

# Noise Estimation for Generative Diffusion Models

Robin San Roman<sup>\*1</sup>, Eliya Nachmani<sup>\* 2,3</sup>, Lior Wolf<sup>2</sup>

<sup>1</sup> École Normale Supérieure Paris-Saclay

<sup>2</sup> Tel-Aviv University

<sup>3</sup> Facebook AI Research

sanroman.robin@gmail.com, enk100@gmail.com, wolf@cs.tau.ac.il

## Abstract

Generative diffusion models have emerged as leading models in speech and image generation. However, in order to perform well with a small number of denoising steps, a costly tuning of the set of noise parameters is needed. In this work, we present a simple and versatile learning scheme that can step-by-step adjust those noise parameters, for any given number of steps, while the previous work needs to retune for each number separately. Furthermore, without modifying the weights of the diffusion model, we are able to significantly improve the synthesis results, for a small number of steps. Our approach comes at a negligible computation cost.

## Introduction

Deep generative models have seen a tremendous advancement in the past few years. The main successful architectures can be divided into two categories: (i) autoregressive models, such as VQ-VAE for images (Razavi, Oord, and Vinyals 2019) and Wavenet for speech (Oord et al. 2016). (ii) non-autoregressive models, for example, StyleGAN (Karras et al. 2020) for vision application and WaveGAN (Donahue, McAuley, and Puckette 2018) for audio synthesis.

An emerging class of non-autoregressive models is the one of Denoising Diffusion Probabilistic Models (DDPM). Such methods use diffusion models and denoising score matching in order to generate images (Ho, Jain, and Abbeel 2020) and speech (Chen et al. 2020). The DDPM model learns to perform a diffusion process on a Markov chain of latent variables. The diffusion process transforms a data sample into Gaussian noise. During inference the reverse process is used, which is called the denoising process. The inference procedure starts from Gaussian noise and iteratively refines the signal. This process is often conditioned on the class and attributes that one wishes to generate.

In order to get high quality synthesis, a large number of denoising steps are used (i.e. 1000 steps). To allow the process to converge to a high quality output with only a small number of denoising steps, a costly grid search is required in order to find a noise schedule that would produce high-fidelity results.

In this paper, we propose a novel method for obtaining the noise schedule based on the conditioning data. The noise schedule that the proposed method produces leads to a high

fidelity synthesis of samples, even for a small number of denoising steps. Moreover, for a given amount of denoising steps, the proposed method obtains better results than the previous models, when their noise schedule is determined by a costly per-sample grid search for the optimal parameters.

Our method introduces a novel neural network that is able to monitor and control the denoising process. Instead of fixing in advance the steps of the reverse process that will be skipped, this network is able, by estimating the amount of noise in the data, to schedule the subsequent steps of the denoising process.

Our results are demonstrated on two major domains: vision and audio. In the first domain, the proposed method is shown to provide a better FID score for generated images, when the number of steps is restricted. For speech data, we show that the proposed method improves various measures, such as Perceptual Evaluation of Speech Quality (PESQ) and short-time objective intelligibility (STOI).

## Related Work

Diffusion Probabilistic Models were first introduced in the seminal work of Sohl-Dickstein et al. (Sohl-Dickstein et al. 2015), who presented the idea of using an iterative neural diffusion process for destroying the structure of a given distribution, while learning the reverse neural diffusion process for restoring the structure in the data. It was shown that the proposed neural diffusion process can learn the data distribution in domains, such as images and time series. The main issue with the proposed neural diffusion process is that during training it requires up to thousands of iterative steps in order to learn the target distribution.

In (Song and Ermon 2019), a new generative model based on the score matching method (Hyvärinen and Dayan 2005) and Langevin dynamics was introduced. The proposed model estimates and samples the logarithm of the data density, which is the Stein score function (Liu, Lee, and Jordan 2016). The proposed method achieves state of the art results for modeling the CIFAR-10 dataset.

The two ideas of (i) neural Diffusion Probabilistic Models and (ii) generative models based on score matching, were combined by the DDPM method of Ho et al. (Ho, Jain, and Abbeel 2020). DDPM presents a generative model based on the neural diffusion process and applies score matching for image generation. Subsequently, in (Chen et al. 2020) a

---

<sup>\*</sup>Equal contribution

generative neural diffusion process based on score matching was applied to speech generation, obtaining state of the art results in comparison to well-established methods, such as Wavenet (Oord et al. 2016), Wavernn (Kalchbrenner et al. 2018) and GAN-TTS (Bińkowski et al. 2019). A parallel contribution presented high fidelity speech generation results using a different neural diffusion process (Kong et al. 2020).

One major limitation of the generative neural diffusion process is that in order to generate a high quality sample, one should use a large number of diffusion steps, e.g., a thousand steps are often used. Denoising Diffusion Implicit Models (DDIMs) (Song, Meng, and Ermon 2021) is an acceleration for the denoising process. It employs non-Markovian diffusion processes, which leads to a faster sample procedure than other diffusion models.

A further development in the score based generative models is to consider this process as a solution to a stochastic differential equation (Song et al. 2020). This method achieves state of the art performance for unconditional image generation on CIFAR-10. An alternative approach trains an energy-based generative model using a diffusion process that is applied to increasingly noisy versions of a dataset (Gao et al. 2020), also presenting results on CIFAR-10.

The recent TimeGrad model (Rasul et al. 2021) is a diffusion process for probabilistic time series forecasting, which was shown empirically to outperform Transformers (Vaswani et al. 2017) and LSTMs (Hochreiter and Schmidhuber 1997) on some datasets. In another concurrent work, a multinomial diffusion process is learned by adding categorical noise to the process (Hoogeboom et al. 2021). Competitive results are presented for image segmentation and language modeling.

