0 \quad (2)$$

For this equation, we can easily sample from $p(\boldsymbol{x}, t)$ if we can sample from $p(\boldsymbol{x}, 0)$ and the velocity function $\boldsymbol{v}(\boldsymbol{x}, t)$ is given. To realize this, we only need to sample from the initial distribution and solve the initial value problem (IVP) of the ordinary differential equation(3), which could be efficiently completed by ODE solvers.

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{v}(\boldsymbol{x}, t) \quad (3)$$

Additionally, for generative models, sample from prior distribution and then solve the IVP of velocity ODE is not enough, the key is to make sure that the probability density become the ground truth distribution at $t = 1$, i.e. $p(\boldsymbol{x}, t), \boldsymbol{v}(\boldsymbol{x}, t)$ indeed solve the boundary value problem (BVP) of probability flow equation where $p(\boldsymbol{x}, t)$ equals to

ground truth distribution and a chosen prior distribution at time $t = 0$ and $t = 1$ correspondingly, different models make different efforts to ensure the BVP is correctly solved, this is the key ingredient of ODE based generative models.

2.2 Mathematical Tools

Now we introduce some mathematical tools with regard to the IVP and BVP.

2.2.1 Instantaneous Change of Variables

This formula is usefull to determine the velocity field when two boundary distributions are given and its proof could be found in (Chen et al., 2018). Let $\boldsymbol{x}(t)$ be a finite continuous random variable with probability density $p(\boldsymbol{x}(t))$ dependent on time. Let $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{dt}} = \boldsymbol{v}(\boldsymbol{x}(t), t)$ be a differential equation describing a continuous-in-time transformation of $\boldsymbol{x}(t)$. Assuming that $\boldsymbol{v}$ is uniformly Lipschitz continuous in $\boldsymbol{x}$ and continuous in t, then the change in log probability density also follows a differential equation:

$$\frac{\mathrm{d}\log p(\boldsymbol{x}(t))}{\mathrm{d}t} = -tr\left(\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right) \tag{4}$$

2.2.2 Expectation and Velocity

This subsection shows that if the probability density function $p(\boldsymbol{x}, t)$ and the velocity function $\boldsymbol{v}(\boldsymbol{x}, t)$ solve the probability flow equation, there is a helpful relationship between them. Let $\boldsymbol{x}(t)$ be a continuously differentiable random process on $t \in [0, 1]$, if the conditional expectation $\mathrm{E}[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{dt}}|\boldsymbol{x}(t)]$ is locally Lipschitz and $p(\boldsymbol{x}, t), \boldsymbol{v}(\boldsymbol{x}, t)$ solve the probability flow equation, then

$$\boldsymbol{v}(\boldsymbol{x}, t) = \mathrm{E}\left[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{dt}}\middle|\boldsymbol{x}(t)\right] \text{ almost surely,} \tag{5}$$

hence this velocity IVP at time $t = 0$ and $t = 1$ become the bridge between the initial and finishing distribution at time $t = 0$ and $t = 1$ of $\boldsymbol{x}$.The proof could be found in Appendix A.

2.2.3 Green's Function Method

Linear partial differential equations (PDEs) with a single function variable $\varphi(\boldsymbol{x}, t)$, could be solved by Green's function method.[2] Although in probability flow equation we have two function variable $p(\boldsymbol{x}, t)$ and $\boldsymbol{v}(\boldsymbol{x}, t)$, but we can consider them as functions of $\varphi(\boldsymbol{x}, t)$. Utilizing the ground truth distribution and a Green's function solution to $\varphi(\boldsymbol{x}, t)$, along with the boundary condition at $t = 1$, the analytical forms of $p(\boldsymbol{x}, t)$ and $\boldsymbol{v}(\boldsymbol{x}, t)$ can be readily derived.Consequently, the velocity field can be trained efficiently. Figuratively speaking, Green's function method allow us to build the velocity field through point charges given by the data at $t = 1$. For example, the Gaussian perturbation kernels in diffusion probabilistic models can be interpreted as the Green's function to the diffusion equation. Detailed examples are provided in Appendix B and Appendix C.

[2]Green's function method: https://en.wikipedia.org/wiki/Green%27s_function

2.3 Continuous Normalizing Flow models

CRF models choose a prior distribution that could be analytically discribed like Gaussian distribution, and then solve the velocity IVP at $t = 1$ to evaluate the log probability density at time $t = 1$. Then during maximal likelihood training, the finishing distribution gradually becomes the ground truth distribution and the velocity function could solve the BVP problem as we want. However, solving the velocity IVP during training with adjoint sensitive method mentioned in (Grathwohl et al., 2018) leads to extremely slow training speed, and CRF vocoders are also hindered by the large variance of trace estimator, which leads to an inferior sample quality.(Wu and Ling, 2020; Kim et al., 2020)

2.4 Score Based Generative Models and Rectified Flow Models

Models for audio synthesis, such as DiffWave, ProDiff (Huang et al., 2022b), and FastDiff, primarily focus on data scoring. Song et al. (2020) have successfully unified Noise Conditional Score Networks (Song and Ermon, 2019) and the Denoising Diffusion Probabilistic Model (Ho et al., 2020) under the umbrella of stochastic differential equations (SDEs). This unification is exemplified in the forward diffusion SDE, which is as equation (6).

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w} \tag{6}$$

And the corresponding backward SDE for generating procress is shown in equation (7).

$$\mathrm{d}\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, t)]\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w} \tag{7}$$

Rectified flow models (Liu et al., 2022) and score based generative models build the velocity function through the expectation property 2.2.2. Rectified flow models choose random process

$$\boldsymbol{x}(t) = t\boldsymbol{x}(1) + (1 - t)\boldsymbol{x}(0), t \in [0, 1] \tag{8}$$

where $\boldsymbol{x}(1), \boldsymbol{x}(0)$ represents the groud truth distribution and the chosen prior distribution, and they are independent sampled. This parameterization for $p(\boldsymbol{x}, t)$ is fixed when the two distributions are given, and it directly satisfy the two boundary condition. Hence the optimization goal for velocity function could be determined by

$$\boldsymbol{v}(\boldsymbol{x}, t) = \mathrm{E}[\boldsymbol{x}(1) - \boldsymbol{x}(0) | \boldsymbol{x}(t)] \tag{9}$$

$$\Leftrightarrow \min_{\boldsymbol{v}} \mathrm{E}[||\boldsymbol{x}(1) - \boldsymbol{x}(0) - \boldsymbol{v}(\boldsymbol{x}(t), t)||^2] \tag{10}$$

Using the expectation property we can just sample from two marginal distribution and learn its corresponding velocity function without solving IVP in the training process. Score based generative models could also be written in similar form, we take VP-SDE as an example. Song et al. (2020) have already derived the pertubation kernel of VP-SDE in their Appendix B, we coule rewrite the result as

$$\boldsymbol{x}(t) = \alpha(t)\boldsymbol{x}(1) + \beta(t)\boldsymbol{x}(0), t \in [0, 1] \tag{11}$$

$$\alpha(t) = exp(-\frac{1}{4}a(1-t)^2 - \frac{1}{2}b(1-t)), \tag{12}$$

$$\beta(t) = \sqrt{1 - \alpha(t)^2} \tag{13}$$

where the coefficient a is given by DDPM's hyperparameters $N(\beta_{max} - \beta_{min}) = 1000(0.02 - 0.0001) = 19.9$ and b is given by $N\beta_{min} = 1000 \times 0.0001 = 0.1$. Similar to rectified flow models, we could find a simple optimization goal for the velocity field with the expectation property:

$$\min_{\boldsymbol{v}} \mathrm{E}[||\alpha'(t)\boldsymbol{x}(1) + \beta'(t)\boldsymbol{x}(0) - \boldsymbol{v}(\boldsymbol{x}(t), t)||^2] \tag{14}$$

It worth mentioning that VP-SDE doesn't solve the BVP precisely at $t = 1$, and $\beta'(t)$ doesn't exist at $t = 1$, thus we should train it in $[0, 1 - \epsilon]$.

