BASS ACCOMPANIMENT GENERATION VIA LATENT DIFFUSION

Marco Pasini [1,2], Maarten Grachten [1], Stefan Lattner [1]

Sony Computer Science Laboratories, Paris, France[1] Queen Mary University, London, UK[2]

ABSTRACT

The ability to automatically generate music that appropriately matches an arbitrary input track is a challenging task. We present a novel controllable system for generating single stems to accompany musical mixes of arbitrary length. At the core of our method are audio autoencoders that efficiently compress audio waveform samples into invertible latent representations, and a conditional latent diffusion model that takes as input the latent encoding of a mix and generates the latent encoding of a corresponding stem. To provide control over the timbre of generated samples, we introduce a technique to ground the latent space to a user-provided reference style during diffusion sampling. For further improving audio quality, we adapt classifier-free guidance to avoid distortions at high guidance strengths when generating an unbounded latent space. We train our model on a dataset of pairs of mixes and matching bass stems. Quantitative experiments demonstrate that, given an input mix, the proposed system can generate basslines with user-specified timbres. Our controllable conditional audio generation framework represents a significant step forward in creating generative AI tools to assist musicians in music production.

Index Terms— music, accompaniment, diffusion, generation, bass

1. INTRODUCTION

Musical accompaniment is an integral part of music composition and performance. The ability to automatically generate an accompaniment that complements and matches the style of existing instrument parts (*stems*) in a music track, has the potential to both enhance the creativity of artists–by proposing novel musical material for them to work with–and to make it easier and more efficient to realize their artistic visions. In recent years, deep learning techniques have shown promising results in the field of music and (to a much lesser extent) accompaniment generation. Many approaches use a symbolic representation of music as the medium [1]–[3], while more recently a number of models that directly generate waveform audio have also been proposed [4]–[6]. Diffusion models [7]–[9] have emerged as a powerful class of generative models capable of producing high-quality samples, although they usually require a computationally expensive iterative sampling procedure. Latent diffusion models [10] have been introduced to increase model inference speed by generating a latent, low-dimensional representation of the data from a pretrained autoencoder model, usually a Variational AutoEncoder [11].

In this work, we propose a general latent generative model for the task of accompaniment generation, and apply it to the generation of basslines. Given an input stem of arbitrary length such as a vocal melody or an input mix of arbitrary numbers of stems, our model is able to generate a complementary bass stem that musically matches the conditioning. Furthermore, we propose controllability features, such as style conditioning and conditioning guidance control, to make our system a more useful tool for artists. The key contributions of our work are:

- The design of an efficient audio autoencoder to encode samples to compressed invertible representations
- The design of a general conditional latent diffusion model that takes a music mix as input and produces a coherent track, while being able to handle inputs and outputs of arbitrary length
- The application of both audio autoencoder and latent diffusion model to the task of encoding and generating basslines given an arbitrary input mix
- The use of style conditioning during the diffusion sampling process to force the generation of a user-defined bass style.

2. RELATED WORK

Accompaniment generation is a type of music generation that involves an additional input conditioning. In this work we focus on audio-based music generation. Autoregressive models such as WaveNet [12], SampleRNN [13], Jukebox [4], MusicLM [5] and MusicGen [14] can generate high quality samples but suffer from slow sequential sampling. Non-autoregressive models based on generative adversarial networks (GANs) [15] such as WaveGAN [16] and GANSynth [17] achieve parallel sampling but are limited to generating fixed-length audio clips. On the other hand, Musika [18] parallelly generates invertible latent representations of audio of

arbitrary length, but the context available to the model is limited. Relevant to our work, BassNet [19] generates bass tracks while offering user control via a latent space variable.

More recently, models such as DiffWave [20] and WaveGrad [21] introduce diffusion to audio modeling for speech synthesis applications. For musical audio generation, Riffusion [22] fine-tunes Stable Diffusion [10] on audio spectrograms to generate music clips. Moûsai [23] trains a latent diffusion model on compressed representations and can generate minute-long coherent music. JEN-1 [24] introduces a large-scale conditional latent diffusion model that can generate long-form music both autoregressively and non-autoregressively. Finally, [6] proposes a multi-source diffusion model trained on single source waveforms that achieves both generation and separation of individual sources.