## Background

Denoising Diffusion Probabilistic Model (DDPM) are neural network that learn the gradients of the data log density  $\nabla_y \log p(y)$ :

$$s(y) = \nabla_y \log p(y) \quad (1)$$

Given those gradients, one can then use Langevin Dynamics to sample from the probability iteratively

$$\tilde{y}_{i+1} = \tilde{y}_i + \frac{\eta}{2} s(\tilde{y}_i) + \sqrt{\eta} z_i \quad (2)$$

Where  $\eta > 0$  is the step size and  $z_i \sim \mathcal{N}(0, I)$ .

The formalization of Denoising Diffusion Probabilistic Models (DDPM) by Ho et al (Ho, Jain, and Abbeel 2020) employs a parameterized Markov chain trained using variational inference, in order to produce samples matching the data after finite time. The transitions of this chain are learned to reverse a diffusion process. This diffusion process is defined by a Markov chain that gradually adds noise in the data with a noise schedule  $\beta_1, \dots, \beta_N$  and is defined as:

$$q(y_{1:N}|y_0) = \prod_{n=1}^N q(y_n|y_{n-1}), \quad (3)$$

where  $N$  is the length of the diffusion process, and  $y_N, \dots, y_n, y_{n-1}, \dots, y_0$  is a sequence of latent variables with the same size as the clean sample  $y_0$ .

---

Algorithm 1: DDPM training procedure

---

```

1: repeat
2:    $y_0 \sim d(y_0)$ 
3:    $n \sim \mathcal{U}(\{1, \dots, N\})$ 
4:    $\sqrt{\bar{\alpha}} \sim \mathcal{U}([l_{n-1}, l_n])$ 
5:    $\varepsilon \sim \mathcal{N}(0, I)$ 
6:    $y_n = \sqrt{\bar{\alpha}} y_0 + \sqrt{1 - |\bar{\alpha}|} \varepsilon$ 
7:   Take gradient descent step on:
     $\|\varepsilon - \varepsilon_\theta(y_s, x, \sqrt{\bar{\alpha}})\|_1$ 
8: until converged

```

---

At each iteration, the diffusion process adds Gaussian noise, according to the noise schedule:

$$q(y_n|y_{n-1}) := \mathcal{N}(y_n; \sqrt{1 - \beta_n} y_{n-1}, \beta_n \mathbf{I}), \quad (4)$$

where  $\beta_n$ , is the noise schedule as defined above.

The diffusion process can be simulated for any number of steps with the closed formula:

$$y_n = \sqrt{\bar{\alpha}} y_0 + \sqrt{1 - \bar{\alpha}} \varepsilon, \quad (5)$$

where  $\alpha_i = 1 - \beta_i$ ,  $\bar{\alpha}_n = \prod_{i=1}^n \alpha_i$  and  $\varepsilon = \mathcal{N}(0, \mathbf{I})$ .

One can use this to implement the DDPM training algorithm (Alg.1) which is defined in (Chen et al. 2020). The input to the training algorithm is the dataset  $d$ . The algorithm samples  $s$ ,  $\bar{\alpha}$  and  $\varepsilon$ . The noisy latent variable  $y_s$  is calculated and fed to the DDPM neural network  $\varepsilon_\theta$ . A gradient descent step is taken in order to estimate the  $\varepsilon$  noise with the DDPM network. By the end of the algorithm, the DDPM network can estimate the noise added during the diffusion process.

When  $\sqrt{\bar{\alpha}}$  is close to 1, the diffusion process adds a small amount of noise and when  $\sqrt{\bar{\alpha}}$  is close to 0, there are large amounts of noise that are added to the generation process.

As mentioned in (Chen et al. 2020), sampling the noise level  $\sqrt{\bar{\alpha}}$  in the uniform distribution  $\mathcal{U}([0, 1])$  gives poor empirical results. This is due to the fact that the network  $\varepsilon_\theta$  would rarely be trained to fine tune good examples ( $\sqrt{\bar{\alpha}}$  close to 1).

Instead, one can sample  $\sqrt{\bar{\alpha}}$  such that the distribution of the training examples match the forward process, i.e., there are an equal amount of training samples that correspond to every step of the diffusion process. The first step is to sample a state  $n$  ( $n \sim \mathcal{U}\{1, \dots, N\}$  line 3) in the forward process and then sample the noise level using:

$$\sqrt{\bar{\alpha}} \sim \mathcal{U}[l_n, l_{n-1}] \quad (6)$$

Where:

$$l_0 = 1, \quad l_n = \sqrt{\prod_{i=0}^n 1 - \beta_i} \quad (7)$$

In Algorithm 1 (line 7) the DDPM is trained to learn the noise  $\varepsilon$  directly, instead of learning the Markov chain gradients. In (Ho, Jain, and Abbeel 2020) the authors show that the following reparametrization leads to better empirical results:

$$\varepsilon = -\sqrt{1 - \bar{\alpha}_n} \nabla_{y_n} \log q(y_n|y_0) \quad (8)$$

---

**Algorithm 2: DDPM sampling algorithm**


---

```

1:  $y_N \sim \mathcal{N}(0, I)$ 
2: for  $n = N, \dots, 1$  do
3:    $z \sim \mathcal{N}(0, I)$ 
4:    $\hat{\varepsilon} = \varepsilon_\theta(y_n, x, \sqrt{\bar{\alpha}_n})$ 
5:    $y_{n-1} = \frac{y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} \hat{\varepsilon}}{\sqrt{\alpha_n}}$ 
6:   if  $n \neq 1$  then
7:      $y_{n-1} = y_{n-1} + \sigma_n z$ 
8:   end if
9: end for
10: return  $y_0$ 

```

---



---

**Algorithm 3:  $P_\theta$  training procedure**


---

```

1: repeat
2:    $y_0 \sim q(y_0)$ 
3:    $s \sim \mathcal{U}(\{1, \dots, N\})$ 
4:    $\sqrt{\bar{\alpha}} \sim \mathcal{U}([l_{s-1}, l_s])$ 
5:    $\varepsilon \sim \mathcal{N}(0, I)$ 
6:    $y_s = \sqrt{\bar{\alpha}} y_0 + \sqrt{1 - |\bar{\alpha}|} \varepsilon$ 
7:    $\hat{\alpha} = P_\theta(y_s)$ 
8:   Take gradient descent step on:
     $\|\log(1 - \bar{\alpha}) - \log(1 - \hat{\alpha})\|_2$ 
9: until converged