2.5 Poisson Flow Generative Model

Poisson Flow Generative Models (PFGMs) (Xu et al., 2022) are proficient visual generative models that exhibit comparable efficiency to score-based models. These models generate samples from the ground truth distribution by utilizing high-dimensional electric fields, which are solutions to the Poisson partial differential equation. To circumvent the issue of mode collapse, the original data is augmented with an additional dimension. The prior distribution is then defined as a uniform distribution on the surface of the superballs. It is noteworthy that this augmented dimension can be interpreted as time, thereby suggesting that PFGMs are essentially modeling a time-dependent Poisson equation as equation 15, where $\varphi(\boldsymbol{x}, t)$ is the electricity potential function.

$$\frac{\partial^2 \varphi}{\partial t^2}(\boldsymbol{x}, t) + \nabla_{\boldsymbol{x}}^2 \varphi(\boldsymbol{x}, t) = 0, \tag{15}$$

This equation cannot be directly interpreted as a probability flow equation. To derive an appropriate equation, the selection could be made by:

$$p(\boldsymbol{x}, t) = \frac{\partial \varphi}{\partial t}(\boldsymbol{x}, t), \boldsymbol{v}(\boldsymbol{x}, t) = \frac{\nabla_{\boldsymbol{x}} \varphi(\boldsymbol{x}, t)}{\frac{\partial \varphi}{\partial t}(\boldsymbol{x}, t)}, \tag{16}$$

Thus, the boundary condition at $t = 1$ becomes:

$$p(\boldsymbol{x}, 1) = \frac{\partial \varphi}{\partial t}(\boldsymbol{x}, 1) = p_{\text{data}}(\boldsymbol{x}) \tag{17}$$

Presently, PFGMs are translated into a fluid equation, and training data can be generated subsequent to the resolution of this linear PDE. Further details regarding the Green's function solution to this equation, as well as the training process of PFGMs from the perspective of fluid equations, are deferred to Appendix D.

What's more, score based generative models could also be viewed as solving a linear PDE with Green's function method, since Kolmogorov forward equation, also known as the Fokker-Planck Equation [3], can be reformulated into a probability flow equation (8) with same $p(\boldsymbol{x}, t)$, and a detailed proof of this transformation is provided in Appendix B.

$$\frac{\partial p}{\partial t}(\boldsymbol{x}, t) + p(\boldsymbol{x}, t)\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, t) + \boldsymbol{f}(\boldsymbol{x}, t) \cdot \nabla_{\boldsymbol{x}}(p(\boldsymbol{x}, t)) - \frac{1}{2} g^2(t) \nabla_{\boldsymbol{x}}^2 p(\boldsymbol{x}, t) = 0 \tag{18}$$

$$\boldsymbol{v}(\boldsymbol{x}, t) = [\boldsymbol{f}(\boldsymbol{x}, t) - \frac{1}{2} g^2(t) \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, t)]. \tag{19}$$

It's also a probability flow equation where velocity function is given by equation (9), and the learning process of score based models could be regarded as fitting the analytical solution of the velocity function given by Green's function method. We will solve a special case $\frac{\partial p}{\partial t}(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}}^2 p(\boldsymbol{x}, t) = 0$ in Appendix C using Fourier Transformation to demonstrate this issue.

[3] Kolmogorov forward Equation or Fokker-Plank Equation: https://en.wikipedia.org/wiki/Fokker%E2%80%93Planck_equation

3 Methods

3.1 Overview and Motivation

With the mathematical tools mentioned above, we have a comparatively comprehensive understanding of how to make the probability density function and velocity function solve the BVP of probability flow equation and then generate the data by solving the velocity IVP. Now we consider the specific mel-conditioned speech synthesis task.

To our knowledge, there already exist rectified flow models which could generate acceptable mel-spectrograms from text in one step (Guan et al., 2023; Guo et al., 2023), but there is no probability flow based model which could generate waveforms from mel-spectrograms with acceptable quality in one step, thus we mainly focus on how to generate high-quality audio in one step.

Consider the three mathematical tools and their corresponding generative models in section 2, continuous normalizing flow models based on instantaneous change of varaible formula need to solve velocity IVP during training, thus there is inherent difficulty for generating high-quality waveform in one step. As for the methods based on Green's function method, we need to construct a good Green's function to make the velocity field be straight enough, which is not obvious in mathematics, actually PFGMs and DDPMs cannot generate accecptable data in one step. Hence, we mainly utilize the expectation property of velocity function.

3.2 Mel-Conditioned Prior Distribution

If the initial and finishing boundary distribution are almost the same, then obviously the one step generation task would be much easier, so we should take full advantage of the mel-spectrogram to provide a prior distribution that is close to the audio distribution. This thought has already been applied to diffusion models (Lee et al., 2021; Koizumi et al., 2022), but these prior distributions are not close to the audio distribution due to the need of stablizing diffusion training objectives. For example, Lee et al. (2021) use $N(\mu, \mathbf{\Sigma})$ as the diffusion prior distribution, and their training objective is:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}(\boldsymbol{x}_0 - \mu) + \sqrt{1 - \bar{\alpha}_t^2}\epsilon, \quad (20)$$

$$\min_{\epsilon}(\epsilon - \epsilon_\theta(\boldsymbol{x}_t, t))^\top \mathbf{\Sigma}^{-1}(\epsilon - \epsilon_\theta(\boldsymbol{x}_t, t)), \quad (21)$$

where $\boldsymbol{x}_0 \sim p_{data}, \epsilon \sim N(\mathbf{0}, \mathbf{\Sigma})$. They choose $\mu = 0$ and $\mathbf{\Sigma}$ to be a diagnoal matrix given by mel-spectrogram. However, to stable the training process, they have to clamp the diagnoal Σ's value between 0.1 and 1, which enlarges the distance between the prior distribution and audio distribution.

According to the training target given by the expectation property, we can directly sample from two boundary distribution without giving their distribution analytically, which means WaveODE could adopt a prior distribution with much more smaller variance. In implementation, we also choose $N(\mathbf{0}, \mathbf{\Sigma})$ as the prior distribution where $\mathbf{\Sigma}$ is diagnoal. We use torchaudio.transforms. MelSpectrogram to gain mel-spectrograms from audios, whose value is ranged in $[0, 32768.0]$. (We take logarithm mel-spectrogram for neural networks.) Since mel-spectrogram records the energy of voice, the square root of their sum at frequency dimension is a good choose for the standard deviation of prior distribution, we divide it by $\sqrt{\text{mel-bands} \times 32768.0}$ to norm it into $[0, 1]$ and repeat a value at time dimension for hopsize time to align its shape with audios. Since the value in a mel-spectrogram is usually far samller than the possible maximal value, the standard deviation could even reach the level of 10^{-4} (at almost silence part).

This mel-conditioned prior distribution improve the inference speed and sample quality to a great extend, and also helps WaveODE generalize better on unseen out of distribution datasets. Experimental evidence could be found in next section.

