Fast Grad-TTS: Towards Efficient Diffusion-Based Speech Generation on CPU

Ivan Vovk, Tasnima Sadekova, Vladimir Gogoryan, Vadim Popov, Mikhail Kudinov, Jiansheng Wei

Huawei Noah's Ark Lab

ivan.vovk@huawei.com, sadekova.tasnima@huawei.com

Abstract

Recently, score-based diffusion probabilistic modeling has shown encouraging results in various tasks outperforming other popular generative modeling frameworks in terms of quality. However, to unlock its potential and make diffusion models feasible from the practical point of view, special efforts should be made to enable more efficient iterative sampling procedure on CPU devices. In this paper, we focus on applying the most promising techniques from recent literature on diffusion modeling to Grad-TTS, a diffusion-based text-to-speech system, in order to accelerate it. We compare various reverse diffusion sampling schemes, the technique of progressive distillation, GAN-based diffusion modeling and score-based generative modeling in latent space. Experimental results demonstrate that it is possible to speed Grad-TTS up to 4.5 times compared to vanilla Grad-TTS and achieve real time factor 0.15 on CPU while keeping synthesis quality competitive with that of conventional text-to-speech baselines.

Index Terms: speech synthesis, text-to-speech, diffusion probabilistic modeling

1. Introduction

Lately, Diffusion Probabilistic Models (DPMs, [1, 2]) have shown good performance in various speech-related tasks such as vocoding [3, 4], acoustic feature generation for text-to-speech [5, 6, 7, 8], voice conversion [9], singing voice generation [10, 11], audio upsampling [12]. Apart from high quality of the samples they produce, diffusion-based models have certain remarkable properties: e.g. it has been shown [13] that diffusion text-to-speech models allow for pitch and content control during speech generation.

The main obstacle preventing diffusion models from being extensively used in practice is their slow iterative inference. For unconditional diffusion models it usually takes hundreds or thousands of reverse diffusion steps to reach state-of-the-art performance [1, 14]. Despite the fact that conditional models with strong conditioning typically require significantly less steps to produce samples of decent quality [3, 6], they still cannot compete with conventional generative modeling frameworks such as Generative Adversarial Networks (GANs) or Normalizing Flows in terms of inference speed [6].

Since sampling from a diffusion model consists in solving a differential equation corresponding to a reverse diffusion, one possible approach to increasing speed of a diffusion model is to employ differential equation solvers providing good trade-off between step sizes of a numerical scheme and discretization errors they lead to. Both general-purpose [15, 16] and diffusion-specific [9, 17, 18] numerical solvers have been considered in the literature. Among other popular approaches to making diffusion models faster one can mention optimizing a forward diffusion design [19] or treating the process of finding the optimal hyperparameters connected with the sampling procedure as an optimization problem [20, 21].

In this paper, we test various methods of accelerating text-to-speech diffusion-based models. Starting from the vanilla Grad-TTS model proposed in [6], we check whether recently proposed Maximum Likelihood Stochastic Differential Equation (SDE) solver [9] can help to increase its sampling speed without loss of quality. Next, we explore another popular sampling method used in Denoising Diffusion Implicit Models (DDIMs, [19]) in combination with the technique of progressive distillation tailored for this sampling method [22]. Finally, we experiment with a class of approaches combining DPMs with other generative models: GANs [23] and Variational Autoencoders (VAEs) [24].

Our paper has the following structure. In Section 2 we describe the vanilla Grad-TTS and the diffusion model it relies on. Section 3 contains the methods of its acceleration we experiment with. Their performance in terms of efficiency and synthesis quality is discussed in Section 4. We conclude in Section 5.

2. Grad-TTS model

Grad-TTS [6] is a text-to-speech model resembling Glow-TTS [25]. It consists of three modules: the encoder, the duration predictor and the decoder. The main difference between Glow-TTS and Grad-TTS is that a flow-based decoder of the former is replaced with a diffusion model conditioned on the outputs of the encoder aligned with speech frames by means of the duration predictor. We use the official Grad-TTS implementation.[1]

2.1. Encoder and Duration Predictor

Input sequence of phonemes is first processed by the encoder. Its outputs are then passed to the duration predictor trying to predict phoneme durations. During inference, the encoder outputs are repeated according to their predicted durations resulting in rough approximation of target mel-spectrograms which is further refined by the decoder. The duration predictor is trained on the outputs of Monotonic Alignment Search algorithm [25].