```

---

The trained model  $\varepsilon_\theta$  can be used to perform inference using a variation of Langevin dynamics (Eq. 2). The following update from (Song, Meng, and Ermon 2021) is used to reverse a step of the diffusion process:

$$y_{n-1} = \frac{y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} \varepsilon_\theta(y_n, x, \sqrt{\bar{\alpha}_n})}{\sqrt{\alpha_n}} + \sigma_n \varepsilon, \quad (9)$$

where  $\varepsilon$  is white noise. Ho et al. (Ho, Jain, and Abbeel 2020) showed that adding white noise of variance  $\sigma_n^2 = \beta_t$  is optimal, when the inference procedure is initialized with gaussian noise ( $y_N \sim \mathcal{N}(0, I)$ ).

One can use this update rule in Eq. 9 to sample from the data distribution, by starting from a Gaussian noise and then step-by-step reversing the diffusion process. Algorithm 2 is used to sample with the network  $\varepsilon_\theta$ .

Since our experiments on images are unconditional, the network no longer needs the input  $x$ . The update equation that we use is defined in (Song, Meng, and Ermon 2021):

$$y_{n-1} = \sqrt{\bar{\alpha}_{n-1}} \hat{y}_{0,n} + \sqrt{1 - \bar{\alpha}_{n-1} - \tilde{\sigma}_n^2} \varepsilon_\theta(y_n, \bar{\alpha}_n) + \tilde{\sigma}_n \varepsilon, \quad (10)$$

where  $\hat{y}_{0,n} = \frac{y_n - \sqrt{1 - \bar{\alpha}_n} \varepsilon_\theta(y_n, \bar{\alpha}_n)}{\sqrt{\bar{\alpha}_n}}$  is the prediction of  $y_0$ ,  $\varepsilon \sim \mathcal{N}(0, I)$  is white noise, and  $\tilde{\sigma}$  is a new parameter of the generative process.

One can apply the rule of Eq.10 with  $\tilde{\sigma} = 0$ , in which case no random noise is added. This makes the process deterministic and leads to the best results in most of the scenarios (Song, Meng, and Ermon 2021).

---

**Algorithm 4: Model inference procedure**


---

```

1:  $N$  Number of iterations
2:  $y_N \sim \mathcal{N}(0, I)$ 
3:  $\alpha, \beta = \text{initialNoiseSchedule}()$ 
4: for  $n = N, \dots, 1$  do
5:    $z \sim \mathcal{N}(0, I)$ 
6:    $\hat{\varepsilon} = \varepsilon_\theta(y_n, \sqrt{\bar{\alpha}_n})$  or  $\varepsilon_\theta(y_n, t)$  where  $\bar{\alpha}_n \in [l_t, l_{t-1}]$ 
7:    $y_{n-1} = \frac{y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} \hat{\varepsilon}}{\sqrt{\alpha_n}}$ 
8:   if  $n \in U$  then
9:      $\hat{\alpha} = P_\theta(y_{n-1})$ 
10:     $\alpha, \beta, \tau = \text{updateNoiseSchedule}(\hat{\alpha}, n)$ 
11:   end if
12:   if  $n \neq 1$  then
13:      $y_{n-1} = y_{n-1} + \sigma_n z$ 
14:   end if
15: end for
16: return  $y_0$ 

```

---

## Method

We note that at training time the data is constructed in such a way (cf. Eq. 5) that we can feed the network  $\varepsilon_\theta$  with the noisy data  $y_n$  and with ground truth noise level  $\sqrt{\bar{\alpha}}$ . However, at a given inference step, the amount of noise in the data  $y_n$  is unknown. In most methods, the conditioning used is a predefined one. As a result, the input conditioning given to the network  $\varepsilon_\theta$  is not exploited at its full potential.

This inexact analysis of the noise is especially detrimental when the number of steps in the generation process is small. In this case, the quality of the samples at intermediate states ( $y_n$ ) varies widely.

To solve this problem, we introduce a novel neural network  $P_\theta$  that estimates the value of  $\bar{\alpha}$ , thus providing better conditioning for the diffusion network. The input to the neural network  $P_\theta$  is the data  $y_n$  generated at step  $n$  and its output is the estimated noise level  $\hat{\alpha}$ . This network provides a continuous control signal to the generation process, by providing the DDPM network  $\varepsilon_\theta$  with a conditioning signal that relates to the actual quality of the generation.

Figure 1 depicts the generation process used. Similarly to (Song, Meng, and Ermon 2021), the idea is to use a pretrained DDPM ( $\varepsilon_\theta$ ) from (Ho, Jain, and Abbeel 2020) and skip some of the states in the process to shorten the generation time. However, our model includes a Noise estimation model ( $P_\theta$ ) to calculate, between denoising steps, adjustments of the noise schedule.

## Noise Level Estimation

The network is trained with a similar procedure as Alg. 1. The sampling of the noise level is done using the distribution described in Eq. 6. Given the input  $y_n$ , the network  $P_\theta$  estimates the noise level ( $\bar{\alpha}$ ).

