Speeding Up Speech Synthesis in Diffusion Models by Reducing Data Distribution Recovery Steps via Content Transfer.

Anonymous authors
Paper under double-blind review

Abstract

Diffusion based vocoders have been criticised for being slow due to the many steps required during sampling. Moreover, the model's loss function that is popularly implemented is designed such that the target is the original input x_0 or error ϵ_0. For early time steps of the reverse process, this results in large prediction errors, which can lead to speech distortions and increase the learning time. We propose a setup where the targets are the different outputs of forward process time steps with a goal to reduce the magnitude of prediction errors and reduce the training time. We use the different layers of a neural network (NN) to perform denoising by training them to learn to generate representations similar to the noised outputs in the forward process of the diffusion. The NN layers learn to progressively denoise the input in the reverse process until finally the final layer estimates the clean speech. To avoid 1:1 mapping between layers of the neural network and the forward process steps, we define a skip parameter $\tau > 1$ such that an NN layer is trained to cumulatively remove the noise injected in the τ steps in the forward process. This significantly reduces the number of data distribution recovery steps and, consequently, the time to generate speech. We show through extensive evaluation that the proposed technique generates high-fidelity speech in competitive time that outperforms current state-of-the-art tools. The proposed technique is also able to generalize well to unseen speech.

1 Introduction

The use of deep generative models is prevalent in speech synthesis Lam et al. (2022) Chen et al. (2020) Kong et al. (2020b) Prenger et al. (2018) Kumar et al. (2019) Kong et al. (2020a). These models use generative adversarial network (GAN) Goodfellow (2016) or likelihood-based techniques. GAN-based models such as Kong et al. (2020a) and Kumar et al. (2019) exploit the training objective to make the model generate data that are indistinguishable from the training data. While GAN based models can generate high quality speech, they are difficult to train due to instability during the training process Mescheder et al. (2018). Likelihood speech synthesis-based techniques are composed of autoregressive models such as Oord et al. (2016) Kalchbrenner et al. (2018) Mehri et al. (2016) Valin & Skoglund (2019), flow-based models Prenger et al. (2019) Kim et al. (2020) Hsu & Lee (2020) and variational auto-encoders (VAE) based models Liu et al. (2022). Autoregressive speech synthesis models generate speech in a sequential nature, where the current sample to be generated is conditioned on the previously generated samples. Due to the sequential nature of speech generation, these models require many computations to generate a sample. This limits their ability to be deployed in application where faster real time generation is required. The flow-based model utilises specialised architectures to model a normalised probability model. These architectures require optimisation of many parameters during training, and hence can be computationally expensive. VAE based models on the other hand do not work well with high dimensional data Bond-Taylor et al. (2021). Another type of likelihood-based generative model that is becoming popular for speech synthesis is the diffusion probability model (DPM) Sohl-Dickstein et al. (2015). It has been explored in speech synthesis in Lam et al. (2022) Chen et al. (2020) Kong et al. (2020b). DPMs are composed of two main processes i.e., the forward and

reverse process. The forward process involves sequentially adding Gaussian noise to a given distribution until eventually it becomes identical to white noise, i.e., pure Gaussian noise. The reverse process starts with white noise and recovers the data distribution by sampling. To learn a given target distribution, DPMs require a significant number of diffusion steps during training, resulting in many reverse steps to recover the data distribution during sampling time. Due to this, speech synthesis tools using diffusion generative model are slow, a property that prohibits their real-world deployment. Recognizing this limitation, speech synthesis tools that employ diffusion generative models employ a number of techniques to reduce sampling steps. WaveGrad Chen et al. (2020) uses a grid search algorithm (GS) to reduce the sampling noise schedule. The use of grid search to shorten the noise schedule has been criticised for being computationally prohibitive when many noising steps N are used Lam et al. (2022). BDDM Lam et al. (2022) reduces the sampling schedule by first training a generative model for speech synthesis using T steps, then uses the optimised score network to train a scheduling network to learn a shorter noise schedule $N << T$ to be used during sampling. In this work, we explore the idea of using content transfer to speed up speech synthesis in DPM. Content transfer which was first used in Gatys et al. (2016) as part of style transfer, involves training layers of a neural network to minimize the distance between representations of a desired style (or content) and a white noise and iteratively transform white noise to the desired style or content. Motivated by this, we also use neural network layers to learn to generate representations of a given audio generated by a given time-step t of the forward process. Since the forward process can have many steps T, we restrict the layers of the neural network used in the reverse process to $N = \frac{T}{\tau}$ where $\tau > 1$. Intuitively, we use the layers of the neural network to reduce the noise schedule of the forward process by training a neural network such that its single layer can remove cumulative noise injected in τ steps during the forward process. Unlike Lam et al. (2022) which optimises two sets of parameters, we train the model to optimise only a single parameter set θ therefore hypothesise that the proposed method will significantly reduce sampling time and, consequently, audio generation time.