3.3 Training Objective

Now the initial and finishing distribution are known, we need to construct a continuously differentiable random process $\boldsymbol{x}(t)$ to train the velocity function. Rectified flow is good choice for general generating task, we adopt an parameterized $\boldsymbol{x}(t)$ near it:

$$\begin{aligned}\boldsymbol{x}(t) &= t\boldsymbol{x}(1) + (1-t)\boldsymbol{x}(0) \\ &+ t(1-t)\boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t)\end{aligned} \quad (22)$$

$$\begin{aligned}\boldsymbol{x}'(t) =& \boldsymbol{x}(1) - \boldsymbol{x}(0) + (1-2t) f_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t) \\ &+ t(1-t)\frac{\partial \boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t)}{\partial t}\end{aligned} \quad (23)$$

$$\min_{\boldsymbol{v}} E[||\boldsymbol{x}'(t) - \boldsymbol{v}_\theta(\boldsymbol{x}(t), t)||^2] \quad (24)$$

This means $p(\boldsymbol{x}, t)$ is dynamically changing during the training process, which allows us control the evolution equation through adding loss functions to $\boldsymbol{x}(t)$. The two functions are approximated by two U-Nets, and mel-spectrogram is also fed to

them as additional condition input. Besides, the partial derivative of $\boldsymbol{f}$ cannot be calculated directly by autograd, we use

$$\frac{\boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t+\epsilon) - \boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t-\epsilon)}{2\epsilon} \tag{25}$$

to estimate the derivative($\epsilon = 5 \times 10^{-4}$). In rectified flow models $\boldsymbol{v}$ is trained by fitting $\boldsymbol{x}'(t)$, now in WaveODE $\boldsymbol{x}'(t)$ is also trainable, which could reduce the training error. Apart from the velocity loss, we also add loss functions to $\boldsymbol{f}$ to regulate the evolution of random variable $\boldsymbol{x}$. Since we want the velocity IVP trajectory be straight and reach it ends as soon as possible, we add two loss functions to achieve the goals, where $t \sim U[0, 1]$.

$$E[||\boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t)||^2], E[||\boldsymbol{x}(t) - \boldsymbol{x}(1)||^2] \tag{26}$$

The first loss avoid the trajectory become too curly and the second loss helps the trajectory reach its end faster. This parameterization may be unhelpful for general generative models, but experimental resuilts show that it could improve the quality for one-step generated waveforms.

3.4 Networks and Distillation

The velocity function is predicated on a U-Net model, we adopt the multi-receptive field (MRF) module in the generator of Hifi-GAN, and they are denoted as ResLayers in Figure 1. The U-net is simple in structure, and could be regarded as the combination of two Hifi-GAN generator and we just change the upsampling transposed convolution 1D layers into strided downsampling convlution 1D layers in the down way, and add the corresponding feature maps to the up way. As for the time term, we follow (Vaswani et al., 2017) to embed the time $t \in [0, 1]$ into an 128-dimensional positional encoding vector, and we multiply t by 100 to keep its magnitude be the same as diffusion models. Time embeddings are added to ResLayers aftering DNN's processing.

$$\begin{aligned}[sin(10^{\frac{0\times4}{63}}100t), \cdots, sin(10^{\frac{63\times4}{63}}100t),\\ cos(10^{\frac{0\times4}{63}}100t), \cdots, cos(10^{\frac{63\times4}{63}}100t)]\end{aligned} \tag{27}$$

The function $\boldsymbol{f}$, we call it mixer since it produce the middle state between $\boldsymbol{x}(0)$ and $\boldsymbol{x}(1)$, it share the same structure with the velocity function, we just multiply with processed time embeddings to $\boldsymbol{x}(0)$ and $\boldsymbol{x}(1)$ and sum them up as $\boldsymbol{x}$ in velocity function.

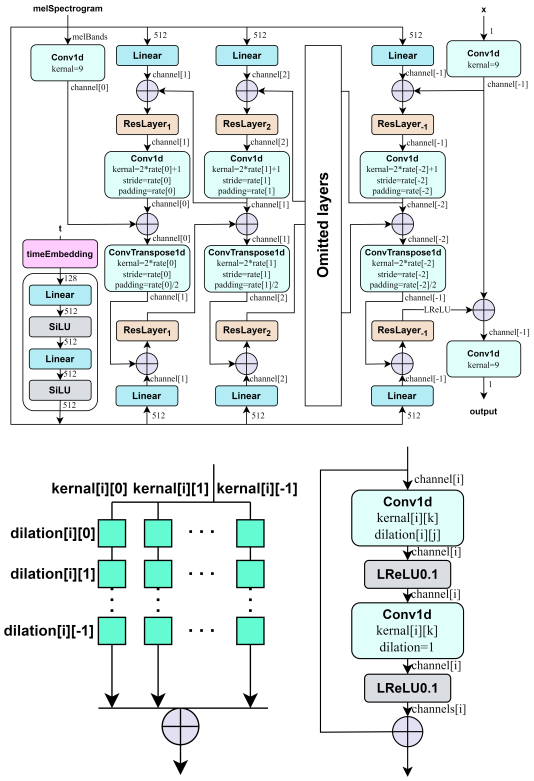

Figure 1: The neural networkssstructure

As for distillation process, We simply train a copy of a trained velocity function through Euler discretilization, where the answer is given by the trained velocity field and Runge-Kutta ODE solvers.The evaluation time $\in [0, 1]$ could also be trained by gradient descent in the same time.

3.5 Training Algorithm

The training procedures for the proposed WaveODE model are summaried in Algorithm 1 (in next page).

4 Experiments

4.1 Datasets

In order to ensure a fair and reproducible comparison against other competing methodologies, we utilize the LibriTTS dataset (Zen et al., 2019) , which is a large-scale corpus of read English speech amounting to 1,000 hours, and comprises more than 350,000 audio clips of 24,000 Hz from mutiple speakers. We feed all training data including train-clean-100, train-clean-360 and train-other-500 to all of the models. As for our mel-spectrogram dataset, we use 100-bands mel-spectrogram and set the frequency range be [0,12] kHz. The FFT size, Hann window size, hop size of mel-spetrogram are set to 1024,1024 and 256 correspondingly.

Algorithm 1 Train WaveODE

Input: Mixer $\boldsymbol{f}$, velocity field $\boldsymbol{v}$, mel condition c, time step $t \sim U[0,1]$, $\epsilon = 5 \times 10^{-4}$

repeat

Sample $\boldsymbol{x}(1) \sim p_{\text{data}}(\boldsymbol{x}|c)$,

Sample $\boldsymbol{x}(0) \sim \mathbf{N}(\mathbf{0}, \text{prior}(c))$

$\boldsymbol{f}_0 = \boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t, c)$

$\boldsymbol{f}_+ = \boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t+\epsilon, c)$

$\boldsymbol{f}_- = \boldsymbol{f}_\theta(\boldsymbol{x}(0), \boldsymbol{x}(1), t-\epsilon, c)$

$\boldsymbol{x}(t) = t\boldsymbol{x}(1) + (1-t)\boldsymbol{x}(0) + t(1-t)\boldsymbol{f}_0$

$\boldsymbol{v}_0 = \boldsymbol{v}_\theta(\boldsymbol{x}(t), t, c)$

$\boldsymbol{x}'(t) = \boldsymbol{x}(1) - \boldsymbol{x}(0) + (1-2t)\boldsymbol{f}_0$
$+t(1-t)\frac{1}{2\epsilon}(\boldsymbol{f}_+ - \boldsymbol{f}_-)$

Loss$= ||\boldsymbol{v}_0 - \boldsymbol{x}'(t)||^2 + ||f_0||^2$
$+||\boldsymbol{x}(t) - \boldsymbol{x}(1)||^2$

Take gradient descent according to loss.

until WaveODE converged

To assess the model's generalization capabilities in out of distribution scenarios, we employ the MUSDB18-HQ music dataset (Rafii et al., 2017). It is a multi-track musical dataset which contains the original mixture audios and four splitted tracks: vocal, drums, bass and other instruments. Synthesising the languages that are not inside the training dataset is also in our consideration. The Multilingual TEDx Corpus (Salesky et al., 2021) contains the audios of TEDx talks in Spanish,Italian, French and Portuguese; The CN-Celeb dataset (Fan et al., 2020) is a large-scale speaker recognition dataset collected in the wild. The dataset contains more than 126,000 utterances from 997 Chinese celebrities, and covers 11 different genres in real world; The deeply Korean read speech corpus dataset(Deeply, 2021) contains short speech audio clips in Korean, and the clips are recorded in three types of different environments.