Empirically, we found that the performance of the network in low noise situations is critical to the performance of our method. Indeed, the last steps of the generation process are responsible for the quality of the final sample. At those stages, the amount of noise is very small, and  $\bar{\alpha}$  ap-

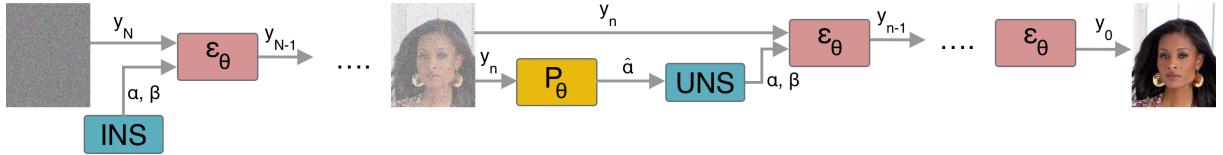


Figure 1: An overview of our generative process. INS and UNS are respectively the functions initializeNoiseSchedule() and updateNoiseSchedule( $\hat{\alpha}$ ).

proaches 1. (See experiments Fig. 9(b)) We, therefore, design our regression loss on  $\bar{\alpha}$  :

$$\mathcal{L}(\bar{\alpha}, \hat{\alpha}) = \|\log(1 - \bar{\alpha}) - \log(1 - \hat{\alpha})\|_2 \quad (11)$$

This loss penalizes with higher cost the errors when close to 1 resulting in better network performances in this region.

### Noise Schedule Adjustments

We wish to use  $\hat{\alpha}$ , the output of the  $P_\theta$ , in order to adjust the noise schedule parameters. In the following, we present how obtain those parameters, assuming they follow either a linear or a Fibonacci distribution. This allows us to define the following function:

$$\alpha, \beta = \text{updateNoiseSchedule}(\hat{\alpha}, n) \quad (12)$$

Where  $\alpha, \beta$  are the set of noise parameters and  $n$  is the number of remaining denoising steps. As we show, in order to define this function, it is sufficient to estimate the first parameters  $\beta_0$  and the type of distribution.

**Linear Schedule** In the general case,  $\bar{\alpha}$  is given by:

$$\bar{\alpha} = \prod_{i=0}^{n-1} (1 - \beta_i) \quad (13)$$

Since the values of the  $\beta_i$  are typically between  $10^{-6}$  and  $10^{-2}$  we can use the Taylor approximation  $\log(1 - \beta_i) \approx -\beta_i$ . We can derive from Eq. 13:

$$\log(\bar{\alpha}) = \sum_{i=0}^{n-1} \log(1 - \beta_i) \approx - \sum_{i=0}^{n-1} \beta_i \quad (14)$$

Assuming the linear schedule, the expression of  $\beta_i$  with respect to  $i$  in the range  $\{0, \dots, n-1\}$  is:

$$\beta_i = \beta_0 + ix, \quad (15)$$

where  $x$  is the step size. Therefore, we have:

$$\log(\bar{\alpha}) = - \left( \sum_{i=0}^{n-1} \beta_0 + ix \right) \quad (16)$$

and

$$x = -2 \frac{(\log(\bar{\alpha}) + n\beta_0)}{n(n-1)} \quad (17)$$

Once  $x$  is recovered, Eq. 15 provides us with the noise parameters required to perform the remaining denoising steps.

**Fibonacci Schedule** Chen et al. (Chen et al. 2020) employ a Fibonacci schedule for a low number of denoising steps:

$$\beta_{i+2} = \beta_i + \beta_{i+1} \quad (18)$$

We can find a closed form for this series given  $\bar{\alpha}$  and  $\beta_0$ , which will allow us to compute all the terms. The homogeneous recurrent equation is:

$$\beta_{i+2} - \beta_{i+1} - \beta_i = 0 \quad (19)$$

Thus, the series  $(\beta_i)$  is of the form:

$$\beta_i = A\varphi^i + B\varphi'^i, \quad (20)$$

where  $\varphi$  and  $\varphi'$  are the solutions of  $x^2 - x - 1 = 0$

With a straightforward induction on  $n$ , one can show that:

$$\forall n > 2, \beta_1 + \sum_{i=0}^{n-1} \beta_i = \beta_{n+1} \quad (21)$$

Combining Eq. 21 with the approximation of Eq. 14, we obtain that  $A$  is the solution of the following system of linear equations:

$$\begin{cases} A\varphi^0 + B\varphi'^0 = \beta_0 \\ A\varphi^1 + B\varphi'^1 - \log(\bar{\alpha}) = A\varphi^{n+1} + B\varphi'^{n+1} \end{cases} \quad (22)$$

The unique solution  $(A, B)$  of these equations is:

$$\begin{aligned} A &= \beta_0 - \frac{\log(\bar{\alpha}) - \beta_0(\varphi - \varphi^{n+1})}{\varphi' - \varphi - (\varphi'^{n+1} - \varphi^{n+1})} \\ B &= \frac{\log(\bar{\alpha}) - \beta_0(\varphi - \varphi^{n+1})}{\varphi' - \varphi - (\varphi'^{n+1} - \varphi^{n+1})} \end{aligned} \quad (23)$$

Thus a closed form solution is obtained that allows to compute the Fibonacci noise schedule  $(\beta, \alpha)$ , given the noise level  $\bar{\alpha}$  and  $\beta_0$ .

**Conditioning on Discrete Indexes** Most DDPMs are not conditioned on the noise level  $\bar{\alpha}$  but instead use a discrete integer index that represents steps in the reverse process. The two types of conditioning are related. Instead of giving the network with the exact value  $\sqrt{\bar{\alpha}}$ , the network is given the integer  $t$  of the interval  $[l_t, l_{t-1}]$  (defined in Eq. 7), which contains the estimated value.

To enable our method to work with this type of DDPM, we estimate the noise level and feed the network with the integer encoding corresponding interval.

	MCD ( $\downarrow$ )	PESQ ( $\uparrow$ )	STOI ( $\uparrow$ )
1000 iterations	2.65	3.29	0.959
Grid Searched	<b>2.76</b>	2.78	0.924
Our method	2.96	<b>3.14</b>	<b>0.943</b>

Table 1: Comparison between a grid searched noise schedule and an adjusting noise schedule for speech generation.

## Inference procedure

Our inference procedure is introduce in algorithm 4. The idea is to have a set of step indexes  $U$  for which, we readjust the noise schedule. For simplicity, in all of our experiments,  $U = \{1, 2, \dots, N\}$  for a given number of steps  $N$ .