2 Denoising diffusion probabilistic model

Given an observed sample x of unknown distribution, the diffusion probabilistic model (DPM) aims to model the true distribution of the data $p(x)$. The modelled distribution $p(x)$ can then be used to generate new samples at will. DPM defines a forward process as

$$q(x_{1:T} \mid x_0) = \prod_{i=1}^{T} q(x_t \mid x_{t-1}) \tag{1}$$

Here, latent variables and true data are represented as x_t with $t = 0$ being the true data. The encoder $q(x_t \mid x_{t-1})$ seeks to convert the data distribution into a simple tractable distribution after the T diffusion steps. $q(x_t \mid x_{t-1})$ models the hidden variables as linear Gaussian models with mean and standard deviation modelled as hyperparameters Ho et al. (2020) or as learnt variables Nichol & Dhariwal (2021) Kingma et al. (2021). The Gaussian encoder is parameterized with mean $u_t(x_t) = \sqrt{\alpha_t} x_{t-1}$ and variance $\Sigma_t(x_t) = (1-\alpha_t)I$ hence

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)I) \tag{2}$$

α_t evolves with time t based on a fixed or learnable schedule such that the final distribution $p(x_T)$ is a standard Gaussian. The reverse process which seeks to recover the data distribution from the white noise is modelled as

$$p_\theta(x_{0:T}) = p(x_T) \prod_{i=1}^{T} p_\theta(x_{t-1} \mid x_t) \tag{3}$$

where

$$p(x_T) = \mathcal{N}(x_T; 0, I) \tag{4}$$

The encoder essentially describes a steady noisification of an input over time by adding Gaussian noise until eventually it becomes identical to pure noise. It is completely modelled as a Gaussian with a defined mean and variance parameters at each timestep hence it is not learned. The goal of DPM is therefore to model the reverse process $p_\theta(x_{t-1} \mid x_t)$ so that it can be exploited to generate new data samples. After the DPM has

been optimized, a sampling procedure entails sampling Gaussian noise from $p(x_T)$ and iteratively running the denoising transitions $p_\theta(x_{t-1} \mid x_t)$ for T steps to generate x_0. The DPM can be optimised through the evidence lower bound (ELBO).

$$\begin{aligned} &\log p(x) = E_q(x_1 \mid x_0)[\log p_\theta(x_0 \mid x_1)]- \\ &D_{KL}(q(x_T \mid x_0)||p(x_T))- \\ &\sum_{t=2}^{T} E_q(x_t \mid x_0)[D_{KL}(q(x_{t-1} \mid x_t, x_0)||p_\theta(x_{t-1} \mid x_t))] \end{aligned} \tag{5}$$

Using the property of isotropic Gaussians, Ho et al. (2020) shows that x_t can be conditioned directly on x_0 as:

$$x_t = \sqrt{\bar{\alpha}} x_0 + \sqrt{(1-\bar{\alpha_t})}\epsilon_0 \tag{6}$$

hence

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}} x_0, (1-\bar{\alpha_t})I) \tag{7}$$

In equation 5, the third term on the right is the denoising term that seeks to model $p_\theta(x_{t-1} \mid x_t)$ to match the ground truth $q(x_{t-1} \mid x_t, x_0)$. In Ho et al. (2020), $q(x_{t-1} \mid x_t, x_0)$ is derived as:

$$\begin{aligned} &q(x_{t-1} \mid x_t, x_0) \\ &\propto \mathcal{N}(\frac{\sqrt{\alpha}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}}_{t-1}(1-\alpha_t)x_0}{(1-\bar{\alpha})}, \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha})}I) \end{aligned} \tag{8}$$

In order to match $p_\theta(x_{t-1} \mid x_t)$ to $q(x_{t-1} \mid x_t, x_0)$ in the reverse process, the mean of $p_\theta(x_{t-1} \mid x_t)$ should be made to match that of $q(x_{t-1} \mid x_t, x_0)$ hence the mean of $p_\theta(x_{t-1} \mid x_t)$ is parameterized as

$$u_\theta(x_t, t) := \frac{\sqrt{\alpha}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}}_{t-1}(1-\alpha_t)\hat{x}_\theta(x_t, t)}{(1-\bar{\alpha})} \tag{9}$$

Here, the score network $\hat{x}_\theta(x_t, t)$ is parameterized by a neural network and it seeks to predict x_0 from a noisy input x_t and time index t. The optimization problem can therefore be simplified as:

$$L = \mathbb{E}_{t,\epsilon}[||\hat{x}_\theta(\sqrt{\bar{\alpha}} x_0 + \sqrt{(1-\bar{\alpha_t}}\epsilon, t) - x_0||_2^2] \tag{10}$$

The loss function is composed of the neural network $\hat{x}_\theta(x_t, t)$ that is conditioned on the discrete time t and noisy input x_t to predict the original ground truth input x_0. By expressing

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}}} \tag{11}$$

an equivalent optimization of modelling a neural network $\hat{\epsilon}_\theta(x_t, t)$ to predict the source noise can be derived Ho et al. (2020).

$$L = \mathbb{E}_{t,\epsilon}[||\hat{\epsilon}_\theta(\sqrt{\bar{\alpha}} x_0 + \sqrt{(1-\bar{\alpha_t}}\epsilon, t) - \epsilon_0||_2^2] \tag{12}$$