3. METHOD

Let $\mathbf{x} = \{x_1, ..., x_T\}$ be the waveform of a mix of arbitrary stems of length T, where x_i is the i-th stereo frame, and let $\mathbf{y} = \{y_1, ..., y_T\}$ be the waveform of a single-stem audio sample with the same length. To sample $\mathbf{y}$ given $\mathbf{x}$, we aim to model the conditional distribution $p(\mathbf{y}|\mathbf{x})$, but since the waveforms are typically very high-dimensional (i.e. T is large), we encode both $\mathbf{x}$ and $\mathbf{y}$ into latent representations $\mathbf{c_x} = \{c_{x,1}, ..., c_{x,T/r_{time}}\}$ and $\mathbf{c_y} = \{c_{y,1}, ..., c_{y,T/r_{time}}\}$ respectively using audio autoencoders, and model $p(\mathbf{c_y}|\mathbf{c_x})$ instead. Here, r_{time} is the time compression ratio of the autoencoders, and we refer to the dimensionality of vectors $c_{x,i}$ and $c_{y,i}$ as dim_x and dim_y, respectively.

3.1. Audio Autoencoder

Our goal is to design an efficient audio autoencoder that can reach high compression ratios while reconstructing samples with reasonable accuracy. To achieve this, we start from the audio autoencoder architecture proposed in Musika [18], where a model is used to reconstruct the magnitude and phase components of a spectrogram s instead of the full waveform, which results in faster inference. However, instead of using the original two-stage design and two-phase training process, we train a single encoder and decoder in a fully end-to-end fashion. We first use a L1 loss between a log-magnitude spectrogram s and the magnitude output of the model:

$$\mathcal{L}_{E,D,rec} = \mathbb{E}_{s\sim p(s)} ||D(E(s))_{mag} - s||_1$$

where E and D are the encoder and decoder, and $D(E(s))_{mag}$ is the magnitude component of the decoder output. We also use the multi-scale spectral distance [25], [26] between the original and the reconstructed waveforms:

$$\begin{aligned} \tilde{w} &= \text{iSTFT}(D(E(s))) \\ \mathcal{L}_{D,mssd} &= \mathbb{E}_{w\sim p(w)} \sum_{h\in\mathcal{H}} || \,\text{STFT}_h(w)^2 - \text{STFT}_h(\tilde{w})^2 \,||_1 \end{aligned}$$

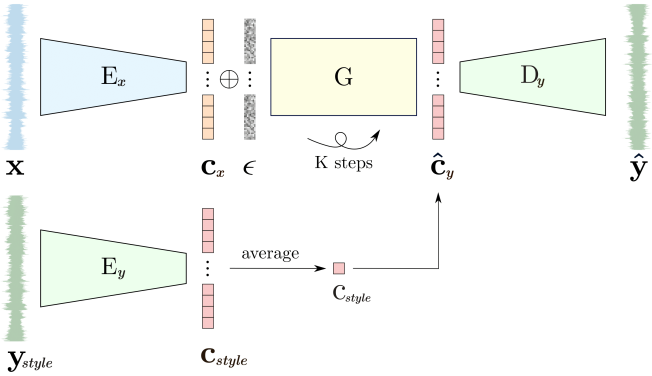

Fig. 1: Inference of the system. Noise is concatenated to the latent representation of the conditioning waveform $\mathbf{c_x}$, and K denoising steps are performed to generate $\hat{\mathbf{c}}_\mathbf{y}$ which is then decoded to waveform. The representation of a user-specified style sample $\mathbf{c_{style}}$ can be used to ground the generated output to a specific style.

where $\mathcal{H}$ is a set of pairs of hop size and window length. The phase component is modelled implicitly by the multi-scale spectral distance loss and the adversarial loss on the log-magnitude spectrogram of the reconstructed waveform:

$$\begin{aligned} \tilde{s} &= \log(\text{STFT}(\tilde{w})^2 + \epsilon) \\ \mathcal{L}_C &= -\mathbb{E}_{s\sim p(s)} \left[\min(0,\ -1 + C(s))\right] \\ &\quad -\mathbb{E}_{s\sim p(s)} \left[\min(0,\ -1 - C(\tilde{s}))\right] \\ \mathcal{L}_{E,D,adv} &= -\mathbb{E}_{s\sim p(s)}\, C(\tilde{s}) \end{aligned}$$

where C is the critic. The final objective used to jointly train encoder and decoder is the following:

$$\mathcal{L}_{E,D} = \mathcal{L}_{E,D,adv} + \lambda_{rec}\mathcal{L}_{E,D,rec} + \lambda_{mssd}\mathcal{L}_{E,D,mssd}$$

Differently from [18], we add a second critic that receives mel-spectrograms. This addition encourages the autoencoder to reconstruct spectral information more accurately in the regions where human pitch perception is more precise.

3.2. Latent Diffusion Model

Diffusion models are trained to reverse a sequential corruption process of samples, and thus are able to retrieve samples from the data distribution by starting from a known distribution and iteratively denoising it. We choose to briefly introduce them with their score-based interpretation [27].

Our goal is to model the score of the conditional target stem latent distribution, given the input mix latent:

$$G_\theta(\mathbf{c_y}, \mathbf{c_x}) \approx \nabla_{\mathbf{c_y}} \log p(\mathbf{c_y}|\mathbf{c_x})$$

where $G_\theta(\mathbf{c_y}, \mathbf{c_x})$ is a neural network with parameters θ.

To achieve this, we minimize the Fisher Divergence between the output of the model and score:

$$\mathbb{E}_{p(\mathbf{c_y},\mathbf{c_x})} \left[\left\| G_\theta(\mathbf{c_y}, \mathbf{c_x}) - \nabla_{\mathbf{c_y}} \log p(\mathbf{c_y}|\mathbf{c_x}) \right\|_2^2 \right]$$

Finally, we can use Langevin dynamics to iteratively generate real samples with a sufficiently large number of iterations K.

In practice, we train our model to denoise noisy latent samples of the target stem $\mathbf{z}_t = \alpha_t \mathbf{c_y} + \beta_t \epsilon$, with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\mathcal{L}_{G_\theta} = \mathbb{E}_{\mathbf{c_y}, \mathbf{c_x} \sim p(\mathbf{c_y}, \mathbf{c_x}), t \sim [0,1]} w_t ||G_\theta(\mathbf{z}_t, t, \mathbf{c_x}) - \mathbf{c_y}||_2^2$$

where α_t and β_t are the signal and noise rates, $\mathbf{c}$ is the latent representation of the corresponding input mix and w_t is the loss weight at timestep t.

The model is based on a U-Net architecture [28], with the addition of self-attention [29] in the lower resolution layers. However, the vanilla self-attention mechanism does not allow the model to generalize to arbitrarily long inputs and outputs [30], which is crucial for a flexible real-world use of the system. To achieve generalization to lengths that are unseen during training, we equip the attention layers with Dynamic Positional Bias (DPB), a technique introduced for the task of arbitrarily-sized image classification [31], [32] which consists in the addition of a learnable Relative Positional Bias (RPB) matrix $\mathbf{B} \in \mathbb{R}^{L \times L}$ where L is the temporal length of the feature map:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{SoftMax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{d}} + \mathbf{B}\right)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$ are query, key and value matrices. Each entry $\mathbf{B}_{i,j}$ is learned with a Multi-Layer Perceptron (MLP) on the relative difference between positions i and j:

$$\mathbf{B}_{i,j} = \text{MLP}(i - j)$$

3.3. Style Grounding

To maximize its utility as a creative tool for music artists, our objective is a generation system that is controllable by the user. To this end, we design a technique that enables the generation of single-stem samples with user-specified timbre characteristics and style. Given a reference audio waveform $\mathbf{y}$ provided by the user to indicate their desired style, we first encode it to a compressed latent representation $\mathbf{c}_{style}$ with the corresponding audio autoencoder. Then, we simply average the latent representation over the timesteps to obtain a single dim_y dimensional vector $\mu_t(\mathbf{c}_{style})$, where $\mu_t(\cdot)$ indicates the average across all timesteps. Finally, during the diffusion model sampling process, we force the generated latent samples at each reverse diffusion timestep to have an average across time that remains close to $\mu_t(\mathbf{c}_{style})$. We weigh this re-centering by the square of the timestep-specific noise rate, so that the effect is stronger at earlier iterations while keeping the model free to deviate when generating the lower-level details of the sample. Given the denoised output of the diffusion model $\hat{\mathbf{c}}_{y,k} \in \mathbb{R}^{T \times dim_y}$ at sampling iteration k we calculate:

$$\hat{\mathbf{c}}_{y,k,ground} = \hat{\mathbf{c}}_{y,k} - \mu_t(\hat{\mathbf{c}}_{y,k}) + \beta_k^2 \mu_t(\mathbf{c}_{style}) + (1 - \beta_k^2)\mu_t(\hat{\mathbf{c}}_{y,k})$$

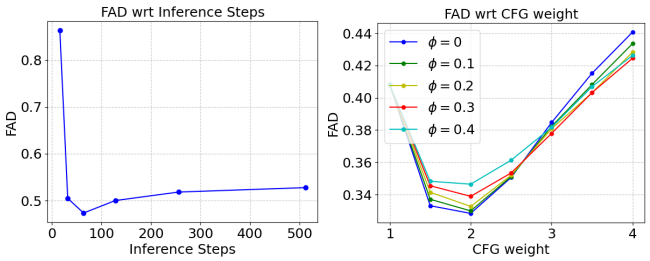

Fig. 2: *Left*: FAD evaluation of unconditional samples with respect to the number of DDIM inference steps. 64 steps result in the lowest FAD, and we use $K = 64$ in all subsequent experiments. *Right*: FAD evaluation of conditional samples with respect to CFG weights and with varying ϕ. When higher CFG weights (> 2.5) are used, the latent rescaling technique results in lower FAD.

This technique exploits the semantically rich latent space produced by the autoencoder to enforce distinct timbre features captured in $\bar{c}_y$ onto the output of the diffusion model.

3.4. Classifier-Free Guidance

Classifier-Free Guidance (CFG) [33] is a technique that allows a conditional diffusion model to generate samples that more closely adhere to the provided input:

$$\hat{\mathbf{c}}_{k,cfg} = G_\theta(\hat{\mathbf{z}}_k, k, \mathbf{c_x}) + \lambda_{cfg}(G_\theta(\hat{\mathbf{z}}_k, k) - G_\theta(\hat{\mathbf{z}}_k, k, \mathbf{c_x}))$$

where $G_\theta(\hat{\mathbf{z}}_k, k)$ is an unconditionally-generated sample at timestep k. However, when high guidance weights λ_{cfg} are used, image generation models are known to generate overly saturated and exposed images [34]. We experience a similar issue in our latent audio generation scenario, with highly distorted and saturated samples being generated. Solutions such as clipping of the guided samples between a defined range of values or dynamic thresholding [34] are not applicable in our case, since our latent space is not bounded. We thus use the technique proposed by [35] for guiding the generation of arbitrary spaces, which controls the increase in standard deviation of the guided samples with an hyperparameter $\phi \in [0, 1]$, and allows us to reduce artifacts at higher guidance weights.

	Grounded	Not Grounded
Cosine Distance	0.269	0.644
Euclidean Distance	0.407	0.836

Table 1: Average Euclidean and Cosine distance between embeddings of style samples from the test set and embeddings of generated samples both using the proposed grounding technique and not using it.

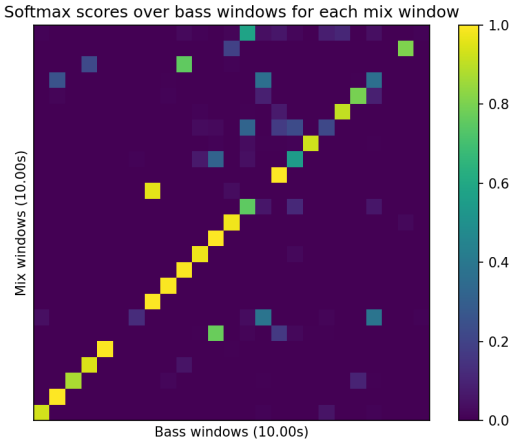

Fig. 3: Soft assignments of 25 random input mixes and corresponding generated basslines by a contrastive model (Section 5). High diagonal values indicate the generated basslines best match their respective conditional inputs.

4. IMPLEMENTATION DETAILS

We train the audio autoencoders on random crops of 1.5 seconds to produce representations with $dim_x = 64$ and $dim_y = 32$, while $r_{time} = 4096$ is kept the same for both models. Input log-magnitude spectrograms for both the autoencoder and the critics are calculated using $hop\_len = 256$ and $win\_len = 4 \cdot hop\_len$. 128 mel-bins are used for the second critic. The architecture of both autoencoder and critics consists of residual convolutional blocks. We choose $\lambda_{rec} = 25, \lambda_{mssd} = 0.002$, and the multi-scale spectral distance loss is calculated using $hop\_len \in [2^5, 2^6, 2^7, 2^8, 2^9, 2^{11}, 2^{12}]$. We always choose $win\_len = 4 \cdot hop\_len$. The autoencoders consist of 37M parameters and are trained using Adam [36] with $\beta_1 = 0.5$ and $\beta_2 = 0.9$ for 500k iterations at a batch size of 32. The latent diffusion model is trained on (mix, stem) pairs, where both samples are $\sim$23 seconds long and are first encoded to 256 timesteps-long latent representations. For a given track, the mix is obtained by mixing a non-empty random subset of stems from the track. The latent diffusion model consists of residual convolutional blocks, with self-attention layers at the lower resolution levels. The latent representation of the conditioning mix is concatenated with the noisy input, while the diffusion timestep information is expressed through sinusoidal embeddings [29] which are concatenated with the feature maps before every block. 15% of input latent representations are zero-ed out to train the model unconditionally, thus allowing CFG. The latent diffusion model consists of 42M parameters and is trained using AdamW [37] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for 500k iterations at a batch size of 128. To train the model we use the v-objective [38] with a cosine schedule, while at inference we use the DDIM sampler [9].

5. EXPERIMENTS AND RESULTS

We train the proposed accompaniment generation system on the task of conditional bassline generation, using an internal dataset of 20 k songs with available stems, among which the bass guitar. 1,500 of the tracks are used as test set. We first train the audio autoencoder used to encode the input mixes on the MTG-Jamendo dataset [39]. The autoencoder used to encode the bass samples is trained on bass stems from our internal dataset and the latent diffusion model is trained on (mix, bass stem) pairs from the same dataset. We first evaluate the quality of unconditionally generated samples with respect to the number of DDIM steps in Fig. 2 (right). We show in Fig. 2 (left) how the CFG rescaling technique can improve the FAD of generated samples for high CFG weights. To evaluate the ability of the system to generate samples that musically match the input mix, we train a contrastive model to assign high scores to matching (mix, bass stem) pairs and low scores to non-matching ones using the same internal dataset. In Fig. 3, we visualize the scores assigned by that model to 25 pairs of random segments of mixes from the test set, and 25 bass stems generated conditionally for each of those segments. A high value on the diagonal means the bass stem generated for that mix matches that mix better than the bass stems generated for the other mixes. To quantitatively evaluate the efficacy of the proposed style grounding technique, we use an off-the-shelf audio classification model [40] to extract embeddings of generated samples with and witout style-grounding (using the same input mix as conditioning), and compare them in Table 1 to embeddings of the target style sample via the Cosine and Euclidean distance. Readers can listen to samples generated by our system at: `https://sonycslparis.github.io/bass_accompaniment_demo/`

6. CONCLUSION

We have presented a novel controllable system for music accompaniment generation using latent diffusion models. When trained on bass stems, our model is able to generate basslines that musically match an arbitrary input mix. We propose the design of an efficient audio autoencoder for producing compressed invertible latent representations, the adaptation of latent diffusion models to handle inputs and outputs of arbitrary length, and a latent-specific style grounding technique to control the timbre of generated samples. Experiments demonstrate that our model can generate basslines that musically match the input mix and that can be grounded with user-provided timbres. A limitation of our system is that it does not offer user control over the exact notes of the generated accompaniment. Future work involves training the model to generate other instruments besides bass. We believe our system can enhance the creative workflow of music artists, creating a variety of bass accompaniments to fit their existing material, while also offering control over the creation process.

This work was supported by UKRI [grant EP/S022694/1].

References

[1] G. Hadjeres *et al.*, "Deepbach: A steerable model for bach chorales generation," in *ICML*, 2017.

[2] C. A. Huang *et al.*, "Music transformer: Generating music with long-term structure," in *ICLR*, 2019.

[3] D. von Rütte *et al.*, "Figaro: Generating symbolic music with fine-grained artistic control," *arXiv preprint arXiv:2201.10936*, 2022.

[4] P. Dhariwal *et al.*, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[5] A. Agostinelli *et al.*, *Musiclm: Generating music from text*, 2023. arXiv: 2301.11325 [cs.SD].

[6] G. Mariani *et al.*, *Multi-source diffusion models for simultaneous music generation and separation*, 2023. arXiv: 2302.02257 [cs.SD].

[7] J. Sohl-Dickstein *et al.*, "Deep unsupervised learning using nonequilibrium thermodynamics," in *ICML*, 2015.

[8] J. Ho *et al.*, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020.

[9] J. Song *et al.*, "Denoising diffusion implicit models," in *ICLR*, 2021.

[10] R. Rombach *et al.*, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[12] A. van den Oord *et al.*, "WaveNet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop*, 2016.

[13] S. Mehri *et al.*, "SampleRNN: An unconditional end-to-end neural audio generation model," in *ICLR*, 2017.

[14] J. Copet *et al.*, "Simple and controllable music generation," *arXiv preprint arXiv:2306.05284*, 2023.

[15] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *NeurIPS*, 2014.

[16] C. Donahue *et al.*, "Adversarial audio synthesis," in *ICLR*, 2019.

[17] J. H. Engel *et al.*, "GANSynth: Adversarial neural audio synthesis," in *ICLR*, 2019.

[18] M. Pasini and J. Schlüter, "Musika! fast infinite waveform music generation," in *ISMIR*, 2022.

[19] M. Grachten *et al.*, "Bassnet: A variational gated autoencoder for conditional generation of bass guitar tracks with learned interactive control," *Applied Sciences*, 2020.

[20] Z. Kong *et al.*, "Diffwave: A versatile diffusion model for audio synthesis," in *ICLR*, 2021.

[21] N. Chen *et al.*, "Wavegrad: Estimating gradients for waveform generation," in *ICLR*, 2021.

[22] S. Forsgren and H. Martiros, *Riffusion - stable diffusion for real-time music generation*, 2022. [Online]. Available: https://riffusion.com/about.

[23] F. Schneider *et al.*, "Mo\^ usai: Text-to-music generation with long-context latent diffusion," *arXiv preprint arXiv:2301.11757*, 2023.

[24] P. Li *et al.*, "Jen-1: Text-guided universal music generation with omnidirectional diffusion models," *arXiv preprint arXiv:2308.04729*, 2023.

[25] J. H. Engel *et al.*, "DDSP: differentiable digital signal processing," in *ICLR*, 2020.

[26] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[27] Y. Song *et al.*, "Maximum likelihood training of score-based diffusion models," in *NeurIPS*, 2021.

[28] O. Ronneberger *et al.*, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[29] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[30] Y. Sun *et al.*, "A length-extrapolatable transformer," in *ACL*, 2023.

[31] W. Wang *et al.*, "Crossformer: A versatile vision transformer hinging on cross-scale attention," in *ICLR*, 2022.

[32] Z. Liu *et al.*, "Swin transformer V2: scaling up capacity and resolution," in *CVPR*, 2022.

[33] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[34] C. Saharia *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *arXiv preprint arXiv:2205.11487*, 2022.

[35] S. Lin *et al.*, "Common diffusion noise schedules and sample steps are flawed," *arXiv:2305.08891*, 2023.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[37] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[38] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *ICLR*, 2022.

[39] D. Bogdanov *et al.*, "The mtg-jamendo dataset for automatic music tagging," in *Machine Learning for Music Discovery Workshop, ICML*, 2019.

[40] Q. Kong *et al.*, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *ACM Trans. Audio Speech Lang. Process.*, 2020.