The encoder consists of a convolutional pre-net and six Transformer blocks with multi-head self-attention followed by the linear projection layer. The duration predictor is composed of two convolutional layers followed by a projection layer.

2.2. Decoder

The decoder is essentially a Mean-Reverting Variance Preserving (MR-VP) DPM described by two SDEs corresponding to the forward X_t and reverse $\hat{X}_t$ diffusion processes:

$$dX_t = \frac{1}{2}\beta_t(\bar{X} - X_t)dt + \sqrt{\beta_t}d\overrightarrow{W}_t\,, \qquad (1)$$

$$d\hat{X}_t = \left(\frac{1}{2}(\bar{X} - \hat{X}_t) - s_\theta(\hat{X}_t, \bar{X}, t)\right)\beta_t dt + \sqrt{\beta_t}d\overleftarrow{W}_t\,, \qquad (2)$$

---

[1]https://github.com/huawei-noah/Speech-Backbones/tree/main/Grad-TTS

where $\bar{X}$ denotes encoder outputs aligned with speech frames, $t \in [0, 1]$, $\overrightarrow{W}_t$ and $\overleftarrow{W}_t$ are independent forward and backward Wiener processes and β_t is non-negative noise schedule. The score matching network s_θ parameterizes the reverse SDE (2) describing the process of transforming the prior distribution $\mathcal{N}(\bar{X}, \mathrm{I})$ where I is identity matrix into that of target mel-spectrograms. The forward SDE (1) allows for explicit solution:

$$X_t = X_0 e^{-\frac{1}{2} \int_0^t \beta_s ds} + \bar{X}(1 - e^{-\frac{1}{2} \int_0^t \beta_s ds}) + \sqrt{\lambda_t}\xi, \quad (3)$$

where $\lambda_t = 1 - e^{-\int_0^t \beta_s ds}$ and ξ is a sample from $\mathcal{N}(0, \mathrm{I})$. The decoder is trained to minimize the following score matching loss:

$$\mathcal{L}_{sm} = \int_0^1 \mathbb{E}_{X_0, \xi} \left[\|\sqrt{\lambda_t} s_\theta(X_t, \bar{X}, t) + \xi\|_2^2 \right] dt. \quad (4)$$

Once the model is trained, sampling from it is enabled by solving either the reverse SDE (2) or the Ordinary Differential Equation (ODE)

$$d\hat{X}_t = \frac{1}{2} \left(\bar{X} - \hat{X}_t - s_\theta(\hat{X}_t, \bar{X}, t) \right) \beta_t dt \quad (5)$$

backwards in variable t on the unit time interval $[0, 1]$ starting from random variable $\hat{X}_1 \sim \mathcal{N}(\bar{X}, \mathrm{I})$. It can be shown [2] that if the score matching network s_θ is trained till optimality, then backward differential equations (2) and (5) describe the same stochastic processes and the resulting sample $\hat{X}_0$ is close in distribution to the target mel-spectrogram X_0.

The decoder is a UNet-based score matching network s_θ processing mel-spectrograms as $2D$ images at 3 resolutions by 3x3 convolutions with channel numbers 64, 128 and 256.

In [6] it was found optimal to solve the ODE (5) instead of the SDE (2) with first-order Euler scheme and start from the prior $\mathcal{N}(\bar{X}, T^{-1} \mathrm{I})$ with temperature $T = 1.5$. The optimal step size was found to be 0.1 resulting in 10 inference steps.

3. Methods of Acceleration

In this section we describe recently proposed methods of DPM acceleration and their application to Grad-TTS.