The adjustment is done using the neural network  $P_\theta$ , which estimates the noise level  $\bar{\alpha}$ . Given this estimation, we deduce the sets of parameters  $\alpha, \beta$  for the remaining denoising steps, as shown in the Noise Schedule adjustments Section.

The noise schedule (vectors  $\alpha$  and  $\beta$ ) is initialised with a set of predefined values (function `initialNoiseSchedule()`). The rest of the algorithm (lines 3-15) is very similar to the algorithm 2. The only difference (lines 7-9) is that we use a set of iteration indexes  $U$  for which we will adjust the noise schedule vectors.

For the iteration  $n \in U$ , we estimate the noise level using the model  $P_\theta$ , then we follow the deterministic method described previously (function `updateNoiseSchedule( $\bar{\alpha}, n$ )`) to compute the adjustment of the noise schedule.

## Experiments

To demonstrate the wide applicability of our approach, we perform experiments in both speech synthesis and image generation. In both cases, we determine the optimal few-step scheduling for state of the art models.

### Speech Generation

For speech generation, we used a WaveGrad implementation that came with a grid searched noise schedule for 6 iterations (Vovk 2020). We trained a small model  $P_\theta$  based on a the version of ConvTASNet (Luo and Mesgarani 2019) architecture.

The model is composed of the encoding and masking module of ConvTASNet. Its decoding part is not utilized. The ConvTASNet architecture cuts, using a separator network, the signal in chunks and processes each chunk individually. Our network further applies a fully connected layer with an output dimension of 1, followed by a sigmoid activation function to each chunk. The final output  $\hat{\alpha}$  is the average of the outputs of all the chunks. The encoder has  $N = 64$  filters of size  $L = 16$  (the parameter names follow (Luo and Mesgarani 2019)). The separator uses stacks of  $X = 4$  convolutional blocks of kernel size  $P = 4$  with  $R = 4$  repeats. It has  $H = 128$  channels in the convolutional blocks and  $B = 64$  in the bottleneck and residual paths.

To train our model  $P_\theta$ , we use the same split of the LJ-Speech dataset (Ito and Johnson 2017) used by (Vovk 2020) for training the WaveGrad model. Our model was trained with

the Adam optimizer (Kingma and Ba 2014) and a learning rate of 0.001. Each batch includes 32 samples from an audio duration of 3 seconds with a sampling rate of 22050 Hz.

Next, we evaluate the synthesis quality. The DDPM employed in our experiments was trained using Mel Spectrogram as the conditioning signal  $x$ . Following (Chen et al. 2020), in our experiments, the ground truth Mel-Spectrogram is given as an input in order to allow the computation of the relative speech metrics (MCD(Kubiczek 1993), PESQ(Rix et al. 2001), STOI(Taal et al. 2011)).

In our experiments, we performed inference for six iterations with an adjustment of the noise schedule at every step i.e.  $U = \{1, \dots, 6\}$ . This noise schedule adjustment uses the Fibonacci method.

Sample results can be found under the following link. As can be heard from the samples that are provided, for few iterations, our method obtains a much better improvement than the baseline method, after applying a costly grid search for the optimal parameters of the baseline method. This is also captured by the qualitative results, presented in Tab. 1. Even though our method results in a small decrease in the MCD, we demonstrate a large improvement in both PESQ and STOI.

### Image Generation

For image generation, we use denoising diffusion models trained in (Jonathan Ho 2020), relying on the implementation available in (Jiaming Song and Ermon 2020). We trained our model  $P_\theta$  on three image datasets (i) CelebA 64x64 (Liu et al. 2015), (ii) LSUN Bedroom 256x256, and (iii) LSUN Church 256x256 (Yu et al. 2015).

The model  $P_\theta$  used for the noise estimation employed a VGG11 (Simonyan and Zisserman 2014) backbone pre-trained on ImageNet (Deng et al. 2009). We added ReLU activations, followed with a fully connected layer with an output dimension 1 and a final sigmoid activation to the backbone. An Adam optimizer(Kingma and Ba 2014) with a learning rate of  $10^{-4}$  was used, with a batch size of size 64.

The Fréchet Inception Distance (FID) (Heusel et al. 2017) is used as the benchmark metric. For all experiments, similarly to the DDIM paper, we compute the FID score with 50,000 generated images using the torch-fidelity implementation (Obukhov et al. 2020).

Figure 4 depicts the FID score of our method using the update in Eq. 9 is comparison to the baseline DDIM method. Both methods use the same DDIM diffusion model provided in (Jiaming Song and Ermon 2020) for CelebA. Our method uses the adjusted noise schedule at every step and generates a linear noise schedule.

Different plots are given for different values of  $\eta$ , which is the parameter that control  $\tilde{\sigma}$  in Eq. 10:

$$\tilde{\sigma} = \eta \sqrt{\beta_n(1 - \bar{\alpha}_{n-1})(1 - \bar{\alpha}_n)} \quad (24)$$

As can be seen, our method improves by a large gap the FID score the DDIM method. For example, for three iteration with  $\eta = 0.0$  our method improve the FID score by 163.4. The gap in performance is maintained for up to 10 iterations.

In Figure 5, we demonstrate the progression over a six iteration denoising process for both our method and the DDIM

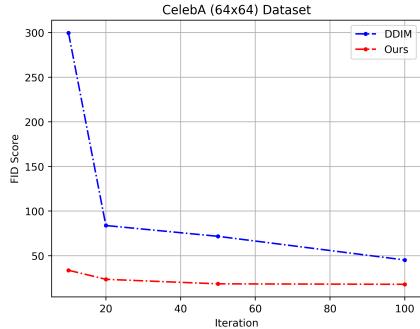


Figure 2: FID Score obtained for CelebA for our method and DDPM as a function of the number of iterations.