4.2 Training and Evaluation Metrics

The detailed architectures and configurations of the models are listed in Appendix E. As for the traning process,the model is trained on a single Nvidia RTX 4090 GPU with a initial learning rate 2×10^{-4} and batch size 16 for the mixer and velocity function. The learning rate would decay by 0.997 times every 1000 steps, and the training process includes 1M steps in total. We use AdamW optimizer for training, where the betas are set to (0.9,0.98) and the weight decay rate is set to 0.01. The distillation process contains 50000 steps in total, and we adjust the learning rate to 2×10^{-5}. For multi-step distillation, the time scheduler is also learnable, we set its learning rate to 0.01.

Since there are out of distribution data need to be tested, the subjective evaluation of audios' quality is conducted through 5-scale Similarity Mean Opinion Score (SMOS) tests mentioned in BigVGAN (Lee et al., 2022), which are crowd-sourced via Amazon Mechanical Turk. The SMOS scores are documented with a 95% confidence interval. For the precision of subjective evaluation, each model generates 150 audio samples per dataset for testing and each sample is evaluated by two distinct workers. In addition to this, we employ supplementary objective automatic evaluation metrics including mean L1 mel-spectrogram error, Perceptual Evaluation of Speech Quality (PESQ) (Rix et al., 2001), Periodicity error and F1 score of voiced/unvoiced classification (V/UV F1)(Morrison et al., 2021) to assess sample equality. The real-time factor (RTF) assessment is also calculated, utilizing a the same RTX4090 GPU.

4.3 Comparsion With Other Models

We conduct a series of experiments on speech synthesis tasks to evaluate our model. Models we have compared with are listed below.

WaveGlow (Prenger et al., 2019), an parallel discrete flow based model;

WaveNODE (Kim et al., 2020), a continuous normalizing flow model using adjoint sensitive method during training.

DiffWave (Kong et al., 2020b), **PriorGrad** (Lee et al., 2021) **BDDM** (Lam et al., 2022), **FastDiff** (Huang et al., 2022a), four diffusion probabilistic models, all been proved to be high-fidelity. We use 6 denoising-steps for DiffWave,PriorGrad, BDDM,and FastDiff;

We train these models following the setups as in the original papers,and the results in Table 2 show that our models is comparable with different kinds of previous models. And with only one inference step, WaveODE is already able to generate acceptable audios.

4.4 Out of Distribution Situation

The generalizability of our proposed model is assessed utilizing two kinds of datasets: the musical dataset MUSDB18-HQ and several speaking datasets with unseen language. The SMOS results in Table 3 indicate that our model exhibits commendable performance in unseen scenarios, exceeding the performance of the baseline models.

Model	SMOS (↑)
Ground Truth	4.52 ± 0.08
WaveODE-1	4.08 ± 0.11
with Snake Activation	4.10 ± 0.08
with Weight Normalization	4.00 ± 0.07
w/o Conditioned Prior	3.73 ± 0.10
w/o Mixer	4.03 ± 0.07

Table 1: Ablation study results.

4.5 Ablation Study

In order to demonstrate our structural designs are effective, we have conducted several ablation studies , and the results are presented in Table 1.

Our observations are concluded as follow:

1. The snake activation function in BigVGAN cannot improve the sample quality of WaveODE, we attribute this to training objective: GANs generate waveforms directly, but we estimating the expectation of random derivatives, the periodical bias could be unhelpful.

2. Weight Normalization cannot improve the sample quality of WaveODE as well, this could be explained as the velocity function has a highly demand of precision, which could be harmed by weight normalization.

3.The conditioned prior significantly improved the one step samples' quality, and the mixer slightly helps the process too. Additionally, for full Runge-Kutta estimation (Dopri5 method) with atol=1e-3 and rtol=1e-3, the conditioned prior could reduce the number of function evaluation from 70+ to around 30.

5 Conclusion

In conclusion, this study has provided a comprehensive review of probability flow equation based models, analysing them under a unified perspective of BVP and IVP. We propose the WaveODE model, a efficient approach to mel-spectrogram conditioned speech synthesis. It could leverage the energy information in mel-spetrograms to generate a small variance prior disitribution, and then use the distilled velocity function $\boldsymbol{v}(\boldsymbol{x}, t)$ to produce accecptable waveforms in one inference step. The SMOS test and auto evaluators have demonstrated that the WaveODE model is competitive with previous diffusion models and flow based models in terms of sample quality and efficiency, and could generalize better to out of distribution data.

Model	SMOS (↑)	MelError(↓)	RTF (↓)
Ground Truth	4.52 ± 0.08	0	-
WaveGlow	3.91 ± 0.10	0.347	57.5x
WaveNODE	4.00 ± 0.09	0.336	5.03x
Diffwave-6	4.15 ± 0.07	0.160	16.1x
PriorGrad-6	4.19 ± 0.09	0.186	16.1x
BDDM-6	4.14 ± 0.10	0.194	16.1x
FastDiff-6	4.13 ± 0.08	0.245	59.6x
WaveODE-1	4.08 ± 0.11	0.165	**76.3x**
WaveODE-6	$\mathbf{4.21 \pm 0.07}$	**0.157**	28.5x

Model	PESQ (↑)	Periodicity(↓)	V/UV F1 (↑)
Ground Truth	4.644	0	1
WaveGlow	**3.133**	0.215	0.861
WaveNODE	2.582	0.213	0.850
Diffwave-6	2.899	0.176	0.890
PriorGrad-6	3.028	0.199	0.866
BDDM-6	2.973	0.148	0.930
FastDiff-6	2.902	0.195	0.876
WaveODE-1	3.02	0.125	0.932
WaveODE-6	3.10	**0.105**	**0.963**

Table 2: The subjective and objective evaluation results on LibriTTS Test dataset.

Model	Mean SMOS on MUSDB18-HQ
Ground Truth	4.43 ± 0.09
WaveGlow	3.51 ± 0.10
WaveNODE	3.45 ± 0.11
Diffwave-6	3.73 ± 0.09
PriorGrad-6	3.85 ± 0.09
BDDM-6	3.82 ± 0.10
FastDiff-6	3.69 ± 0.12
WaveODE-1	3.81 ± 0.10
WaveODE-6	$\mathbf{3.98 \pm 0.10}$

Model	Mean SMOS on Languages
Ground Truth	4.46 ± 0.09
WaveGlow	3.75 ± 0.11
WaveNODE	3.78 ± 0.12
Diffwave-6	4.10 ± 0.07
PriorGrad-6	4.15 ± 0.10
BDDM-6	4.05 ± 0.10
FastDiff-6	4.02 ± 0.08
WaveODE-1	4.05 ± 0.09
WaveODE-6	$\mathbf{4.20 \pm 0.11}$

Table 3: The SMOS results on MUSDB18-HQ data and multiple non-English datasets

6 Limitations and Potential Risks

Our main limitation is that the training objectives only improves the sample quality slightly, which is not desired since it could make the training process become slower. There still need furthur explorations to take full advantages of the probability flow equation.