3 Related work

Deep neural network generative techniques for speech synthesis (vocoders) are either implemented using likelihood technique or generative adversarial network Goodfellow (2016). Likelihood methods are composed of autoregressive, VAE, flow, and diffusion-based vocoders. Autoregressive models such as Oord et al. (2016) Kalchbrenner et al. (2018) Mehri et al. (2016) and Valin & Skoglund (2019) are models that generate speech sequentially. The models learn the joint probability over speech data by factorizing the distribution into a product of conditional probabilities over each sample. Due to their sequential nature of speech generation, autoregressive models require a large number of computations to generate a sample. This limits their ability to be deployed in application where faster real time generation is required. However, there are models such as Paine et al. (2016), Hsu & Lee (2020) and Mehri et al. (2016) which propose techniques to speed up speech

generation in autoregressive models. Another likelihood-based speech synthesis technique is the flow-based models Rezende & Mohamed (2015) used in Prenger et al. (2019) Kim et al. (2020) Hsu & Lee (2020). These models use a sequence of invertible mappings to transform a given probability density. During sampling, flow-based models generate data from a probability distribution through the inverse of these transforms. Flow based models implement specialized models that are is complicated to train Tan et al. (2021). Denoising diffusion probabilistic models (DDPM) have recently been exploited in speech synthesis using tools such as PriorGrad Lee et al. (2021), WaveGrad Chen et al. (2020), BDDM Lam et al. (2022) and DiffWave Kong et al. (2020b). These models exploit a neural network that learns to predict the source noise that was used in the noisification process during the forward process. Diffusion-based vocoders can generate speech with very high voice quality but are slow due to the high number of sampling steps. Tools such as BDDM Lam et al. (2022) propose techniques to speed up speech generation while using diffusion models. Our proposed work also looks at how to speed up speech synthesis in diffusion models. Finally, GAN based models such as Kong et al. (2020a) and Kumar et al. (2019) exploit the training objective to make the model generate data that is indistinguishable from the training data. While GAN based models can generate high quality speech, they are difficult to train due to instability during the training process Mescheder et al. (2018). A complete review of the vocoders can be found in Tan et al. (2021).

4 Problem definition

To enable faster sampling and potentially avoid sampling with hundreds and thousands of steps, we propose a reduced noise scheduling network which sequentially learns to recover data distribution in much more fewer steps that the steps used to inject the noise during the forward process. Given the forward process of diffusion parameterized by noise schedule $\alpha \in R^T$ where $0 < \alpha_t < 1$ with $1 \leq t \leq T$, we seek to establish a shortened noise scheduling $\hat{\alpha} \in R^N$ where $0 < \hat{\alpha}_t < 1$ with $1 \leq t \leq N$. The reduced noise schedule $\hat{\alpha}$ is to be established when only the forward diffusion noise schedule α given. Equation 10 and 12 are designed such that the the targets are the original input x_0 and error ϵ_0 respectively. We hypothesise that for early time steps of the reverse process, this will result in large prediction errors which can lead to speech distortions Zhou et al. (2022) and increase the learning time. We propose a setup where the targets are the different outputs of the forward process time steps with the goal of reducing the magnitude of prediction errors. Through this, we hope to generate high fidelity speech and achieve faster convergence during training.

5 Proposed Method

Our proposed method is based on content transfer between audio in the forward process and audio generated by the layers of neural network in the reverse process. The number of steps N used to recover the data distribution is fixed by selecting a skip parameter $1 \leq \tau < T$ such that $N = \frac{T}{\tau}$. If $\tau = 1$, then each time step t in the forward process is mapped to a neural network layer in the reverse process. However, the goal is to fast track the reverse of the diffusion process, and hence we aim to select $\tau > 1$ that significantly reduces the sampling time. By doing this, a layer n that is mapped to a time step t of the forward process can eliminate noise injected from $t = t - \tau$ to $t = t$ in the diffusion process.

5.1 Representation generation

The representations discussed in this paper are generated by Wav2Vec 2.0 Baevski et al. (2020). Wav2Vec 2.0 is a speech pre-trained model that is composed of two main blocks. The first block, feature extractor is made of seven 1D convolution layers and a normalization layer. It accepts raw audio waveform and generates a representation $Z = \{z_1, \cdots, z_n\}$ with 20 ms stride between samples where each sample has a receptive field of 25ms. The second block is composed of a transformer with 24 layers that establish a contextual representation $C = \{c_1, \cdots c_n\}$ of a given audio. We do not use the quantisation module part of Wav2Vec 2.0 that is employed during self-supervised pre-training. For the forward process of the diffusion process, we use Wav2Vec to generate representations of an audio resulting from time step t (see figure 1). For the reverse process, the Wav2Vec layer-wise representations are used. We describe the details in the next section.