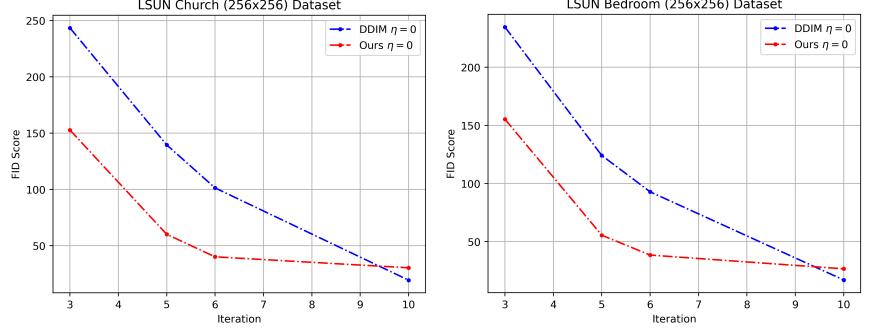


Figure 3: The FID score, over a small number of iterations, obtained for LSUN (a) Church and (b) Bedroom classes, for DDIM method and for our method.

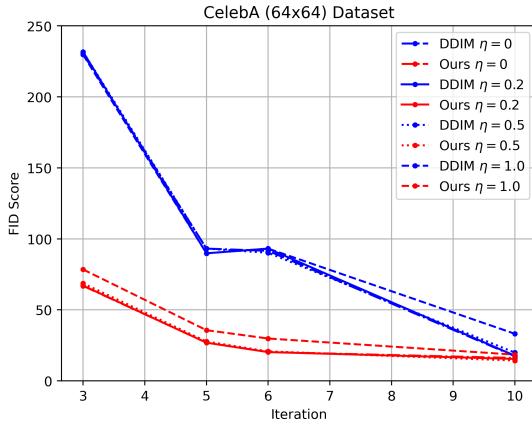


Figure 4: The FID score, over a small number of iterations, obtained for the CelebA dataset, for different  $\eta$  values, for DDIM method and for our method.

$ U $ (# adjustments)	5	3	2	1	0	Grid Search
PESQ	3.14	3.11	3.02	2.61	2.54	2.78

Table 2: PESQ score for six-iteration generations with respect to the number of noise schedule adjustments.

method that we use as a baseline. Evidently, our generated images are more coherent and sharper than the DDIM method.

Figure 6 presents samples for celebA 64x64 generation, given a different number of target steps. Our inference procedure is able to generate decent images for as little as 3 denoising steps. We also show convincing generations results for 5 and 6 steps generation processes. This results demonstrate that our method helps toward finding the best non-Markovian denoising process, especially when generating with very few iterations. Figures 7 and 8 depicts comparison of generative processes and generated samples between our method and the DDIM baseline. Our method overall clearly improves sharpness and contrast over the baseline.

In Figure 2, we compare our method with DDPM for the



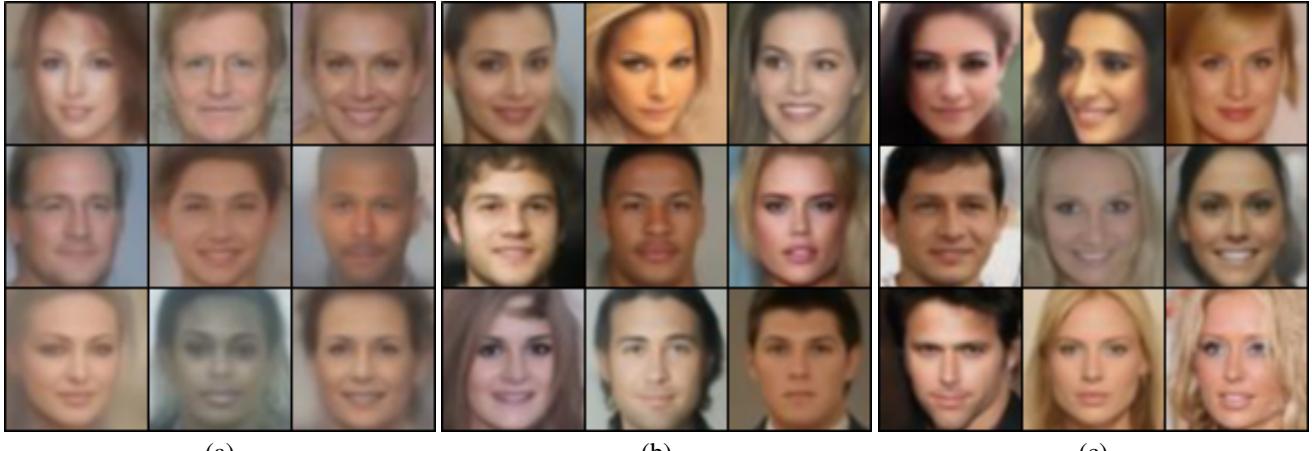
Figure 5: Typical examples of the denoising processes for 6 iterations  $\eta = 0$ . For three different noise inputs, we compare our method (top) to DDIM (bottom).

Method/#Iteration	10	20	50	100
No adjustment (DDPM/DDIM)	0.41	0.80	2.02	4.03
Adjustment every iteration	0.59	1.19	2.92	6.02

Table 3: Mean image generation time in second obtained for CelebA 64x64 for our method and DDPM/DDIM.

CelebA dataset. As can be seen, our method improves, by a large margin, the results of the DDPM method up to 100 iterations. For example, for 10 iteration, we improve the FID score by 266.03.

Similarly, in Figures 3 we provide the FID score for LSUN



(a)

(b)

(c)

Figure 6: Generated images with our method  $\eta = 0$  for (a) 3, (b) 5 and (c) 6 iterations.

(a)

(b)

Figure 7: LSUN 256x256 synthesis examples for  $N = 6$  iterations. The same input noise used for both process. (a) Church dataset, (b) Bedroom dataset. The top row in each example is our method and the bottom row is the DDIM method.