3.1. Maximum Likelihood SDE Solver

Maximum Likelihood (ML) SDE solver proposed in [9] is a fixed step first order reverse SDE solver maximizing log-likelihood of the forward diffusion trajectories. In case of MR-VP DPM (1-2) ML solver can be expressed as follows:

$$\gamma_{s,t} = \exp\left(-\frac{1}{2}\int_s^t \beta_u du\right), \quad \mu_{s,t} = \gamma_{s,t}\frac{1 - \gamma_{0,s}^2}{1 - \gamma_{0,t}^2}, \quad (6)$$

$$\nu_{s,t} = \gamma_{0,s}\frac{1 - \gamma_{s,t}^2}{1 - \gamma_{0,t}^2}, \quad \sigma_{s,t}^2 = \frac{(1 - \gamma_{0,s}^2)(1 - \gamma_{s,t}^2)}{1 - \gamma_{0,t}^2}, \quad (7)$$

$$E_\theta(x, \bar{x}, t) = \bar{x} + \frac{1}{\gamma_{0,t}}\left((1 - \gamma_{0,t}^2)s_\theta(x, \bar{x}, t) + x - \bar{x}\right), \quad (8)$$

$$\hat{X}_{t-h} = \mu_{t-h,t}\hat{X}_t + \nu_{t-h,t}E_\theta(\hat{X}_t, \bar{X}, t) \\ + (1 - \mu_{t-h,t} - \nu_{t-h,t})\bar{X} + \sigma_{t-h,t}\xi_t, \quad (9)$$

where $t \in \{1, 1-h, .., h\}$, h is step size and ξ_t are standard normal random variables. Recall that $\bar{X}$ stands for a rough target mel-spectrogram approximation given by aligned encoder outputs. The ML solver was shown to perform better than conventional solvers with the same step size in diffusion-based voice conversion [9]. Note that in the original paper the ML solver for MR-VP DPMs has an alternative formulation equivalent to the one given by the formulae (6-9) that we use in this paper.

3.2. Progressive DPM Distillation

DDIM sampling for MR-VP DPM (1-2) is given by the formula

$$\hat{X}_{t-h} = \bar{X} + \left(E_\theta(\hat{X}_t, \bar{X}, t) - \bar{X}\right)\gamma_{0,t-h} \\ + \left(\hat{X}_t - \bar{X} - \left(E_\theta(\hat{X}_t, \bar{X}, t) - \bar{X}\right)\gamma_{0,t}\right)\frac{\alpha_s}{\alpha_t}, \quad (10)$$

where $t \in \{1, 1-h, .., h\}$, $\alpha_t = 1 - \gamma_{0,t}^2$ and the function E_θ is defined in (8). Informally speaking, $E_\theta(X_t, \bar{X}, t)$ is the best possible guess to what the target mel-spectrogram X_0 is equal given that its rough approximation is $\bar{X}$ and its noisy version (i.e. X_0 diffused up to time t) is X_t.

Recently, the algorithm of progressive distillation has been proposed for DDIM sampling scheme [22]. In k-th round of this progressive distillation, a student model producing samples in $m2^{n-k}$ iterations is trained to replicate the behaviour of a teacher model whose samples are generated in $m2^{n-k+1}$ steps. After each round the resulting student model becomes the teacher model for the next round; the initial model (the vanilla Grad-TTS in our case) plays the role of the teacher model in the first round. After n rounds of distillation the final student model should produce samples in m steps with the quality comparable with that of the initial model taking $m2^n$ steps.

More formally, consider k-th round of distillation and denote the step sizes for the teacher and the student models $h^{(t)} = 2^{-(n-k+1)}/m$ and $h^{(s)} = 2^{-(n-k)}/m = 2h^{(t)}$ respectively. Let $X_{u-2h^{(t)}}^{(t)}$ and $X_{u-h^{(s)}}^{(s)}$ be the results of two steps of DDIM sampling from the teacher model and one step of DDIM sampling from the student model correspondingly given that they both are at the same state X_u at time u. Then the distillation loss is calculated as

$$\mathcal{L}_{dist} = \mathbb{E}_{X_0, u} \left[\frac{1}{\alpha_u} \left\| X_{u-h^{(s)}}^{(s)} - X_{u-2h^{(t)}}^{(t)} \right\|_2^2 \right]. \quad (11)$$

To distill the teacher model into the student model at k-th round, we sample ground truth mel-spectrograms X_0, sample u uniformly from the set $\{h^{(s)}, 2h^{(s)}, .., 1\}$, calculate X_u by the formula (3) and compute the distillation loss (11).