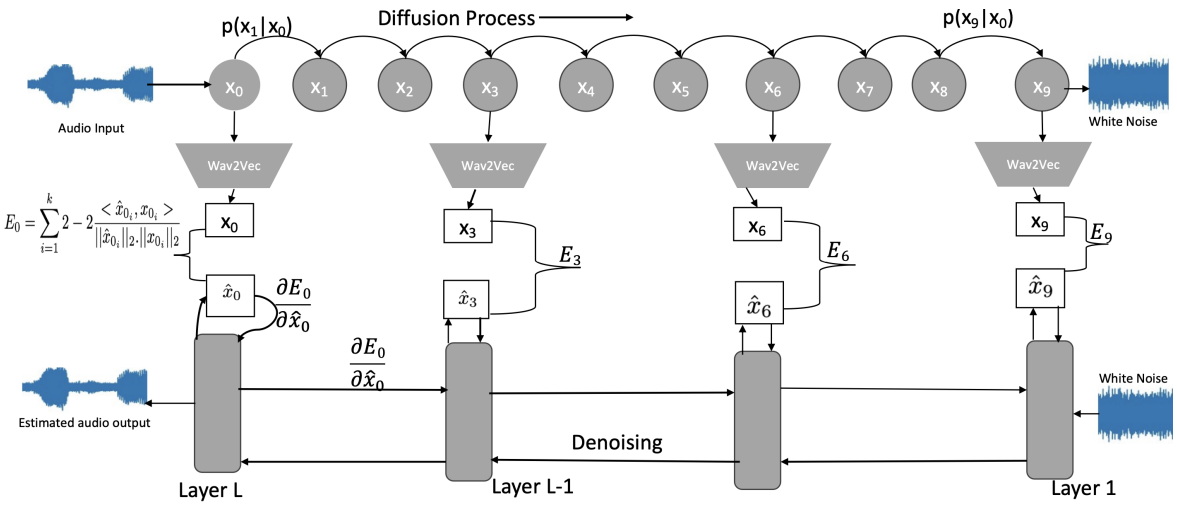

Figure 1: An overview of the unconditioned audio generation. An audio file is first processed by the forward process (10 steps in this case including input). An audio generated at a selected time step t is processed by a pre-trained model and its representation stored. In the reverse process, white noise X_N is accepted by the first layer of the neural network and processed through the layers. For each layer we store its generated representation. A layer is mapped to a given time step t. If a layer l is mapped to time step t, we minimize the mean squared error between their respective embeddings.

5.2 Unconditional speech generation

The forward process proceeds similarly to Equation 1. Recall that in equation 5, the third term on the right is the denoising term with the goal to learn a transition step $p_\theta(x_{t-1} \mid x_t)$ that estimates the tractable ground truth $q(x_{t-1} \mid x_t, x_0)$. When the KL divergence of the two denoising steps is minimised, denoising is achieved. In our case, we reparametize the third term of equation 5 as:

$$\begin{aligned}&\sum_{t=2}^{T} E_q(x_n \mid x_{n-1})[D_{KL}(q(x_{n-1} \mid x_n)||p_\theta(\hat{x}_{n-1} \mid \hat{x}_n))] = \\ &\sum_{t=2}^{T} E_q(x_{t+\tau} \mid x_t)[D_{KL}(q(x_t \mid x_{t+\tau})||p_\theta(\hat{x}_t \mid \hat{x}_{t+\tau}))]\end{aligned} \tag{13}$$

The fundamental change being that we condition x_n on x_{n-1} rather than x_0 and the model $p_\theta(\hat{x}_{n-1} \mid \hat{x}_n)$ is parametized by estimates $\hat{x}_{n-1}$ and $\hat{x}_n$ rather than actual x_{n-1} and x_n. Using Bayes theorem

$$q(x_{n-1} \mid x_n) = \frac{q(x_n \mid x_{n-1})q(x_{n-1})}{q(x_n)} \tag{14}$$

Using Markov property equation 14 becomes:

$$q(x_{n-1} \mid x_n) = \frac{q(x_n \mid x_{n-1})q(x_{n-1} \mid x_{n-2})}{q(x_n \mid x_{n-1})} = q(x_{n-1} \mid x_{n-2}) \tag{15}$$

Where

$$q(x_{n-1} \mid x_{n-2}) \propto \mathcal{N}(x_{n-1}; \sqrt{\alpha_{n-1}}x_{n-2}, (1-\alpha_{n-1})I) \tag{16}$$

Equation 13 can now be rewritten as:

$$\sum_{t=2}^{T} E_q(x_n \mid x_{n-1})[D_{KL}(q(x_{n-1} \mid x_{n-2})||p_\theta(\hat{x}_{n-1} \mid \hat{x}_n))] \tag{17}$$

The goal is to model $p_\theta(\hat{x}_{n-1} \mid \hat{x}_n)$ to estimate $q(x_{n-1} \mid x_{n-2})$ established during the forward process. Due to this, we formulate the loss function as follows:

$$L = \arg\min_\theta D_{KL}(q(x_{n-1} \mid x_{n-2})||p_\theta(\hat{x}_{n-1}|\hat{x}_n)) \tag{18}$$

$$L = \arg\min_\theta D_{KL}(\mathcal{N}(x_{n-1}; \mu_q, \Sigma_q(n))||\mathcal{N}(\hat{x}_{n-1}; \hat{\mu}_\theta, \Sigma_q(n))) \tag{19}$$

where $\mu_q = \sqrt{\alpha_{n-1}}x_{n-2}$, and $\Sigma_q(n) = 1 - \alpha_{n-1}$. $p_\theta(\hat{x}_{n-1}|\hat{x}_n)$ is supposed to be modelled to have a similar distribution to that of $q(x_{n-1} \mid x_{n-2})$ as much as possible Ho et al. (2020). Hence, the distribution of $p_\theta(\hat{x}_{n-1}|\hat{x}_n)$ is modelled as a Gaussian with mean $\hat{\mu}_\theta = \sqrt{\alpha_{n-1}}\hat{x}_\theta(\hat{x}_{n-1}, n-2)$ and variance $\Sigma_q(n)$. With $\hat{x}_\theta(\hat{x}_{n-1}, n-2)$ being parameterized by a neural network that seeks to predict x_{n-2} from the estimate $\hat{x}_{n-1}$ and the time step $n-2$. Based on this, equation 19 can be expressed as:

$$L = \arg\min_\theta \frac{1}{2\sqrt{\alpha_{n-1}}}[||\sqrt{\alpha_{n-1}}\hat{x}_\theta(\hat{x}_{n-1}, n-2) - \sqrt{\alpha_{n-1}}x_{n-2}||_2^2] \tag{20}$$

$$L = \arg\min_\theta \frac{1}{2}\sqrt{\alpha_{n-1}}[||\hat{x}_\theta(\hat{x}_{n-1}, n-2) - x_{n-2}||_2^2] \tag{21}$$

Equation 21 can be generalized as

$$L = \arg\min_\theta \frac{1}{2}\sqrt{\alpha_{n+1}}[||\hat{x}_\theta(\hat{x}_{n+1}, n) - x_n||_2^2] \tag{22}$$

Therefore, optimizing the loss function boils down to learning a neural network $\hat{x}_\theta$ to predict x_n established during the forward process. The neural network should be conditioned on an estimate $\hat{x}_{n+1}$ and time step n to predict x_n. Unlike the loss in equation 10 where the neural network $\hat{x}_\theta(x_t, t)$ conditioned on a random noisy input x_t predicts the original noiseless input x_0, the loss in equation 22, conditions the neural network on an estimate $\hat{x}_{n+1}$ of the previous step $n+1$ of the reverse process to predict the noisy output generated at a time step n in the forward process. For instance to recover the original input x_0 using the loss function in equation 22, we will need to design a neural network conditioned on estimate $\hat{x}_1$ and then minimise the loss:

$$L = \arg\min_\theta \frac{1}{2}\sqrt{\alpha_{n+1}}[||\hat{x}_\theta(\hat{x}_1, 0) - x_0||_2^2] \tag{23}$$

Ideally, to recover the original input x_0, we will need to design N neural networks each for each time step. To avoid this, we exploit N layers $\hat{x}_\theta^l(.)$ of NN to perform the time step predictions of the reverse process. Hence, equation 22 is implemented as:

$$L = \arg\min_\theta \frac{1}{2}\sqrt{\alpha_{n+1}}[||\hat{x}_\theta^l(\hat{x}_{n+1}) - x_n||_2^2] \tag{24}$$

where $\hat{x}_{n+1}$ is the prediction of the previous neural network layer, i.e., $\hat{x}_\theta^{l-1}(\hat{x}_{n+2})$. We no longer condition the neural network layer on the discrete time n since the layers encode time (i.e., the time steps are encoded by the neural network layers which are implemented sequentially). Intuitively, since $n = t + \tau$, the network layer $p_\theta(\hat{x}_{n-1} \mid \hat{x}_n)$ is supposed to remove the cumulative noise injected in multiple steps τ during the forward process. Based on this, in the unconditioned implementation, the reverse process starts with white noise $x_N \sim N(0, I)$ and takes $N = \frac{T}{\tau}$ steps (layers) to recover the data distribution. During implementation, the white noise x_N is passed through the first layer of the Wav2Vec and each of the representation $\hat{x}_n \in R^{k \times h}$ encoded by the 24 layers of Wav2Vec is stored. Here, k represents the number of samples generated by the feature extractor of the Wav2Vec. We then minimize the loss between the $L2$ normalized embeddings $\hat{x}_n \in R^{k \times h}$ and the embeddings $x_n \in R^{k \times h}$ (established by fine-tuned Wav2Vec) of the audio generated by time step n during the forward process using mean squared error.

$$L_n = ||\hat{x}_\theta^l(\hat{x}_{n+1}) - x_n||_2^2 = \sum_{i=1}^{k} 2 - 2\frac{< \hat{x}_{n_i}, x_{n_i} >}{||\hat{x}_{n_i}||_2.||x_{n_i}||_2} \tag{25}$$

The gradients of loss with respect to the layer's activations are then computed using standard error back-propagation.

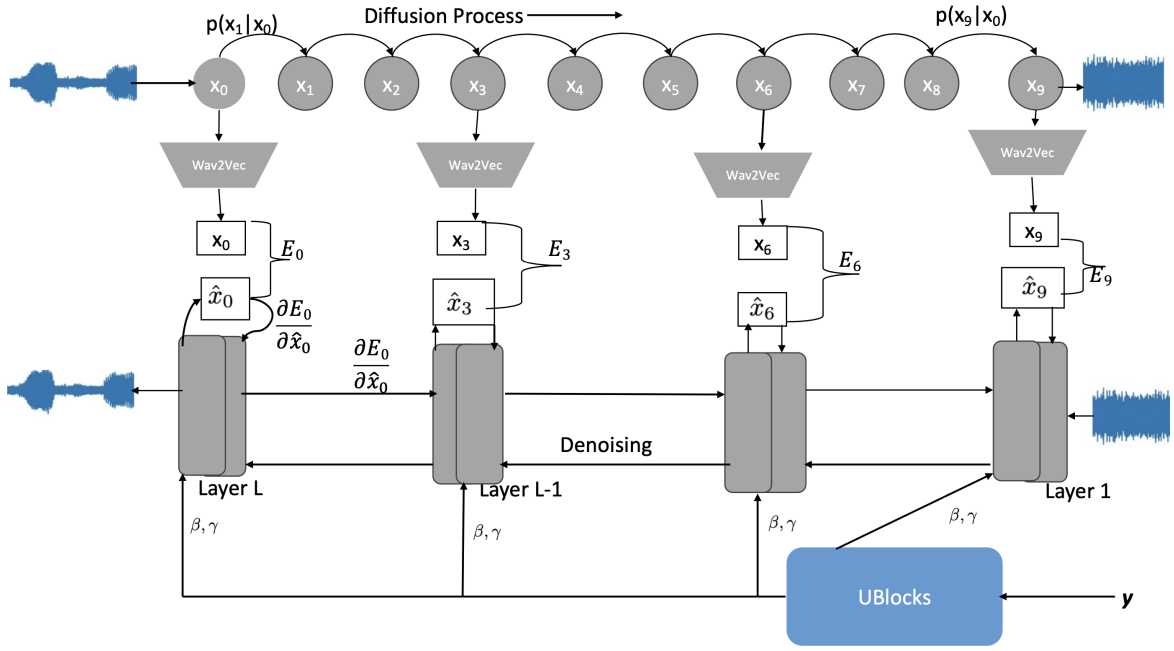

Figure 2: An overview of the conditioned audio generation. Compared to unconditioned audio generation in figure 1, the conditioned audio generation includes upsampling blocks that accepts Mel-spectrogram y that contains acoustic features of the audio to be generated. The upsampling blocks process the Mel-spectrogram to generate $\beta = f(y)$ and $\gamma = f(y)$. Both β and γ are used to modulate the activation of the neural network layer according to equation 27.

5.3 Conditional speech generation

To enable the model to generate speech that follow a given acoustic features, we condition the loss in equation 24 on acoustic features y as:

$$L = ||\hat{x}_\theta^l(\hat{x}_{n+1}, y) - x_n||_2^2 \tag{26}$$

Therefore, we design the score network $\hat{x}_\theta^l(.,.)$ such that it can process both noisy estimate $\hat{x}_{n+1}$ and acoustic features y. To achieve this, we exploit feature-wise linear modulation (FiLM) Perez et al. (2018) which has also been used in Chen et al. (2020). Through FiLM, we adaptively influence neural network layer estimates by applying affine transformation to layer activation based on the input Mel spectrogram y (see Figure 2).

$$FiLM(\hat{x}_{n-1}) = \gamma \odot \hat{x}_{n-1} + \beta \tag{27}$$

where γ and $\beta \in R^{k \times h}$ modulates $\hat{x}_{n+1}$ based on a certain Mel-spectrogram y and $\odot$ is the hadamard product.

6 Evaluation

This section discusses how we developed and evaluated the proposed technique which we refer as **DiCon** (**Di**noising by **Con**tent transfer).

Dataset: In keeping with the trend in the speech synthesis domain and to allow for comparison with other existing tools, we used the most popular datasets i.e., LJSpeech dataset for single speaker evaluation and VCTK dataset for multi-speaker evaluation. LJSpeech dataset consists of 13,100 audio clips sampled at 22KHz. The audio clips are from a female speaker that vary in length from 1 to 10 seconds with a total of about 24hrs. For multispeaker, we used the VCTK dataset, which is sampled at 48KHz and consists of 109 English speakers with various accents. VCTK was downsampled to 22KHz. Similar to Chen et al. (2020), for LJSpeech, we used 12,764 utterance subset which is approximately 23 hours for training the model and

evaluated it on test set of 130 utterances subset. For multi-speaker evaluation, we used the data split used in Lam et al. (2022) of 100 speakers for training and 9 were used for evaluation. From each audio we extracted a 128-dimensional Mel spectrogram features. Similar to Chen et al. (2020) we used a 50-ms Hanning window, 12.5-ms frame shift, and a 2048-point FFT with upper and lower frequencies of 20 Hz and 12 kHz lower.
Training parameters: The model was trained using a single NVDIA V100 GPU. We used Adam Optimiser and the cyclical learning rate Smith (2017) with a minimum learning rate of $1e-4$ and a maximum of $1e-1$. We used a batch size of 32 and trained for 1M steps. Similar to Chen et al. (2020), for conditioned audio generation we used Mel-spectrogram extracted from ground truth audio as conditioning audio features during training while during testing we used Mel-spectrogram generated by Tacotron 2 model Shen et al. (2018). To generate the FiLM parameters β and γ, we use the upsampling blocks proposed in Chen et al. (2020) and use the parameters to modulate the activations of a given layer as described in equation 27.
Baseline models: We compared the proposed method with other state-of-the-art vocoders. We used models that have publicly available implementations where we can generate a sample for human evaluation. The baseline models used include WaveNet Oord et al. (2016)[1], WaveGlow Prenger et al. (2018) [2], MelGAN Kumar et al. (2019) [3], HiFi-GAN Kong et al. (2020a) [4], WaveGrad Chen et al. (2020) [5], DiffWave Kong et al. (2020b) [6], BDDM Lam et al. (2022) [7] and FastDiff Huang et al. (2022) [8].
Metrics: For subjective evaluation, we used the Mean Opinion Score (MOS) metric to evaluate the performance of the proposed model compared to the baseline tools. For each model we collected samples generated by the model. We also randomly selected samples from original audio samples. Each of the samples was presented to human evaluators one at time for them to rate the quality of speech on its naturalness on a 5-point Mean Opinion Score (MOS) scale. The scores used were Bad: 1, Poor: 2, Fair: 3, Good: 4, Excellent: 5 with a rating increment of 0.5. A single evaluator was required to rate 10 samples. Human evaluators were contracted via Amazon Mechanical Turk where they were required to wear headphones and be English speakers. For objective evaluation, we use Short-time objective intelligibility (STOI) Taal et al. (2011), perceptual evaluation of speech quality (PESQ) algorithm Rix et al. (2001), Deep Noise Suppression MOS (DNSMOS) which is a reference-free metric that evaluates perceptual speech quality Reddy et al. (2021). It is a DNN based model trained on human ratings obtained by using an online framework for listening experiments based on ITU-T P.808. We also use SIG, BAK, and OVRL: The non-intrusive speech quality assessment model DNSMOS P.835 Reddy et al. (2022)is based on a listening experiment according to ITU-T P.835 and evaluates speech based on three MOS scores: speech quality (SIG), background noise quality (BAK), and the overall quality (OVRL) of speech. To evaluate the speed of speech generation we used real-time factor (RTF).
Model Configurations: To train the model, we experimented with different number of steps in the forwards process while the reverse steps were kept constant at 24. We experimented with forward step (fsteps) of 1200, 960, 720 and 240 while the reverse steps (rsteps) were kept constant at 24 hence we selected a skip parameter $\tau = \{50, 40, 30, 10\}$ respectively. The model accepts a 0.3 second input of audio. For the forward process α_i increases linearly from α_1 to α_N defined as $Linear(\alpha_1, \alpha_N, N)$ such as $Linear(1 \times 10^{-4}, 0.005, 1200)$.

6.1 Results

6.1.1 Single speaker

For conditional speech generation on a single speech dataset, the subjective MOS and objective STOI, PESQ, DNSMOS, SIG, BAK and OVRL results are shown in table 1. Table 1 also shows the speech generation time RTF. For MOS, the best performing configuration of the proposed technique DiCon(1200,24) differs from the best performing tool DiffWave(200 steps) by a margin of 0.03. The MOS value of DiCon(1200,24) has 0.30 difference with that of ground truth. DiCon(1200,24) registers the best results in all the objective

[1]https://github.com/r9y9/wavenet_vocoder
[2]https:// github.com/NVIDIA/waveglow
[3]https://github.com/descriptinc/melgan-neurips
[4]https://github.com/jik876/hifi-gan
[5]https://github.com/tencent-ailab/bddm
[6]https://github.com/tencent-ailab/bddm
[7]https://github.com/tencent-ailab/bddm
[8]https://FastDiff.github.io/

metrics of STOI, PESQ, DNSMOS, SIG, BAK and OVRL. With regards to speed of speech generation, all the configuration of the proposed model are competitive with the observation that as the step size τ becomes smaller the speed of generation increases. We hypothesize that this is because a neural network layer has a reduced load of the amount of noise it is supposed to remove. We also note that the more forward steps, the better quality of audio the model can generate. However, more forward steps make the audio generation slower.

Table 1: Evaluation results of the conditioned version of the proposed method and how it compares to other state of the art tools on the evaluation metrics when single-speaker dataset is used.

LJSpeech test-dataset								
Model	**MOS(↑)**	**STOI(↑)**	**PESQ (↑)**	**RTF (↓)**	**DNSMOS (↑)**	**SIG (↑)**	**BAK (↑)**	**OVRL (↑)**
Ground truth	4.68±0.15	-	-	-	4.77	4.67	4.82	4.70
BDDM(12 steps)	4.37±0.15	0.967	3.68	0.543	4.45	4.24	4.44	4.34
DiffWave(200 steps)	**4.41**± 0.13	0.966	3.62	5.9	4.38	4.32	4.43	4.40
WaveGrad(1000 steps)	4.34± 0.15	0.909	3.41	38.2	4.41	4.29	4.34	4.32
HIFI-GAN	4.29± 0.14	0.957	3.27	0.0134	4.14	4.09	4.17	4.15
MelGAN	3.52± 0.12	0.946	2.67	0.00396	3.62	3.39	3.78	3.33
WaveGlow	3.11± 0.14	0.961	3.17	0.0198	3.67	3.34	3.01	3.06
WaveNet	3.51 ± 0.15	0.921	2.93	318.6	3.71	3.66	3.83	2.45
DiCon(fsteps:1200 rsteps 24)	4.38 ± 0.12	**0.977**	**3.74**	0.0042	**4.49**	**4.43**	**4.52**	**4.41**
DiCon(fsteps:960 rsteps 24)	4.31 ± 0.15	0.9573	3.70	0.00371	4.45	4.39	4.40	4.38
DiCon(fsteps:720 rsteps 24)	4.13 ± 0.15	0.948	3.68	0.002912	4.21	4.18	4.25	4.20
DiCon(fsteps:240 rsteps 24)	3.77± 0.13	0.8932	3.13	**0.00182**	3.87	3.78	3.86	3.83

6.1.2 Multi-speaker

The results of the performance of the proposed technique on the multi-speaker dataset are shown in table 2. For this dataset, the proposed technique can generalize to unseen speakers and DiCon(1200,24) configuration has the best MOS score of 4.39 which has a gap of 0.17 from the ground truth. It also registers the best performance on all the objective metrics.

Table 2: Evaluation results of the conditioned version of the proposed method and how it compares to other state of the art tools on the evaluation metrics when multi-speaker dataset is used.

VCTK test-dataset								
Model	**MOS(↑)**	**STOI(↑)**	**PESQ (↑)**	**RTF (↓)**	**DNSMOS (↑)**	**SIG (↑)**	**BAK (↑)**	**OVRL (↑)**
Ground Truth	4.56±0.05	-	-	-	4.78	4.73	4.80	4.77
BDDM(12 steps)	4.33±0.05	0.9610	3.61	0.438	4.47	4.38	4.43	4.32
DiffWave(200 steps)	4.37± 0.04	0.9678	3.68	5.9	4.44	4.28	4.36	4.36
WaveGrad(1000 steps)	4.31± 0.05	0.9630	3.43	38.2	4.38	4.21	4.44	4.26
HIFI-GAN	4.12± 0.05	0.943	3.51	0.0134	4.21	4.19	4.33	4.19
MelGAN	3.42± 0.05	0.8965	2.65	0.00396	3.67	3.48	3.52	3.37
WaveGlow	3.38± 0.04	0.8702	2.56	0.0198	3.79	3.61	3.88	3.48
WaveNet	3.73 ± 0.05	0.8989	2.98	318.6	3.91	3.74	3.90	3.85
DiCon(fsteps:1200 rsteps 24)	**4.39** ± 0.05	**0.981**	**3.81**	0.0042	**4.53**	**4.41**	**4.56**	**4.47**
DiCon(fsteps:960 rsteps 24)	4.26 ± 0.05	0.9500	3.67	0.00371	4.49	4.39	4.58	4.41
DiCon(fsteps:720 rsteps 24)	4.09 ± 0.03	0.9430	3.61	0.002912	4.11	4.33	4.40	4.14
DiCon(fsteps:240 rsteps 24)	3.08 ± 0.05	0.8709	2.78	0.00182	3.35	3.38	3.65	3.19

6.1.3 Unconditional speech generation

Here, the model was trained using multi-speaker dataset. To generate a speech sample, we sample white noise at random and process it through the trained model without conditioning it on any acoustic features. The results for unconditional speech generation are shown in table 3. For short clips of 0.3s DiCon(fsteps:1200 rsteps 24) attains the MOS score of 3.08. Listening to the audio clips, we noticed a phenomenon where the clips begin by generating coherent sounding sentences, but the coherence drops with time. We will investigate the reason for this phenomenon in our future work. However, the model can generate clean sounding speeches, i.e., almost free of noise or artefacts. This is captured by the high BAK score that measures the background noise.

Table 3: Results of the unconditioned proposed method on multi-speaker dataset.

VCTK test-dataset					
Model	**MOS(↑)**	**DNSMOS (↑)**	**SIG (↑)**	**BAK (↑)**	**OVRL (↑)**
DiCon(fsteps:1200 rsteps 24)	3.08 ± 0.05	3.14	3.06	4.51	3.06
DiCon(fsteps:960 rsteps 24)	3.04 ± 0.05	3.09	2.99	4.42	3.03
DiCon(fsteps:720 rsteps 24)	3.01 ± 0.03	3.06	2.97	4.25	2.99
DiCon(fsteps:240 rsteps 24)	2.93 ± 0.05	2.95	2.87	3.86	2.88

7 Conclusion

This paper presents DiCon, a technique for speeding up speech generation in diffusion models using neural network layers. We exploit the layers of the neural network to progressively recover the data distribution from white noise. Using content transfer, we demonstrate how an NN network layer can be exploited to implicitly perform denoising. We use a skip parameter τ to guide the mapping of NN layers to the forward process, and hence reduce the number of distribution recovery steps. In conditional speech generation, we use FiLM to infuse the acoustic features of a given speech into the denoising process. Based on evaluation, we demonstrate that the proposed technique generates superior quality speech samples at a competitive speed.

References

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33: 12449–12460, 2020.

Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

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

Po-chun Hsu and Hung-yi Lee. Wg-wavenet: Real-time high-fidelity speech synthesis without gpu. *arXiv preprint arXiv:2005.07412*, 2020.

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

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

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

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

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

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

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

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

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

Yanqing Liu, Ruiqing Xue, Lei He, Xu Tan, and Sheng Zhao. Delightfultts 2: End-to-end speech synthesis with adversarial vector-quantized auto-encoders. *arXiv preprint arXiv:2207.04646*, 2022.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pp. 3481–3490. PMLR, 2018.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

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

Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A Hasegawa-Johnson, and Thomas S Huang. Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

R Prenger, R Valle, and B Catanzaro. A flow-based generative network for speech synthesis, 2018.

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

Chandan KA Reddy, Vishak Gopal, and Ross Cutler. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6493–6497. IEEE, 2021.

Chandan KA Reddy, Vishak Gopal, and Ross Cutler. Dnsmos p. 835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 886–890. IEEE, 2022.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra. Perceptual evaluation of speech quality (PESQ) - A new method for speech quality assessment of telephone networks and codecs. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2:749–752, 2001. ISSN 15206149. doi: 10.1109/icassp.2001.941023.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472. IEEE, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time-frequency weighted noisy speech. *IEEE Transactions on Audio, Speech and Language Processing*, 19(7):2125–2136, 2011. ISSN 15587916. doi: 10.1109/TASL.2011.2114881.

Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. A survey on neural speech synthesis. *arXiv preprint arXiv:2106.15561*, 2021.

Jean-Marc Valin and Jan Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5891–5895. IEEE, 2019.

Rui Zhou, Wenye Zhu, and Xiaofei Li. Single-Channel Speech Dereverberation using Subband Network with A Reverberation Time Shortening Target. *arXiv preprint arXiv:2210.11089*, 2022. URL `http://arxiv.org/abs/2204.08765`.