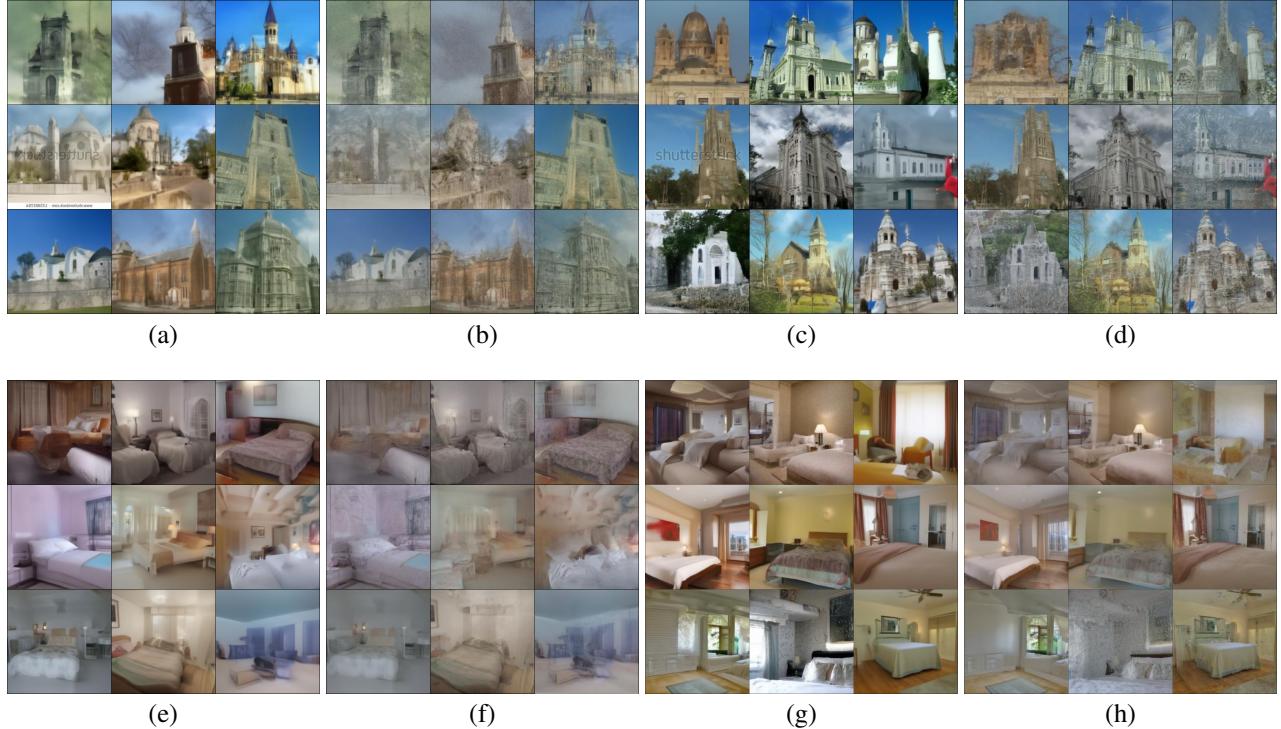


Figure 8: LSUN Church and Bedroom 256x256 datasets. Comparison of our method and the DDIM baseline for 6 and 10 iterations. First row is for LSUN Church 256x256 (a) ours with 6 iterations, (b) DDIM with 6 iteration, (c) ours with 10 iterations, (d) DDIM with 10 iterations. Second row is for LSUN Bedroom 256x256 (e) ours with 6 iterations, (f) DDIM with 6 iterations, (g) ours with 10 iterations, (h) DDIM with 10 iterations.

Church and Bedroom 256x256 datasets, using the published DDIM models. As can be seen for a small number of iteration, i.e. less then 10, our method greatly improves the results of the baseline method DDIM. In (Luhman and Luhman 2021), authors propose a distillation framework that can sample in one iteration. The method performs FID scores of 54.09 on LSUN Church and 60.97 on Bedroom. As can be seen our method gets better results for as few as 6 iterations.

**Additional Experiments** We perform ablation experiments to justify some claims and choice in our method. In Figure 9(b) we computed the optimal noise schedule for a data sample. We perturbed the values of  $\bar{\alpha}_t$  for the different  $t$  by  $10^{-4}$ . One can clearly see that perturbations on  $\bar{\alpha}_0, \bar{\alpha}_1, \bar{\alpha}_2$  results in huge drop of the performance whereas the others do not change the results much. It shows that the performance of the sampling algorithm is mostly dependent on the last steps, where the alpha values are small. This is why we use the loss 11 and want our model to perform his best when  $\bar{\alpha}$  is close to 1. In order evaluate the accuracy of  $P_\theta$  in recovering  $\bar{\alpha}$ , noisy speech signals are generated according to Eq. 5 with  $\bar{\alpha}$  values between 0 and 1. The noise level is then estimated by the trained network  $P_\theta$ . The results of this experiment are presented in Figure 9(a). The Mean Square Error (MSE) between the ground truth  $\bar{\alpha}$  (used to generate the noisy speech) and the network estimation  $\hat{\alpha}$  is depicted for  $\bar{\alpha}$  between 0 and 1. Each point is computed with 16 audio files of 3 secs from the validation set. As can be seen, our model is able

to estimate the noise level within an error of at most  $10^{-4}$  and with even better performance when alpha is close to 1, which is where the precision is critical to the performance of our method. In Table 2 we provide the PESQ score for 6 denoising steps, with various number of adjustments. As can been seen, the best results are when readjusting every steps, yet our method already outperforms the grid search for two adjustments. Table 3 depicts the mean run-time to generate an image on an Nvidia Titan X over 1000 generations. It compares our method that adjusts the noise schedule at every steps with a fully predefined schedule. This table shows that even though we use VGG11 which is not a cost efficient method to estimate the noise, the generation time increases by a moderate amount.