3.3. GAN-based Denoising Sampling

GANs allow to rapidly generate high-quality samples belonging to quite complex distributions, therefore modeling denoising distributions with GANs may lead to speeding up generation process in DPMs. This idea was developed in [23] where DPM reverse diffusion process was parameterized with GAN to get rid of Gaussian assumption for the denoising distribution at each reverse diffusion step and the necessity for a large number of small denoising steps at inference.

Suppose $\bar{x}$ is the rough approximation of the clean mel-spectrogram x_0 given by aligned encoder outputs and x_k is obtained as adding Gaussian noise to x_0 in k steps according to the noise schedule δ_k:

$$Law(x_k|x_{k-1}) = \mathcal{N}(\sqrt{1 - \delta_t}x_{k-1}, \delta_t \mathrm{I}) \quad (12)$$

where timestep $k \in [1, .., N]$ (note than in this framework DPMs are considered discrete-time Markov chains rather than continuous-time stochastic processes as in all other sections). In this setting, the discriminator $D_\phi(x_{k-1}, x_k, k)$ with parameters ϕ estimates the probability of x_{k-1} being a version of a noisy sample x_k denoised up to timestep $k - 1$. The generator tries to fool the discriminator and to produce realistic samples $\tilde{x}_{k-1}$

by first generating $\tilde{x}_0 = G_\theta(x_k, \bar{x}, k)$ and then adding noise to it according to the distribution $Law(x_{k-1}|x_k, x_0)$ which, as follows from (12), is also Gaussian. Non-saturating GAN loss function is used for training in [23] leading to good results on image synthesis with as few as $N = 4$ diffusion steps. As in [8], we apply denoising diffusion GAN to text-to-speech, but within the base model of Grad-TTS. The general approach is illustrated in Figure 1.
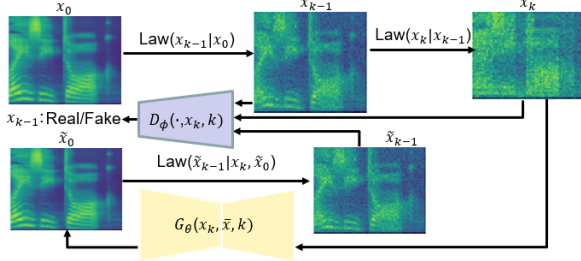

Figure 1: *Denoising diffusion GAN for text-to-speech.*

The encoder and the duration predictor are the same as in the vanilla Grad-TTS model. These modules are frozen during the denoising diffusion GAN training. The generator follows Grad-TTS architecture but with twice more convolutional channels. Similar to [8] latent variable $z \sim \mathcal{N}(0, I)$ isn't passed to the generator as in standard unconditional GANs because we condition our generator on the aligned encoder output $\bar{x}$ as in the vanilla Grad-TTS. Mean Square Error between ground truth mel-spectrograms x_0 and reconstructed mel-spectrograms $\tilde{x}_0$ is added to the adversarial loss of the generator.

The discriminator is similar to the downsampling part of the Grad-TTS decoder followed by the mini-batch standard deviation layer [26], global sum pooling and the fully connected layer. As in [23] all activation functions are replaced with Leaky ReLU with a negative slope 0.2. Also, in addition to adversarial loss we adopt regularization term which penalizes large gradients of the discriminator with respect to its input.

3.4. Score-based Generative Modelling in Latent Space

Iterative sampling from a DPM in data space may be inefficient if data dimensionality is high. To alleviate this problem, the authors of [24] proposed to implement DPM directly in the low-dimensional latent space of VAE. This approach may lead to performing significantly less floating-point operations at inference while joint training of VAE and DPM makes the prior distribution of VAE richer which results in improved sample quality. In what follows, we will refer to this approach as *Latent Score-based Generative Modelling* (LSGM).