As for the risks, our proposed model lowers the requirements for high-fidelity speech synthesis, which is related to the potential risks concerning media or telephone fraud.

References

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Deeply. 2021. Deeply korean read speech corpus,https://github.com/deeplyinc/Korean-Read-Speech-Corpus.

Lawrence C Evans. 2022. *Partial differential equations*, volume 19. American Mathematical Society.

Yue Fan, JW Kang, LT Li, KC Li, HL Chen, ST Cheng, PY Zhang, ZY Zhou, YQ Cai, and Dong Wang. 2020. Cn-celeb: a challenging chinese speaker recognition dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7604–7608. IEEE.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2018. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

Wenhao Guan, Qi Su, Haodong Zhou, Shiyu Miao, Xingjia Xie, Lin Li, and Qingyang Hong. 2023. Reflow-tts: A rectified flow model for high-fidelity text-to-speech. *arXiv preprint arXiv:2309.17056*.

Yiwei Guo, Chenpeng Du, Ziyang Ma, Xie Chen, and Kai Yu. 2023. Voiceflow: Efficient text-to-speech with rectified flow matching. *arXiv preprint arXiv:2309.05027*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Rongjie Huang, Max WY Lam, Jun Wang, Dan Su, Dong Yu, Yi Ren, and Zhou Zhao. 2022a. Fastdiff: A fast conditional diffusion model for high-quality speech synthesis. *arXiv preprint arXiv:2204.09934*.

Rongjie Huang, Yi Ren, Ziyue Jiang, Chenye Cui, Jinglin Liu, and Zhou Zhao. 2023. Fastdiff 2: Revisiting and incorporating gans and diffusion models in high-fidelity speech synthesis. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6994–7009.

Rongjie Huang, Zhou Zhao, Huadai Liu, Jinglin Liu, Chenye Cui, and Yi Ren. 2022b. Prodiff: Progressive fast diffusion model for high-quality text-to-speech. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2595–2605.

Won Jang, Dan Lim, Jaesam Yoon, Bongwan Kim, and Juntae Kim. 2021. Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation. *arXiv preprint arXiv:2106.07889*.

Fritz John. 1991. *Partial differential equations*, volume 1. Springer Science & Business Media.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR.

Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. 2020. Wavenode: A continuous normalizing flow for speech synthesis. *arXiv preprint arXiv:2006.04598*.

Ji-Hoon Kim, Sang-Hoon Lee, Ji-Hyun Lee, and Seong-Whan Lee. 2021. Fre-gan: Adversarial frequency-consistent audio synthesis. *arXiv preprint arXiv:2106.02297*.

Yuma Koizumi, Heiga Zen, Kohei Yatabe, Nanxin Chen, and Michiel Bacchiani. 2022. Specgrad: Diffusion probabilistic model based neural vocoder with adaptive noise spectral shaping. *arXiv preprint arXiv:2203.16749*.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020a. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020b. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. 2019. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32.

Max WY Lam, Jun Wang, Dan Su, and Dong Yu. 2022. Bddm: Bilateral denoising diffusion models for fast and high-quality speech synthesis. *arXiv preprint arXiv:2203.13508*.

Sang-gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. 2021. Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior. *arXiv preprint arXiv:2106.06406*.

Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. 2022. Bigvgan: A universal neural vocoder with large-scale training. *arXiv preprint arXiv:2206.04658*.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Ziming Liu, Di Luo, Yilun Xu, Tommi Jaakkola, and Max Tegmark. 2023. Genphys: From physical processes to generative models. *arXiv preprint arXiv:2304.02637*.

Yulong Lu, Jianfeng Lu, and James Nolen. 2019. Accelerating langevin sampling with birth-death. *arXiv preprint arXiv:1905.09863*.

Max Morrison, Rithesh Kumar, Kundan Kumar, Prem Seetharaman, Aaron Courville, and Yoshua Bengio. 2021. Chunked autoregressive gan for conditional waveform synthesis. *arXiv preprint arXiv:2110.10139*.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Wei Ping, Kainan Peng, Kexin Zhao, and Zhao Song. 2020. Waveflow: A compact flow-based model for raw audio. In *International Conference on Machine Learning*, pages 7706–7716. PMLR.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. 2019. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE.

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. 2017. Musdb18-a corpus for music separation.

Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. 2001. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE.

Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W Oard, and Matt Post. 2021. The multilingual tedx corpus for speech recognition and translation. *arXiv preprint arXiv:2102.01757*.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ning-Qian Wu and Zhen-Hua Ling. 2020. Waveffjord: Ffjord-based vocoder for statistical parametric speech synthesis. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7214–7218. IEEE.

Yilun Xu, Ziming Liu, Max Tegmark, and Tommi Jaakkola. 2022. Poisson flow generative models. *Advances in Neural Information Processing Systems*, 35:16782–16795.

Yilun Xu, Shangyuan Tong, and Tommi Jaakkola. 2023. Stable target field for reduced variance score estimation in diffusion models. *arXiv preprint arXiv:2302.00670*.

Zhenhui Ye, Rongjie Huang, Yi Ren, Ziyue Jiang, Jinglin Liu, Jinzheng He, Xiang Yin, and Zhou Zhao. 2023. Clapspeech: Learning prosody from text context with contrastive language-audio pre-training. *arXiv preprint arXiv:2305.10763*.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. 2019. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*.

Bernt Øksendal. 2003. *Stochastic Differential Equations*. Springer Berlin, Heidelberg.

Bernt Øksendal. 2013. *Stochastic Differential Equations: An Introduction with Applications*. Springer Berlin.

A Proof of Expectation Property

Let $\boldsymbol{x}(t)$ be a continuously differentiable random process on $t \in [0,1]$, if the conditional expectation $\mathrm{E}[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}|\boldsymbol{x}(t)]$ is locally Lipschitz and $p(\boldsymbol{x},t), \boldsymbol{v}(\boldsymbol{x},t)$ solve the probability flow equation, then

$$\boldsymbol{v}(\boldsymbol{x},t) = \mathrm{E}\left[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}\middle|\boldsymbol{x}(t)\right] \text{ almost surely,} \tag{28}$$

Proof: Since $p(\boldsymbol{x},t), \boldsymbol{v}(\boldsymbol{x},t)$ solve the probability flow equation,

$$\frac{\partial p}{\partial t}(\boldsymbol{x},t) + \nabla_{\boldsymbol{x}}(p(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t)) = 0 \tag{29}$$

Then for any finite supported continuously differentiable function $h(\boldsymbol{x})$, we have:

$$\int h(\boldsymbol{x})(\frac{\partial p}{\partial t}(\boldsymbol{x},t) + \nabla_{\boldsymbol{x}}(p(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t)))d\boldsymbol{x} = 0 \tag{30}$$

$$\frac{\partial}{\partial t}\int h(\boldsymbol{x})p(\boldsymbol{x},t)d\boldsymbol{x} = -\int h(\boldsymbol{x})\nabla_{\boldsymbol{x}}(p(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t)))d\boldsymbol{x} \tag{31}$$

