Diffusion Model for Music Synthesis

Faezeh Pouya Mehr [1] Hena Ghonia [1] Krishna Maneesha Dendukuri [1] Mahdi Kleit [1]

Abstract

In this work, we explore Denoising Diffusion for Implicit Model for multi-instrument music datasets. We use a simple two-stage process: an audio file (.wav file) is first converted to a spectrogram which is then used as input to a U-net model and then we convert the generated spectrograms to audio using a pretrained generative adversarial network (GAN) spectrogram inverter. In order to measure the performance of our implementation, we make use of the Fréchet Audiso Distance to evaluate our generations with respect to the real dataset. We then extend our experiments to Indian classical music using the Indian Classical Carnatic dataset. We show that even though diffusion models are considered to be slow generating methods, we are still able to generate high quality samples without needing relatively long training times.

We have provided the link for the code repository used to carry out all our experiments. We also provide some generated music samples.

1. Introduction

Audio generation algorithms, known as vocoders in text-to-speech (TTS) systems and synthesizers in music, respond to higher-level control signals to create fine-grained audio waveforms which is a demanding task for a considerable range of applications such as text-to-speech synthesis, music synthesis, and voice conversion. The special difficulty of audio generation is that there is often a very large difference between the dimensionality of the raw audio signal and that of the effective semantic-level signal.

Neural audio synthesis first proved feasible with autoregressive models of raw waveforms such as WaveNet (Van den Oord Aaron et al., 2021) and SampleRNN (Mehri et al., 2016). While these models can render a broad range of realistic audio, using them as expressive musical instruments is not straightforward as they need fine-grained domain-specific conditioning information, such as phonemes (Oord et al., 2016) or mel-spectrograms (Skerry-Ryan et al., 2018) for speech, and are prohibitively slow at rendering audio due to requiring to run a forward pass for every sample of the waveform.

Techniques to overcome these speed restrictions of waveform autoregression have concentrated on generating audio directly with a single forward pass. These models often focus on limited domains such as generating a single instrument/note/voice at a time (Wu et al., 2021; Manzelli et al., 2018; Kim et al., 2019; Défossez et al., 2018).

Researchers have also overcome temporal context limitations of waveform autoregression by adopting a multistage approach, first creating coarser audio representations at a lower sample rate, and then modeling those representations with a predictive model before decoding them back into audio (Dhariwal et al., 2020; Zeghidour et al., 2021; Hawthorne et al., 2022a).

The success of this approach led to an outbreak of research into spectrogram inversion models, including streamlined waveform autoregression, GANs, normalizing flows, and Denoising Diffusion Probabilistic Models (DDPMs).

Diffusion models can use a diffusion (noise-adding) process without learnable parameters to obtain the "whitened" latent from training data. Therefore, no extra neural networks are required for training in contrast to other models (e.g., the ecoder in VAE (Welling & Kingma, 2014) or the discriminator in GAN (Goodfellow et al., 2020). This avoids the problematic "posterior collapse" or "mode collapse" issues originating from the joint training of two networks and hence is valuable for high-fidelity audio synthesis.

DDPMs and Score-based Generative Models (SGMs) have proven well-suited to generating audio representations and raw waveforms. Speech researchers have shown high-fidelity spectrogram inversion kong2020diffwave, chen2020wavegrad, text-to-speech (Popov et al., 2021a; Jeong et al., 2021), upsampling (Lee & Han, 2021), and voice conversion (Kameoka et al., 2020; Popov et al., 2021b).

[*]Equal contribution [1]Group 12, University of Montreal, MILA, Canada. Correspondence to: Name <name.lastname@umontreal.ca>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

2. Literature Review

We study following paper to understand state of art research on Diffusion models for music synthesis.
Denoising Diffusion Probabilistics Models
Diffusion probabilistic (Ho et al., 2020b) models are a class of latent variable models. A diffusion probabilistic model is a parameterized Markov chain trained to produce samples matching the data after some time T. Training is performed by optimizing the variational bound on negative log likelihood. The process consists of two distinct processes ; the first process, the forward diffusion process q adds Gaussian noise to an image until only pure noise remains. The approximate posterior $q(x_{0:T})$ is fixed to a Markov chain that gradually adds noise. Noise is added according to a variance schedule β_1,$\beta_2 \ldots$, β_T. Variances β_t of the forward process can also be learned by reparameterization and thus are not the same at every t. In this paper, they fixed them to constants for the sake of experimentation - meaning that the posterior has no learnable parameters and thus is a constant that can be ignored. The second process, the reversed denoising process p, is trained to denoise an image starting from the pure noise generated by the forward pass. It is expressed as a joint distribution $p_\theta(x_{0:T})$ where the first element $p(x_T)$ is obtained from a standard Gaussian distribution. Every other image is then sampled from a conditional Gaussian probability with parameters Σ_θ and μ_θ. The authors decided to keep the variance Σ_θ fixed and have the network learn only μ_θ of the conditional probability distribution.

Symbolic music generation with diffusion models[(Mittal et al., 2021)]
In this paper, authors use MusicVAE to encode discrete sequences of notes which results in continuous latent embedding(z_k). As Sequence VAE is difficult to train , authors use 2-bar phrases using the pre-trained 2-bar melody MusicVAE. Thereafter, linear feature scaling is performed which ensures consistently scaled input embeddings. These latent embeddings and positional encoding(sinusoidal encoding)are input to the Transformer diffusion model. Author's demonstrate post-hoc conditioning(or the Infilling procedure) which extends sampling procedure which is useful for artists without the computational resources to modify or re-train deep models for new tasks.

Multi-Instrument Music Synthesis using Spectrogram Diffusion[(Hawthorne et al., 2022b)]
Multi-Instrument Music Synthesis using Spectrogram Diffusion[3] The authors use a two-staged system for the music synthesis problem. The first stage involves converting the note sequence of MIDI-like note events into spectrograms. The second stage is to then invert these spectrograms to yield audio samples.For the first stage, they use an encoder-decoder Transformer architecture. The encoder here takes in the note sequences and returns the embeddings to the decoder which generates a spectrogram corresponding to the input note sequence. Also, in order to ensure smooth transitions between the audio segments, a second encoder is introduced which passes the spectrogram from the previous (timestep) segment as a context for the current segment.For the decoder part, the authors explore an autoregressive decoder and a DDPM-based decoder, and find that DDPM approach is superior both qualitatively and as measured by audio reconstruction and Fréchet distance metric. For the second stage they use generative adversarial network (GAN) spectrogram inverter to translate the model's magnitude spectrogram output to audio

3. Method details

3.1. Architecture

3.1.1. U-Net

As discussed earlier, the architecture of the model used Hawthorne et al. (2022b) consists of an encoder-decoder Transformer and of two phases. In our implementation, a wav file is converted to a spectogram and to it is then concatenated noise according to a diffusion schedule. From there, the input is fed into a U-net and the mean squared error loss is computed and backpropagated as shown in 1. The main difference between our implementation and the one by Hawthorne et al. (2022b) is the use of a U-net, in a similar fashion to Ho et al. (2020a).

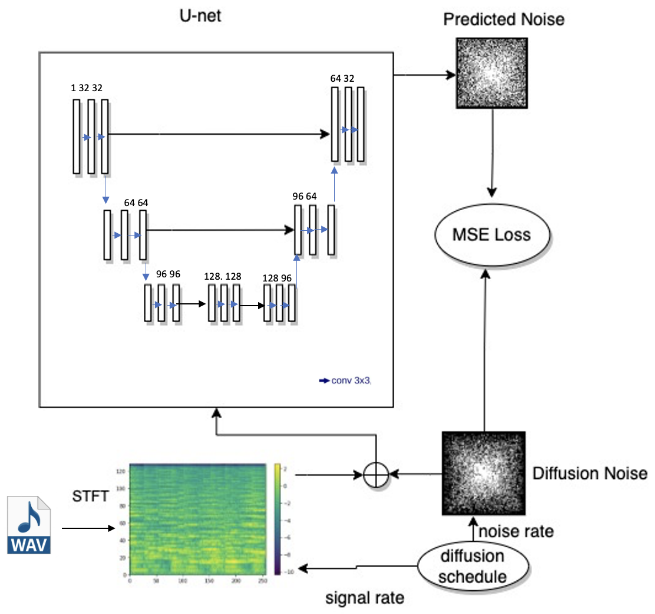

Figure 1. Model Architecture

A U-Net model consists of a bottleneck that makes sure that the network only learns the most important information.

It works by downsampling the input and then upsampling it. The use of a U-Net has been motivated by the improvements observed in ResNet (He et al., 2015) that resulted in improved gradients flows due to the introduction of residual connections between the encoder and the decoder. In our implementation, an attention layer is applied after every Up Block.

3.1.2. Diffusion models

Diffusion models were first introduced by Sohl-Dickstein et al. (2015) as a novel approach in the realm of deep generative models. Since then, progress has been made first by Ho et al. (2020a) and after by Song et al. (2020). Ho et al. (2020a) proposed a framework - DDPM - (Denoising Diffusion Probabilistic Models) in order to show that these models had the capabilities to produce high quality samples, some that are even better than the ones produced by other generative models. Yet, simulating a Markov chain for many steps in order to produce a sample resulted in relatively long sampling ; Song et al. (2020) presented a more efficient alternative to DDPM: DDIM (Denoising Diffusion Implicit Models). DDIM were built on non-Markovian diffusion processes, corresponding to deterministic generative processes that led to implicit models. The main difference had to do with the sampling method used - DDIM uses deterministic sampling whereas DDPM uses stochastic sampling, meaning that part of the predicted noise is replaced with random noise.

Let us express their differences in a more mathematical form. First, we express the forward process of a diffusion model as $q(x_t|x_{t-1}) := \mathcal{N}(x_t; \mu_t, \Sigma_t)$ with $\mu_t = \sqrt{1-\beta_t}x_{t-1}$ and $\Sigma_t = \beta_t\mathbf{I}$. The β_t is the variance schedule which adds Gaussian noise at each time step t. This forward process can also be re-written in order to condition every distribution $q(x_t)$ on x_0 instead of the previous datapoint x_{t-1}. We obtain $q(x_t|x_0) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}}x_0, (1-\bar{\alpha})\mathbf{I})$ with $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. The reverse process is defined as $p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ with θ being the parameters of the neural network, which are updated by gradient descent. The estimation of x_{t-1} is done through $p_t(x_{t-1}|x_t) = \frac{q_t(x_t|x_{t-1})p_t(x_{t-1})}{p_t(x_t)}$. However, $p_t(x_{t-1})$ and $p_t(x_t)$ being both intractable, we can condition our estimation on x_0 and thus we obtain

$$p_t(x_{t-1}|x_t, x_0) = \frac{q_t(x_t|x_{t-1}, x_0)q_t(x_{t-1}|x_0)}{q_t(x_t|x_0)}$$

giving us the following distribution

$$p(x_{t-1}|x_t, x_0) := \mathcal{N}(x_{t-1}, \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t\mathbf{I})$$

where $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$.

In the context of a DDIM, the new sampling schedule is defined as $\{\tau_1, ..., \tau_S\}$ with $S < T$. This allows us to revisit the forward process by parameterizing it with respect to a desired standard deviation σ_t and obtain $p_\sigma(x_{t-1}|x_t, x_0) := \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})$. This means that, according to the definition of $\tilde{\beta}_t$, we have $\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$. If we let $\sigma_t^2 = \eta \cdot \tilde{\beta}_t$ then setting $\eta = 0$ would make the sampling process *deterministic* and thus would result in a DDIM. The inference process becomes $p_{\sigma,\tau}(x_{\tau_{i-1}}, |x_{\tau_t}, x_0)$ defined by:

$$\mathcal{N}(x_{\tau_{i-1}}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_{\tau_i}-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})$$

3.2. Training details

Training samples are in form of .wav or .mp3 format which are converted to Spectrogram. The frequency content of a sound clip can be represented graphically through the use of an audio spectrogram. Time is shown along the x-axis, and frequency along the y-axis. Each pixel's hue represents the relative volume of the sound at the time and frequency indicated by the pixel's row and column. Along with spectrogram, its sinosuidal embedding and noise (initially randomly sampled) which is denoted as Diffusion noise in 1 is passed in U-net architecture. Noise loss i.e mean square error is calculated between predicted noise and the noise added during that time step of the forward Diffusion process. Similarly reconstruction loss(mean square error) is calculated between reconstructed data and spectrogram input during each iteration. This process happens during reverse diffusion. While generating music samples we run learned model(U-net) over training samples for inference time steps(S) and pass it to pre-trained GAN to invert spectrogram into audio(.wav) files.

3.3. Datasets

We explored a variety of music datasets to experiment with, varying across the different properties of music data such as sampling rate, timbre, genre of music etc. These included both single instrument based [guitarset REFS TO DO] and multi-instrument based datasets[GTZAN, SaRaGa REFS TO DO]. Due to memory and compute challenges, we trained the model on a smaller subsets of each of the datasets mentioned. For Guitarset[(Xi et al., 2018)] that contains Guitar (only) samples, we trained on 50 44100Hz Mono samples. GTZAN [(Tzanetakis et al., 2001)] consists of 1000 multi-instrument based audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. We hence only trained on one of the genres(classical). The tracks are all 22050Hz Mono 16-bit audio files in .wav format. In the case of MAESTRO[(Hawthorne et al., 2019)]; which contains a Piano(only)-based samples, was trained on 234 samples in total. Finally we also test the model with an Indian Classical music dataset[(Gulati et al., 2014)]. To be specific, these were Carnatic-based (originally from South India) multi-instrument 44100 Hz music samples. We based majority of our experiments on the GTZAN dataset as it has relatively lower resolution and hence requires lesser pre-processing steps.

3.4. Evaluation Metric: Frechet Audio Distance

We use Frechet Audio Distance proposed in (Kilgour et al., 2019). Compared to other audio evaluation measures, FAD compares embedding statistics generated on the entire evaluation set with embedding statistics generated on a significant set of clean music instead of focusing on specific audio segments. As a result, FAD becomes a reference-free free metric that may be used to evaluate a set of data without access to the ground truth reference audio.

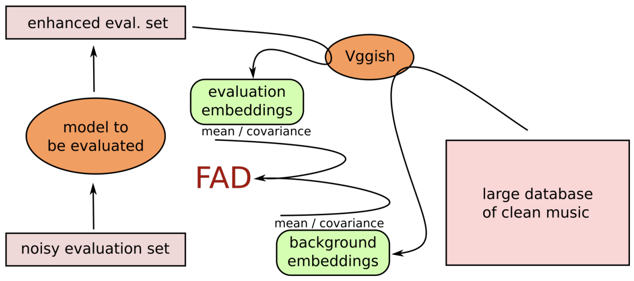

Figure 2. Architecture of Frechet Audio Distance calculation (Kilgour et al., 2019)

As shown in 2 once we have generated samples we pass it to pre-trained VGGish((Hershey et al., 2017)) model which was trained on a large dataset of YouTube videos, similar to YouTube-8M((Abu-El-Haija et al., 2016)) as an audio classifier with over 3000 classes. Also, original training set is passed to pre-trained VGGish model and then on both embeddings obtained from trained and evaluations, its distribution is obtained by calculating the mean and covariance. Thereafter distance between distribution is calculated using below formula: where $\mathcal{N}_e(\mu_e, \Sigma_e)$ is evaluation set embeddings and $\mathcal{N}_b(\mu_b, \Sigma_b)$ embeddings obtained from training set and tr is the trace of a matrix. Lower the value of FAD, better the model performs.

$$\mathbf{F}(\mathcal{N}_b, \mathcal{N}_e) = \|\mu_b - \mu_e\|^2 + \operatorname{tr}\left(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e}\right)$$

3.5. Hyper parameter tuning

Hyperparameters such as learning rate, weight decay, minimum signal rate and maximum signal rate used in Beta cosine schedule, block depth for U-net architecture were tuned.

We used the optuna library (Akiba et al., 2019) which provides effective use of the searching and pruning techniques and enables lightweight experimentation with an interactive interface, providing scale distributed computing. We ran for 6 trials and the best hyperparameters obtained were:

$learning\_rate \rightarrow 3.310677931449062^{-5}$
$weight\_decay \rightarrow 0.06881110177645029$
$min\_signal\_rate \rightarrow 0.1952300573099572$
$max\_signal\_rate \rightarrow 0.9326085838636291$
$block\_depth \rightarrow 4$

4. Experiment results

Dataset -size	audio length	#epochs	S	batch size	FAD Score
GTZAN-100	5.12s	100	1000	16	6.129
GTZAN-100	5.12s	200	1000	16	7.875
GTZAN-10	5.12s	200	1000	1	5.00
GTZAN-10	10.24s	200	1000	1	5.011
GTZAN-50	5.12s	200	1000	4	2.992
GTZAN-50	5.12s	200	500	4	3.715
GTZAN-50	5.12s	200	100	4	3.905
Carnatic-159	5.12s	100	1000	16	7.030
Carnatic-50	5.12s	200	1000	4	7.342

All of the experiments ran for this project were done on Google Colab using a standard GPU.

Given the belief that diffusion models are more 'faithful' to the data and also tend to memorize the data, we ran some experiments to test if DDIM can overfit on a small dataset of size 10 samples. We tried this with varied input audio lengths, batch sizes and number of inference time steps. Eventually, we observed that a higher number of time steps during inference helped enhance the FAD score. This makes sense given that the inference/sampling consists of starting with random noise and running S number of denoising steps. We also performed a few more experiments with the single instrument based datasets such as GuitarSet and MAESTRO which comparatively have higher resolution. The generations for these datasets were a lot more noisier and returned higher FAD scores.

5. Discussion

In general, we observed that running experiments for more iterations gave better results. Due to limited compute resources, we could not run experiments for more than 200 epochs as they would require unreasonably long time. Even if we were able to obtain encouraging results, they could be improved by adding context. It can be done if we input the spectogram from the previous segment as a context for the current segment while doing the forward pass. This would allow the model to yield better performances as it would learn the structure and the patterns of music and then use that knowledge in order to generate higher quality samples. Looking at 4, we can observe that a scaling of the dataset size as well as the time steps used for training could result in better FAD scores. We can observe that the FAD scores obtained seem to increase as we increase the number of epochs for simpler datasets. Indeed, if we compare the results obtained on GTZAN with a complete dataset (100 samples), we can see that a greater number of epochs results in a greater FAD score. This could mean that our model is overfitting and thus cannot generate higher quality samples than when using a lesser number of epochs. Even though we were able to obtain promising results using a simpler DDIM architecture, we still faced some challenges. As mentioned earlier, sampling is extremely slow for these models making them less attractive than other deep generative models such as GANs. This known issue was initially addressed through DDIM, but these remain yet very slow compared to other models.

Acknowledgements

We acknowledge that we referred to Keras implementation of DDIM for image generation and used tensorflow hub to load pretrained GAN and VGGish model.

References

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Défossez, A., Zeghidour, N., Usunier, N., Bottou, L., and Bach, F. Sing: Symbol-to-instrument neural generator. *Advances in neural information processing systems*, 31, 2018.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Gulati, S., Bellur, A., Salamon, J., HG, R., Ishwar, V., Murthy, H. A., and Serra, X. Automatic tonic identification in indian art music: approaches and evaluation. *Journal of New Music Research*, 43(1):53–71, 2014.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1lYRjC9F7`.

Hawthorne, C., Jaegle, A., Cangea, C., Borgeaud, S., Nash, C., Malinowski, M., Dieleman, S., Vinyals, O., Botvinick, M., Simon, I., et al. General-purpose, long-context autoregressive modeling with perceiver ar. *arXiv preprint arXiv:2202.07765*, 2022a.

Hawthorne, C., Simon, I., Roberts, A., Zeghidour, N., Gardner, J., Manilow, E., and Engel, J. Multi-instrument music synthesis with spectrogram diffusion. 2022b. doi: 10.48550/ARXIV.2206.05408. URL `https://arxiv.org/abs/2206.05408`.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 131–135. IEEE, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020a. URL `https://arxiv.org/abs/2006.11239`.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020b.

Jeong, M., Kim, H., Cheon, S. J., Choi, B. J., and Kim, N. S. Diff-tts: A denoising diffusion model for text-to-speech. *arXiv preprint arXiv:2104.01409*, 2021.

Kameoka, H., Kaneko, T., Tanaka, K., Hojo, N., and Seki, S. Voicegrad: Non-parallel any-to-many voice conversion with annealed langevin dynamics. *arXiv preprint arXiv:2010.02977*, 2020.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *INTERSPEECH*, pp. 2350–2354, 2019.

Kim, J. W., Bittner, R., Kumar, A., and Bello, J. P. Neural music synthesis for flexible timbre control. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 176–180. IEEE, 2019.

Lee, J. and Han, S. Nu-wave: A diffusion probabilistic model for neural audio upsampling. *arXiv preprint arXiv:2104.02321*, 2021.

Manzelli, R., Thakkar, V., Siahkamari, A., and Kulis, B. Conditioning deep generative raw audio models for structured automatic music. *arXiv preprint arXiv:1806.09905*, 2018.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

Mittal, G., Engel, J., Hawthorne, C., and Simon, I. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., and Kudinov, M. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pp. 8599–8608. PMLR, 2021a.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., Kudinov, M., and Wei, J. Diffusion-based voice conversion with fast maximum likelihood sampling scheme. *arXiv preprint arXiv:2109.13821*, 2021b.

Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R., Clark, R., and Saurous, R. A. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In *international conference on machine learning*, pp. 4693–4702. PMLR, 2018.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. URL http://arxiv.org/abs/1503.03585.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020. URL https://arxiv.org/abs/2010.02502.

Tzanetakis, G., Essl, G., and Cook, P. Automatic musical genre classification of audio signals, 2001. URL http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

Van den Oord Aaron, D. S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet, a generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, 2021.

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

Wu, Y., Manilow, E., Deng, Y., Swavely, R., Kastner, K., Cooijmans, T., Courville, A., Huang, C.-Z. A., and Engel, J. Midi-ddsp: Detailed control of musical performance via hierarchical modeling. *arXiv preprint arXiv:2112.09312*, 2021.

Xi, Q., Bittner, R. M., Pauwels, J., Ye, X., and Bello, J. P. Guitarset: A dataset for guitar transcription. In *ISMIR*, pp. 453–460, 2018.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.