3.4.1. General Description

Similar to [24] our LSGM-based decoder utilizes multi-scale hierarchical NVAE [27] as a backbone model. Given a set of latent codes $\{z^l\}_{l=1}^L$ where L is the number of scales, NVAE can be represented as a generative model $p(X, z) = p(z)p(X|z)$ with hierarchical prior $p(z) = \prod_l p(z^l|z^{<l})$. NVAE is trained as a common VAE employing hierarchical approximate posterior $q(z|X) = \prod_l q(z^l|z^{<l}, X)$ by minimizing a negative lower bound $\mathcal{L}_{\text{NVAE}}$ on data log-likelihood $\log p(X)$:

$$\mathcal{L}_{\text{NVAE}} = \mathbb{E}_{q(z|X)} \left[-\log p(X|z) \right] + \mathcal{L}_{KL} \\ + D_{\text{KL}} \Big(q(z^1|X) || p(z^1) \Big) \qquad (13)$$

$$\mathcal{L}_{KL} = \sum_{l=2}^L D_{\text{KL}} \Big(q(z^l|z^{<l}, X) || p(z^l|z^{<l}) \Big) \qquad (14)$$

where $q(z^{<l}|X) = \prod_{i=1}^{l-1} q(z^i|z^{<i}, X)$, and $q(z^l|z^{<l}, X)$ uses residual normal distribution parametrization [27].

As for the DPM part of this model, it operates only on the bottom-level latent space z^1. As in the vanilla Grad-TTS, we chose MR-VP DPM (1-2) for z^1-space of NVAE: its encoder first transforms aligned encoder outputs $\bar{X}$ into $\bar{z}^1$ which serves as the mean for the prior of this DPM, i.e. $z_1^1 \sim \mathcal{N}(\bar{z}^1, I)$ where $\{z_t^1\}_{t \in [0,1]}$ stands for the diffusion process in the latent space z^1. By analogy with the vanilla Grad-TTS, such DPM prior choice should lead to less iterations necessary to get to the target latent code z_0^1 compared to Gaussian prior with zero mean. Also, similar to [24] we set LSGM prior density at any time t to a geometric mixture $p(z_t^1|\bar{z}^1) \propto \varphi(z_t^1|0, I)^{1-\alpha} p_\theta(z_t^1|\bar{z}^1)^\alpha$ where φ is the probability density function of the Gaussian distribution, p_θ is the probability density function of the noisy latent variable z_t^1 and θ denotes the parameters of the score matching network s_θ having the same UNet-based architecture as in the vanilla Grad-TTS. Following [24] we make $\alpha \in [0, 1]$ a trainable parameter allowing the model to decide whether it needs more simple prior (α close to 0) or more complex one (α close to 1) during training. In principle, in the former case LSGM model can perform faster DPM inference to obtain z_0^1 because discretization errors produced by fast sampling schemes have less impact on the geometric mixture $\varphi(z_t^1|0, I)^{1-\alpha} p_\theta(z_t^1|\bar{z}^1)^\alpha$ for smaller values of α.

To sum up, at inference the text encoder and the duration predictor borrowed from the vanilla Grad-TTS described in Section 2 produce rough mel-spectrogram reconstruction $\bar{X}$ which is then passed to the NVAE encoder to get normal distribution parameters for latent layers $l = 2, .., L$ and the bottom-level latent code $\bar{z}^1$. Then, we sample z_0^1 from the MR-VP DPM with the prior $\mathcal{N}(\bar{z}^1, I)$ described above. In our experiments we found it optimal to use 4 steps of the Maximum Likelihood SDE solver described in Section 3.1. Finally, we pass z_0^1 together with $\{\bar{z}^l\}_{l=2}^L$ sampled from normal distributions parameterized with the NVAE encoder outputs into the NVAE decoder to reconstruct a mel-spectrogram.

In our experiments we construct fully-convolutional $L = 4$ latent scale NVAE. On each scale we apply a downsampling layer, pass the resulting feature map into the residual block consisting of N $2D$-dimensional convolutions with d channels ($N \in \{20, 10, 5, 2\}$ and $d \in \{32, 64, 128, 256\}$ for each scale from top to down respectively) and apply Squeeze and Excitation (SE) activation [28]. Downsampling is performed in such a way that a mel-spectrogram of a shape $1 \times 80 \times F$ becomes $256 \times 20 \times \frac{F}{16}$ feature map where F is the number of acoustic frames. Next, we apply a final projection to get the bottom-level latent code $z^1 \in \mathbb{R}^{64 \times \frac{F}{16}}$. Thus, our DPM model operates on latent space whose dimensionality is $\times 20$ smaller than data dimensionality. The NVAE decoder pipeline is the same as the encoder one but in reverse order with additional convolutional modules at each scale related to conditionals $q(z^l|z^{<l}, X)$.