Integrating by parts to the right hand side, since h is finite supported, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\int h(\boldsymbol{x})p(\boldsymbol{x},t)d\boldsymbol{x} = \int (p(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t))^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{x})d\boldsymbol{x} \tag{32}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}E[h(\boldsymbol{x}(t))] = E[v(\boldsymbol{x},t)^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{x}(t))] \tag{33}$$

$$E[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{x}(t))] = E[v(\boldsymbol{x},t)^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{x}(t))] \tag{34}$$

By the tower property of expectation we have:

$$E[E[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}|\boldsymbol{x}(t)]^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{x}(t))] = E[v(\boldsymbol{x},t)^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{x}(t))] \tag{35}$$

Since we could arbitrarily choose finite supported continuous differentiable function h,

$$v(\boldsymbol{x},t) = E[\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}|\boldsymbol{x}(t)] \text{ almost surely} \tag{36}$$

B Interpreting score based models into linear PDEs

We consider the general Kolmogorov forward equation:

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{\mu}(\boldsymbol{x},t)\mathrm{d}t + \boldsymbol{\sigma}(\boldsymbol{x},t)\mathrm{d}\boldsymbol{w}, \tag{37}$$

where $\boldsymbol{\mu}(\boldsymbol{x},t)$ is a vector function from $\mathbb{R}^n \times \mathbb{R}$ to $\mathbb{R}$, $\boldsymbol{\sigma}(\boldsymbol{x},t)$ is a matrix function from $\mathbb{R}^n \times \mathbb{R}$ to $\mathbb{R}^{n\times n}$, and $\mathrm{d}\boldsymbol{w}$ is the infinitesimal of n-dimentional standard Wiener process (also called Brown Motion)(Øksendal, 2003, 2013).

Now $\boldsymbol{x}(t)$ becomes a random variable, we denote its probability density function as $p(\boldsymbol{x},t)$.Assume f is an arbitrary function $\in \mathcal{C}^{(2)}$, and T is a arbitrary fixed positive time, using the tower property of conditioned expectation, we have:

$$\mathbb{E}[f(\boldsymbol{x}(T))] = \mathbb{E}[\mathbb{E}[f(\boldsymbol{x}(T))|\boldsymbol{x}(t) = \boldsymbol{x}]], \forall t \in [0,T], \tag{38}$$

we denote $\mathbb{E}[f(\boldsymbol{x}(T))|\boldsymbol{x}(t) = x]$ as $\boldsymbol{u}(x,t)$,then we have:

$$\mathbb{E}[f(\boldsymbol{x}(T))] = \int p(\boldsymbol{x},t)\mathbb{E}[f(\boldsymbol{x}(T))|\boldsymbol{x}(t) = x]\mathrm{d}x = \int p(\boldsymbol{x},t)u(\boldsymbol{x},t)\mathrm{d}x, \forall t \in [0,T]\ , \tag{39}$$

then we denote the integration as a inner product between $p(\boldsymbol{x},t)$ and $u(\boldsymbol{x},t)$, and noticing that the left hand side has nothing to do with variable t, taking derivative at $t=T$ we have:

$$\begin{aligned} 0 &= \left.\frac{\mathrm{d}\mathbb{E}[f(\boldsymbol{x}(T))]}{\mathrm{d}t}\right|_{t=T} = \left.\frac{\mathrm{d}\left\langle p(\boldsymbol{x},t), u(\boldsymbol{x},t)\right\rangle}{\mathrm{d}t}\right|_{t=T} \\ &= \left\langle \left.\frac{\partial p(\boldsymbol{x},t)}{\partial t}\right|_{t=T}, u(\boldsymbol{x},T)\right\rangle + \left\langle p(\boldsymbol{x},T), \left.\frac{\partial u(\boldsymbol{x},t)}{\partial t}\right|_{t=T}\right\rangle, \end{aligned} \tag{40}$$

now we obtain an equation with $\left.\frac{\partial p(\boldsymbol{x},t)}{\partial t}\right|_{t=T}$, where $\left.\frac{\partial u(\boldsymbol{x},t)}{\partial t}\right|_{t=T}$ could be further computed:

$$\begin{aligned} \left.\frac{\partial u(\boldsymbol{x},t)}{\partial t}\right|_{t=T} &= \lim_{t\to 0^-}\frac{u(\boldsymbol{x},t+T)-u(\boldsymbol{x},T)}{t} \\ &= \lim_{t\to 0^-}\frac{\mathbb{E}[f(\boldsymbol{x}(T))|\boldsymbol{x}(t+T)=\boldsymbol{x}]-f(\boldsymbol{x})}{t}, \end{aligned} \tag{41}$$

then according to Itô [4] lemma we do Taylor expansion at $t=T$ for $f(x(t))$ and gain:

$$= -\left(\sum_{i=1}^{n}\mu_i(\boldsymbol{x},t)\left.\frac{\partial f(\boldsymbol{x})}{\partial x_i}\right|_{t=T} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sigma_{ik}(\boldsymbol{x},t)\sigma_{jk}(\boldsymbol{x},t)\left.\frac{\partial^2 f(\boldsymbol{x})}{\partial x_i\partial x_j}\right|_{t=T}\right), \tag{42}$$

we regard this formula as a linear operator L act on function $f(\boldsymbol{x})$, where

$$\mathrm{L}(f)(\boldsymbol{x}) = \sum_{i=1}^{n}\mu_i(\boldsymbol{x},t)\left.\frac{\partial f(\boldsymbol{x})}{\partial x_i}\right|_{t=T} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sigma_{ik}(\boldsymbol{x},t)\sigma_{jk}(\boldsymbol{x},t)\left.\frac{\partial^2 f(\boldsymbol{x})}{\partial x_i\partial x_j}\right|_{t=T} \tag{43}$$

$$\left.\frac{\partial u(\boldsymbol{x},t)}{\partial t}\right|_{t=T} = -\mathrm{L}(f)(\boldsymbol{x}) \tag{44}$$

Now we have transformed the SDE into equation:

$$\left\langle \left.\frac{\partial p(\boldsymbol{x},T)}{\partial t}\right|_{t=T}, f(\boldsymbol{x})\right\rangle + \langle p(\boldsymbol{x},T), -\mathrm{L}(f)(\boldsymbol{x})\rangle = 0. \tag{45}$$

Since L is a linear operator, we could find its dual operator L^* with the integration inner product between functions using the formula of integration by parts: ($\langle \mathrm{L}(f), g\rangle = \langle f, \mathrm{L}^* g\rangle$ is the definition to dual operator L^*)

$$\mathrm{L}^*(f)(x) = -\sum_{i=1}^{n}\left.\frac{\partial[\mu_i(\boldsymbol{x},t)f(\boldsymbol{x})]}{\partial x_i}\right|_{t=T} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left.\frac{\partial^2}{\partial x_i\partial x_j}\right|_{t=T}\left(\sum_{k=1}^{n}\sigma_{ik}(\boldsymbol{x},t)\sigma_{jk}(\boldsymbol{x},t)f(\boldsymbol{x})\right), \tag{46}$$

Now the SDE can be further transformed into:

$$\left\langle \left.\frac{\partial p(\boldsymbol{x},t)}{\partial t}\right|_{t=T}, f(\boldsymbol{x})\right\rangle - \langle \mathrm{L}^*(p)(\boldsymbol{x},T), f(\boldsymbol{x})\rangle = 0. \tag{47}$$

$$\left\langle \left.\frac{\partial p(\boldsymbol{x},t)}{\partial t}\right|_{t=T} - \mathrm{L}^*(p)(\boldsymbol{x},T), f(\boldsymbol{x})\right\rangle = 0. \tag{48}$$