## Conclusions

When employing diffusion models with a limited number of steps, it is required to carefully choose the schedule of the synthesis process. Some of the previous methods perform a grid search to find the optimal global noise parameters per each number of steps. However, this fixed selection does not adjust according to the specific sample that is being generated. Our method adjusts on-the-fly the noise parameters and thus alters the subsequent states of the process. Our solution is based on estimating, during inference, the current noise level. It is, therefore, generic and independent, given the current sample, from the conditioning parameters. It remains for

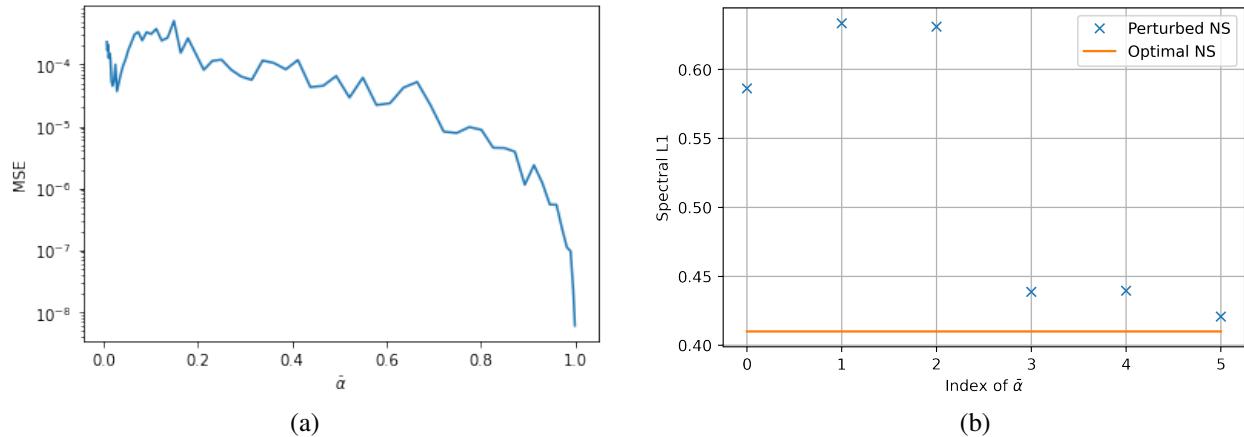


Figure 9: (a) Performance of the network  $P_\theta$  for the speech data for the noise estimation task itself, (b) L1 distance between the mel spectrogram of the groundtruth and the generated sample w.r.t the index of the perturbed

future work to check whether the same  $P_\theta$  network can be used across multiple datasets.

## References

- Bińkowski, M.; Donahue, J.; Dieleman, S.; Clark, A.; Elsen, E.; Casagrande, N.; Cobo, L. C.; and Simonyan, K. 2019. High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*.
- Chen, N.; Zhang, Y.; Zen, H.; Weiss, R. J.; Norouzi, M.; and Chan, W. 2020. WaveGrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*.
- Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.
- Donahue, C.; McAuley, J.; and Puckette, M. 2018. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.
- Gao, R.; Song, Y.; Poole, B.; Wu, Y. N.; and Kingma, D. P. 2020. Learning Energy-Based Models by Diffusion Recovery Likelihood. *arXiv preprint arXiv:2012.08125*.
- Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*.
- Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*.
- Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.
- Hoogeboom, E.; Nielsen, D.; Jaini, P.; Forré, P.; and Welling, M. 2021. Argmax Flows and Multinomial Diffusion: Towards Non-Autoregressive Language Models. *arXiv preprint arXiv:2102.05379*.
- Hyvärinen, A.; and Dayan, P. 2005. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).
- Ito, K.; and Johnson, L. 2017. The LJ Speech Dataset. <https://keithito.com/LJ-Speech-Dataset/>.
- Jiaming Song, C. M.; and Ermon, S. 2020. Denoising Diffusion Implicit Models. <https://github.com/ermongroup/ddim>.
- Jonathan Ho, P. A., Ajay Jain. 2020. Denoising Diffusion Probabilistic Models. <https://github.com/hojonathanho/diffusion>.
- Kalchbrenner, N.; Elsen, E.; Simonyan, K.; Noury, S.; Casagrande, N.; Lockhart, E.; Stimberg, F.; Oord, A.; Dieleman, S.; and Kavukcuoglu, K. 2018. Efficient neural audio synthesis. In *International Conference on Machine Learning*, 2410–2419. PMLR.
- Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119.
- Kingma, D.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.
- Kong, Z.; Ping, W.; Huang, J.; Zhao, K.; and Catanzaro, B. 2020. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.
- Kubichek, R. 1993. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, volume 1, 125–128 vol.1.
- Liu, Q.; Lee, J.; and Jordan, M. 2016. A kernelized Stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, 276–284. PMLR.
- Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
- Luhman, E.; and Luhman, T. 2021. Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed. *CoRR*, abs/2101.02388.
- Luo, Y.; and Mesgarani, N. 2019. Conv-tasnet: Surpassing ideal time-frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8): 1256–1266.
- Obukhov, A.; Seitzer, M.; Wu, P.-W.; Zhydenko, S.; Kyl, J.; and Lin, E. Y.-J. 2020. High-fidelity performance metrics for generative models in PyTorch. Version: 0.2.0, DOI: 10.5281/zenodo.3786540.
- Oord, A. v. d.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.
- Rasul, K.; Seward, C.; Schuster, I.; and Vollgraf, R. 2021. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. *arXiv preprint arXiv:2101.12072*.
- Razavi, A.; Oord, A. v. d.; and Vinyals, O. 2019. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*.
- Rix, A. W.; Beerends, J. G.; Hollier, M. P.; and Hekstra, A. P. 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, 749–752 vol.2.
- Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2256–2265. PMLR.
- Song, J.; Meng, C.; and Ermon, S. 2021. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*.
- Song, Y.; and Ermon, S. 2019. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*.
- Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2020. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv preprint arXiv:2011.13456*.

Taal, C. H.; Hendriks, R. C.; Heusdens, R.; and Jensen, J. 2011. An Algorithm for Intelligibility Prediction of Time–Frequency Weighted Noisy Speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7): 2125–2136.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vovk, I. 2020. WaveGrad. <https://github.com/ivanvovk/WaveGrad>.

Yu, F.; Zhang, Y.; Song, S.; Seff, A.; and Xiao, J. 2015. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*.