3.4.2. Training details

Once the text encoder and the duration predictor are trained in the way described in Section 2.1 and frozen after that, LSGM text-to-speech model optimization is performed in two stages discussed next: 1) the NVAE pre-training; 2) joint training of the NVAE and the DPM on the latent space.

During the NVAE pre-training we omit the last term in the

traditional NVAE loss $\mathcal{L}_{\text{NVAE}}$ (13) thus not specifying the bottom-level prior $p(z^1)$ since the next stage of LSGM training assumes that it could finally become any complex distribution.

The second stage of LSGM training is the joint NVAE and DPM optimization with the following loss:

$$\mathcal{L}_{\text{joint}} = \mathbb{E}_{q\left(z_0^1|X\right),q\left(\bar{z}^{>1}|\bar{X}\right)}\left[-\log p\left(X|z_0^1, \bar{z}^{>1}\right)\right] + \mathcal{L}_{KL}$$
$$+ \frac{1}{2}\mathbb{E}_{t\sim r(t)}\left[C_t\mathbb{E}_{q(z_0^1|X),\xi}\|\sqrt{\lambda_t}s_\theta\left(z_t^1, \bar{z}^1, t\right) + \xi\|_2^2\right],$$

$$(15)$$

where X is target mel-spectrogram and z_t^1 is sampled according to the analogue of the formula (3) for the latent space z^1 with the stochastic term $\sqrt{\lambda_t}\xi$. We adapt importance sampling necessary for variance reduction of the score-matching loss as proposed in [24]: time t is sampled from the proposal distribution $r(t) \propto \frac{1}{\lambda_t}\frac{d\lambda_t}{dt} = \frac{d\log\lambda_t}{dt}$, and $C_t = \frac{\log\lambda_1 - \log\lambda_\varepsilon}{1-\lambda_t}$ for $\varepsilon \approx 0$ is a weight in importance sampling.

4. Experiments

We experimented with Grad-TTS and its several modifications and tested their performance both in terms of synthesized speech quality as evaluated by Mean Opinion Score (MOS) and efficiency. We also added two baseline models in the comparison: *Glow-TTS* [25] and *FastSpeech2* [29]. We used the pre-trained checkpoint from the official repository for *Glow-TTS* and the open-source implementation available at GitHub[2] for *FastSpeech2*. All the models under comparison were trained on a single-speaker LJSpeech dataset [30]. We plan to release[3] the models and the code for training and inference for Grad-TTS modifications introduced in Section 3.

We checked the vanilla version of Grad-TTS with 10 reverse diffusion steps *Grad-TTS-Vanilla-10*, the same model with 4 steps of the Maximum Likelihood solver *Grad-TTS-ML-4* and its distilled version with 2 steps of DDIM sampling scheme *Grad-TTS-Distilled-2*. Distillation was performed in 5 rounds (reducing the number of reverse diffusion iterations from 2^6 to 2) for 35 epochs each with Adam optimizer, initial learning rate 10^{-5} and batch size 16. The NVAE in *LSGM-NVAE* text-to-speech model was pre-trained with Adam optimizer with learning rate $5 \cdot 10^{-5}$ on mini-batches of size 8 with KL warmup for approximately 500k iterations. Joint training of the NVAE and the DPM of *LSGM-NVAE* was done with Adam optimizer with learning rate 10^{-5} and mini-batches of size 8 for around 530k iterations. The diffusion part in *LSGM-NVAE* employed the ML solver with 4 steps. Denoising diffusion *GAN-DPM* was trained for 150 epochs with batch size 28. Learning rates for Adam optimizer for the generator and the discriminator were initialized with $1.6 \cdot 10^{-4}$ and 10^{-4} correspondingly and decreased with cosine learning rate decay during training. *GAN-DPM* was trained to take $N = 4$ diffusion steps at inference. All the models mentioned above used the same encoder and duration prediction modules. It also holds for the baseline models except for *FastSpeech2* having the encoder with slightly less Transformer blocks and no pre-net and pitch and energy predictors in addition to the duration predictor.

Amazon Mechanical Turk was used for MOS evaluation. Only Master assessors completed listening tests in which they were asked to assess the overall speech quality on 5-point Likert scale with a step of 0.5. The total number of unique participants

---

Table 1: *Comparison of various models*

Model	MOS	RTF	Size
Grad-TTS-Vanilla-10	4.00 ± 0.09	0.68	14.8m
Grad-TTS-ML-4	3.93 ± 0.09	0.28	14.8m
Grad-TTS-Distilled-2	3.82 ± 0.09	0.15	14.8m
GAN-DPM-4	3.71 ± 0.10	0.65	64.1m
LSGM-NVAE-ML-4	3.36 ± 0.11	0.18	24.1m
FastSpeech2	3.58 ± 0.11	0.02	34.6m
Glow-TTS	3.35 ± 0.10	0.03	28.6m
Ground truth	4.38 ± 0.07	—	—

was 20. Each model synthesized 20 sentences from the test set. As for efficiency comparison, it was made by estimating Real Time Factor (RTF is equal to how many seconds are necessary to synthesize 1 second of speech) and model size. RTF was calculated on 2.60GHz Intel CPU. The demo page[4] contains some speech samples used in subjective evaluation.

Table 1 demonstrates the results of our experiments. We can conclude that two best performing methods of increasing model efficiency (both in terms of inference speed and model size) are the ones connected with alternative differential equation solvers, and although applying the maximum likelihood solver with 4 reverse diffusion steps leads to the best performance, the progressive distillation scheme allows to reduce the number of reverse diffusion steps to 2 (equivalent to 0.15 RTF) while keeping synthesis quality still better than that of the baseline models. The model *LSGM-NVAE* combining VAE and Grad-TTS shows a significant quality drop, however still being comparable with *Glow-TTS*. Although each iteration of *LSGM-NVAE* requires less time compared to the vanilla Grad-TTS, it cannot produce high quality samples in a few steps. It corresponds well with the observation made in [23] that LSGMs cannot solve the problem of approximating non-Gaussian denoising distributions as well as GAN-based diffusions do. Indeed, one can see that *GAN-DPM* has MOS much higher than *LSGM-NVAE*. However, we found experimentally that denoising diffusion GAN generated high-quality speech only for generators of sufficiently large sizes. Thus, despite being able to use only 4 steps at inference, *GAN-DPM* is still slow and does not lead to significant speedup compared to the vanilla Grad-TTS.

5. Conclusion

In this paper, we have conducted a thorough investigation on inference acceleration methods for Grad-TTS, a text-to-speech model with diffusion generative engine. One of the major downsides of the DPM approaches is their slow inference speed making them an unappealing choice for practical applications. We tested the most promising techniques of making DPMs more efficient such as employing various differential equation solvers and combining Grad-TTS with GANs and VAEs. The experiments we made demonstrate that the methods focusing on diffusion-specific solvers yield better results in terms of the trade-off between overall quality of audio samples and inference speed. In particular, the progressive distillation of Grad-TTS leads to the model of high quality generating speech approximately 7x faster than real time on CPU.

---

[2]https://github.com/ming024/FastSpeech2

[3]https://github.com/huawei-noah/Speech-Backbones [coming soon]

[4]https://fast-grad-tts.github.io

6. References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* Curran Associates, Inc., 2020, vol. 33.

[2] Y. Song, J. Sohl-Dickstein, D. P. Kingma *et al.*, "Score-Based Generative Modeling through Stochastic Differential Equations," in *International Conference on Learning Representations*, 2021.

[3] N. Chen, Y. Zhang, H. Zen *et al.*, "WaveGrad: Estimating Gradients for Waveform Generation," in *International Conference on Learning Representations*, 2021.

[4] Z. Kong, W. Ping, J. Huang *et al.*, "DiffWave: A Versatile Diffusion Model for Audio Synthesis," in *International Conference on Learning Representations*, 2021.