Since $f(\boldsymbol{x})$ is an arbitrary function $\in \mathcal{C}^{(2)}$,we have:

$$\left.\frac{\partial p(\boldsymbol{x},t)}{\partial t}\right|_{t=T} - \mathrm{L}^*(p)(\boldsymbol{x},T) = 0, \quad \forall\, T\in[0,+\infty), \tag{49}$$

[4] Itô lemma: https://en.wikipedia.org/wiki/It%C3%B4%27s_lemma

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t}+\sum_{i=1}^{n}\frac{\partial[\mu_i(\boldsymbol{x},t)p(\boldsymbol{x},t)]}{\partial x_i}-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2}{\partial x_i\partial x_j}\left(\sum_{k=1}^{n}\sigma_{ik}(\boldsymbol{x},t)\sigma_{jk}(\boldsymbol{x},t)p(\boldsymbol{x},t)\right)=0, \quad (50)$$

and this is the partial equation that the probability density function should obey. Now we can review the simple situation:

$$\mathrm{d}\boldsymbol{x}=\boldsymbol{f}(\boldsymbol{x},t)\mathrm{d}t+g(t)\mathrm{d}\boldsymbol{w} \quad (51)$$

$$\boldsymbol{\mu}(\boldsymbol{x},t)=\boldsymbol{f}(\boldsymbol{x},t),\boldsymbol{\sigma}(\boldsymbol{x},t)=g(t)\boldsymbol{I}, \quad (52)$$

the equation can be simplified into:

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t}+\nabla_{\boldsymbol{x}}[\boldsymbol{f}(\boldsymbol{x},t)p(\boldsymbol{x},t)]-\frac{1}{2}g^2(t)\nabla^2_{\boldsymbol{x}}p(\boldsymbol{x},t)=0, \quad (53)$$

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t}+p(\boldsymbol{x},t)\nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x},t)+\boldsymbol{f}(\boldsymbol{x},t)\cdot\nabla_{\boldsymbol{x}}p(\boldsymbol{x},t)-\frac{1}{2}g^2(t)\nabla^2_{\boldsymbol{x}}p(\boldsymbol{x},t)=0, \quad (54)$$

which is a linear non-compressible passive fluid partial differential equation.

C Solving standard diffusion equation

Diffusion equation, which is also known as heat equation, is a parabolic partial differential equation that could be found in many PDE textbooks(Evans, 2022; John, 1991).

To make the derivation easier, we could do change variable for t, we project the ground truth distribution at $t=1$ to $t=0$, and project the prior distribution at $t=0$ to large enough $t=T$.

Then we derive the Green's function solution to the standard diffusion equation and we assume the source point $\boldsymbol{x}'=0$ for simplicity:

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t}-\nabla^2_{\boldsymbol{x}}p(\boldsymbol{x},t)=\delta(\boldsymbol{x})\delta(t)\ , \quad (55)$$

the Fourier transformation of $p(\boldsymbol{x},t)$ is denoted as:

$$\tilde{p}(\boldsymbol{k},t)=\mathcal{F}[p]\equiv\int p(\boldsymbol{x},t)e^{-i\boldsymbol{k}\cdot\boldsymbol{x}}\mathrm{d}^N\boldsymbol{x}, \quad (56)$$

the corresponding reverse Fourier transformation of $\tilde{p}(\boldsymbol{k},t)$ is denoted as:

$$p(\boldsymbol{x},t)=\mathcal{F}^{-1}[\tilde{p}]=\frac{1}{(2\pi)^N}\int\tilde{p}(\boldsymbol{k},t)e^{i\boldsymbol{k}\cdot\boldsymbol{x}}d^N\boldsymbol{k}. \quad (57)$$

Fourier transformation's nice properties could remove the $\nabla_{\boldsymbol{x}}$ operator in some PDEs:

$$\mathcal{F}[\nabla_{\boldsymbol{x}}p]=i\boldsymbol{k}\tilde{p},\mathcal{F}[\nabla^2_{\boldsymbol{x}}p]=-|\boldsymbol{k}|^2\tilde{p}. \quad (58)$$

Apply Fourier transformation to the standard diffusion equation,we have:

$$\frac{\partial\tilde{p}}{\partial t}+|\boldsymbol{k}|^2\tilde{p}=\delta(t)\ ,$$

$$\iff \quad \frac{\partial\tilde{p}}{\partial t}+|\boldsymbol{k}|^2\tilde{p}=0\ (t>0),\quad \tilde{p}(\boldsymbol{k},0)=1\ .$$

$$\iff \quad \tilde{p}(\boldsymbol{k},t)=\exp(-|\boldsymbol{k}|^2t), \quad (59)$$

which is a Gaussian distribution in $\boldsymbol{k}$ domain. Now we transform it back to $\boldsymbol{x}$ domain:

$$\begin{aligned}
p(\boldsymbol{x},t) &= \mathcal{F}^{-1}[\tilde{p}] = \frac{1}{(2\pi)^N}\int \exp(-|\boldsymbol{k}|^2 t)\exp(i\boldsymbol{k}\cdot\boldsymbol{x})\mathrm{d}^N\boldsymbol{k} \\
&= \prod_{j=1}^{N}\left[\frac{1}{2\pi}\int_{-\infty}^{+\infty}\exp(ik_jx_j)\exp(-k_j^2t)\mathrm{d}k_j\right] \\
&= \prod_{j=1}^{N}\left[\frac{\exp(-\frac{x_j^2}{4t})}{2\pi}\int_{-\infty}^{+\infty}\exp\left[-t\left(k_j-\frac{ix_j}{2t}\right)^2\right]\mathrm{d}k_j\right] \\
&= \prod_{j=1}^{N}\left[\sqrt{\frac{\pi}{t}}\cdot\frac{\exp\left(-\frac{x_j^2}{4t}\right)}{2\pi}\right] \\
&= \frac{1}{(4\pi t)^{\frac{N}{2}}}\exp\left(-\frac{|\boldsymbol{x}|^2}{4t}\right)
\end{aligned} \tag{60}$$

which is the Green's function solution whose source is at $\boldsymbol{x}' = \boldsymbol{0}$, thus for arbitrary source position:

$$p(\boldsymbol{x},t;\boldsymbol{x}') = \frac{1}{(4\pi t)^{\frac{N}{2}}}\exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{x}'|^2}{4t}\right) \tag{61}$$