[5] S. Lee, H. Kim, C. Shin *et al.*, "PriorGrad: Improving Conditional Denoising Diffusion Models with Data-Dependent Adaptive Prior," in *International Conference on Learning Representations*, 2022.

[6] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 8599–8608.

[7] M. Jeong, H. Kim, S. Cheon *et al.*, "Diff-TTS: A Denoising Diffusion Model for Text-to-Speech," in *Proc. Interspeech 2021*, 2021, pp. 3605–3609.

[8] S. Liu, D. Su, and D. Yu, "DiffGAN-TTS: High-Fidelity and Efficient Text-to-Speech with Denoising Diffusion GANs," 2022. [Online]. Available: http://arxiv.org/abs/2201.11972

[9] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, "Diffusion-Based Voice Conversion with Fast Maximum Likelihood Sampling Scheme," in *International Conference on Learning Representations*, 2022.

[10] S. Liu, Y. Cao, D. Su, and H. Meng, "DiffSVC: A Diffusion Probabilistic Model for Singing Voice Conversion," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 741–748.

[11] J. Liu, C. Li, Y. Ren, F. Chen, P. Liu, and Z. Zhao, "DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism," 2021. [Online]. Available: https://arxiv.org/abs/2105.02446

[12] J. Lee and S. Han, "NU-Wave: A Diffusion Probabilistic Model for Neural Audio Upsampling," in *Proc. Interspeech 2021*, 2021, pp. 1634–1638.

[13] J. Tae, H. Kim, and T. Kim, "EdiTTS: Score-based Editing for Controllable Text-to-Speech," 2022. [Online]. Available: https://arxiv.org/abs/2110.02584

[14] P. Dhariwal and A. Q. Nichol, "Diffusion Models Beat GANs on Image Synthesis," in *Advances in Neural Information Processing Systems*, 2021.

[15] L. Liu, Y. Ren, Z. Lin, and Z. Zhao, "Pseudo Numerical Methods for Diffusion Models on Manifolds," in *International Conference on Learning Representations*, 2022.

[16] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer *et al.*, "Gotta Go Fast When Generating Data with Score-Based Models," *ArXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2105.14080

[17] F. Bao, C. Li, J. Zhu, and B.Zhang, "Analytic-DPM: An Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models," in *International Conference on Learning Representations*, 2022.

[18] H. Tachibana, M. Go, M. Inahara *et al.*, "Itô-Taylor Sampling Scheme for Denoising Diffusion Probabilistic Models using Ideal Derivatives," *ArXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2112.13339

[19] J. Song, C. Meng, and S. Ermon, "Denoising Diffusion Implicit Models," in *International Conference on Learning Representations*, 2021.

[20] D. Watson, J. Ho, M. Norouzi, and W. Chan, "Learning to Efficiently Sample from Diffusion Probabilistic Models," 2021. [Online]. Available: https://arxiv.org/abs/2106.03802

[21] R. San-Roman, E. Nachmani, and L. Wolf, "Noise Estimation for Generative Diffusion Models," 2021. [Online]. Available: https://arxiv.org/abs/2104.02600

[22] T. Salimans and J. Ho, "Progressive Distillation for Fast Sampling of Diffusion Models," in *International Conference on Learning Representations*, 2022.

[23] Z. Xiao and K. Kreis and A. Vahdat, "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs," in *International Conference on Learning Representations*, 2022.

[24] A. Vahdat, K. Kreis, and J. Kautz, "Score-based Generative Modeling in Latent Space," in *Advances in Neural Information Processing Systems*, 2021.

[25] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 8067–8077.

[26] T. Karras, S. Laine, M. Aittala *et al.*, "Analyzing and Improving the Image Quality of StyleGAN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[27] A. Vahdat and J. Kautz, "NVAE: A Deep Hierarchical Variational Autoencoder," in *Advances in Neural Information Processing Systems*, 2020.

[28] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[29] Y. Ren, C. Hu, X. Tan *et al.*, "FastSpeech 2: Fast and High-Quality End-to-End Text to Speech," in *International Conference on Learning Representations*, 2021.

[30] K. Ito, "The LJ Speech Dataset," 2017. [Online]. Available: https://keithito.com/LJ-Speech-Dataset/