Now the diffusion equation could be solved by superposition method since the boundary condition at $t=0$ could be regarded as $p_{\text{data}}(\boldsymbol{x})\delta(t)$:

$$\begin{aligned}
p(\boldsymbol{x},t) &= \int p(\boldsymbol{x},t;\boldsymbol{x}')p_{\text{data}}(\boldsymbol{x}')\mathrm{d}^N\boldsymbol{x}' \\
\boldsymbol{v}(\boldsymbol{x},t) &= -\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x},t) = -\frac{1}{p(\boldsymbol{x},t)}\int \nabla_{\boldsymbol{x}}p(\boldsymbol{x},t;\boldsymbol{x}')p_{\text{data}}(\boldsymbol{x}')\mathrm{d}^N\boldsymbol{x}' \\
&= \frac{1}{p(\boldsymbol{x},t)}\int p(\boldsymbol{x},t;\boldsymbol{x}')\frac{\boldsymbol{x}-\boldsymbol{x}'}{2t}p_{\text{data}}(\boldsymbol{x}')\mathrm{d}^N\boldsymbol{x}' \\
&= \int p(\boldsymbol{x}'|\boldsymbol{x},t)\frac{\boldsymbol{x}-\boldsymbol{x}'}{2t}\mathrm{d}^N\boldsymbol{x}' \\
&= \mathbb{E}_{x\sim p(\boldsymbol{x}'|\boldsymbol{x},t)}\left[\frac{\boldsymbol{x}-\boldsymbol{x}'}{2t}\right]
\end{aligned} \tag{62}$$

where

$$\begin{aligned}
p(\boldsymbol{x}'|\boldsymbol{x},t) &\propto p_{\text{data}}(\boldsymbol{x}')p(\boldsymbol{x},t;\boldsymbol{x}') \\
&\propto p_{\text{data}}(\boldsymbol{x}')\exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{x}'|^2}{4t}\right)
\end{aligned} \tag{63}$$

when t is large enough, $p(\boldsymbol{x},t)$ is approximately proportional to $\exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{x}'|^2}{4t}\right)$, which could serve as a prior distribution. Now the inference process has a proper beginning and the velocity field could be trained efficiently through adding Gaussian noises to the origin clear data like diffusion probabilistic models. It's worth mentioning that, this conditioned expectation is also similar to another efficient training objective for diffusion models called stable target field objective (Xu et al., 2023), which means that the original data could be regarded as point charges, Green's function determine the analytical form of the electricity field, and the velocity field could be viewed as the join electricity field of the point charges.

D Solving time-dependent Poisson equation

Again, to make the derivation easier, we could do change variable for t, we project the ground truth distribution at $t=1$ to $t=0$, and project the prior distribution at $t=0$ to large enough $t=T$.

Firstly,we also needs to fine the Green's function solution:

$$\begin{aligned} & \frac{\partial^2 \varphi}{\partial t^2}(\boldsymbol{x}, t) + \nabla_{\boldsymbol{x}}^2 \varphi(\boldsymbol{x}, t) = \delta(\boldsymbol{x})\delta(t) \\ \Longleftrightarrow \quad & \frac{\partial^2 \varphi}{\partial t^2}(\boldsymbol{x}, t) + \nabla_{\boldsymbol{x}}^2 \varphi(\boldsymbol{x}, t) = 0 (t > 0), \quad \frac{\partial \varphi}{\partial t}(\boldsymbol{x}, 0) = \delta(\boldsymbol{x}) \end{aligned} \tag{64}$$

Similar to Appendix B, we apply Fourier transformation to the equation:

$$\begin{aligned} & \frac{\partial^2 \tilde{\varphi}}{\partial t^2}(\boldsymbol{k}, t) - |\boldsymbol{k}|^2 \tilde{\varphi}(\boldsymbol{k}, t) = 0 (t > 0), \quad \frac{\partial \tilde{\varphi}}{\partial t}(\boldsymbol{x}, 0) = \delta(x) \\ \Longleftrightarrow \quad & \tilde{\varphi}(\boldsymbol{k}, t) = \frac{u \exp(-|\boldsymbol{k}|t) + v \exp(|\boldsymbol{k}|t)}{|\boldsymbol{k}|}, \quad -u + v = 1 \end{aligned} \tag{65}$$

Since $t \to \infty, \varphi(\tilde{\boldsymbol{k}}, t) \to 0$,we have $u = -1, b = 0, \tilde{\varphi}(\boldsymbol{k}, t) = \frac{1}{|\boldsymbol{k}|} \exp(-|\boldsymbol{k}|t)$,then apply reverse Fourier transformation with some properties of hypergeometric[5]function and n-dimensional spherical coordinates mentioned in (Liu et al., 2023):

$$\varphi(\boldsymbol{x}, t) = \frac{\Gamma\left(\frac{N-1}{2}\right)}{2\pi^{\frac{N+1}{2}}} \frac{1}{(t^2 + |\boldsymbol{x}|^2)^{\frac{N-1}{2}}}, \tag{66}$$

which is the n-dimensional electricity potential function of a unit point charge at $\boldsymbol{x}' = 0$, and for arbitraty source position $\boldsymbol{x}$,we have:

$$\varphi(\boldsymbol{x}, t; \boldsymbol{x}') = \frac{\Gamma\left(\frac{N-1}{2}\right)}{2\pi^{\frac{N+1}{2}}} \frac{1}{(t^2 + |\boldsymbol{x} - \boldsymbol{x}'|^2)^{\frac{N-1}{2}}}, \tag{67}$$

Actually since Poisson equation is very special, a more simpler method to solve it could be found in PFGMs' original paper(Xu et al., 2022). Now we have:

$$p(\boldsymbol{x}, t) = \frac{\partial \varphi}{\partial t}(\boldsymbol{x}, t) = \int \frac{\partial \varphi(\boldsymbol{x}, t; \boldsymbol{x}')}{\partial t} p_{\text{data}}(\boldsymbol{x}') \mathrm{d}\boldsymbol{x}' \tag{68}$$

$$\begin{aligned} \boldsymbol{v}(\boldsymbol{x}, t) &= \frac{\nabla_{\boldsymbol{x}} \varphi(\boldsymbol{x}, t)}{\frac{\partial \varphi}{\partial t}(\boldsymbol{x}, t)} = \frac{1}{p(\boldsymbol{x}, t)} \int \nabla_{\boldsymbol{x}} \varphi(\boldsymbol{x}, t; \boldsymbol{x}') p_{\text{data}}(\boldsymbol{x}') \mathrm{d}^N \boldsymbol{x}', \\ &= \frac{1}{p(\boldsymbol{x}, t)} \int \frac{\partial \varphi(\boldsymbol{x}, t; \boldsymbol{x}')}{\partial t} \frac{\boldsymbol{x} - \boldsymbol{x}'}{t} p_{\text{data}}(\boldsymbol{x}') \mathrm{d}\boldsymbol{x}' \\ &= \mathbb{E}_{x \sim p(\boldsymbol{x}'|\boldsymbol{x}, t)} \left[\frac{\boldsymbol{x} - \boldsymbol{x}'}{t} \right] \end{aligned} \tag{69}$$

where

$$\begin{aligned} p(\boldsymbol{x}'|\boldsymbol{x}, t) &\propto p_{\text{data}}(\boldsymbol{x}') \frac{\partial \varphi(\boldsymbol{x}, t; \boldsymbol{x}')}{\partial t} \\ &\propto \frac{p_{\text{data}}(\boldsymbol{x}')}{(t^2 + |\boldsymbol{x} - \boldsymbol{x}'|^2)^{\frac{N+1}{2}}} \end{aligned} \tag{70}$$

Then we could use a training process that is very similar to diffusion models in Appendix B to train this velocity field by changing Gaussian perturbation kernel according to corresponding Green's function.

[5]hypergeometric function:https://en.wikipedia.org/wiki/Hypergeometric_function

Hyperparameters	Values
velocityChannels	[512,256,128,64,32],
velocityUpSampleRates	[8,8,2,2],
velocityKernelSizes	[[3,7,11],[3,7,11],[3,7,11],[3,7,11]],
velocityDilations	[[1,3,5],[1,3,5],[1,3,5],[1,3,5]],
mixerChannels	[128,64,32,16],
mixerUpSampleRates	[8,8,4],
mixerKernelSizes	[[3,5],[3,5],[3,5]],
mixerDilations	[[1,3],[1,3],[1,3]],
parameters	28.7M

E Hyperparameters of The Networks

The hyperparameters in our model are as follow: