CRASH: RAW AUDIO SCORE-BASED GENERATIVE MODELING FOR CONTROLLABLE HIGH-RESOLUTION DRUM SOUND SYNTHESIS

Simon Rouard
Sony CSL - CentraleSupélec
simon.rouard@student-cs.fr

Gaëtan Hadjeres
Sony CSL
gaetan.hadjeres@sony.com

ABSTRACT

In this paper, we propose a novel score-base generative model for unconditional raw audio synthesis. Our proposal builds upon the latest developments on diffusion process modeling with stochastic differential equations, which already demonstrated promising results on image generation. We motivate novel heuristics for the choice of the diffusion processes better suited for audio generation, and consider the use of a conditional U-Net to approximate the score function. While previous approaches on diffusion models on audio were mainly designed as speech vocoders in medium resolution, our method termed CRASH (Controllable Raw Audio Synthesis with High-resolution) allows us to generate short percussive sounds in 44.1kHz in a controllable way. Through extensive experiments, we showcase on a drum sound generation task the numerous sampling schemes offered by our method (unconditional generation, deterministic generation, inpainting, interpolation, variations, class-conditional sampling) and propose the *class-mixing* sampling, a novel way to generate "hybrid" sounds. Our proposed method offers flexible generation capabilities with lighter and easier-to-train models than GAN-based methods.

1. INTRODUCTION AND RELATED WORK

After multiple works in the spectral domain [1, 2], deep generative models in the waveform domain have recently shown the ability to produce high fidelity results with different methods: autoregressive [3, 4], flow-based [5], energy-based [6] or based on Generative Adversarial Networks [7].

In the task of generating drum sounds in the waveform domain, GAN-based approaches have been explored in [7], [8] and [9]. Interactive sound design is often a major motivation behind these works: in [10] the authors use Variational Autoencoders (VAE) in order to generate spectrograms of drums apply a principal component analysis on the latent space of the VAE in order to explore the drum timbre space. One of the disadvantages of this model is that the reconstruction of the sounds by the VAE tends to be blurry. In [11], the authors use a VQ-VAE2 [12] in order to perform inpainting on instrument sound spectrograms.

Score-based generative models [13–16] propose a different approach to generative modeling, which consists in estimating the gradient of noise-corrupted data log-densities (score function): by iteratively denoising a sampled noise, these approaches obtained promising results, but mainly on image data.

To this day, only two score-based generative models in the waveform domain have been published [17, 18] and they are mostly focused on the task of neural vocoding with conditioning on a mel-spectrogram. In [17], the authors achieved the task of generating audio with an unconditioned model trained on the speech command dataset [19]. The inference scheme of [17] does not provide a flexible sampling scheme because it is trained on a fixed discrete noise schedule whereas [18] is trained on a continuous scalar indicating the noise level.

In the image domain, [16] generalizes the works of [14, 15, 20] by framing the noise corruption procedure as stochastic differential equation.

Score-based generative models offer the following advantages over GAN-based approaches:

- Training time is reduced and training is more stable since there is only one network to train.
- Class-conditioning generation can be achieved by training a classifier a posteriori, which lets us train a model only one time.
- Data can be mapped to a latent space without the need to train an additional encoder compared to GANs [21], which makes the interpolation between two given input data readily available with only one model.

These properties alleviate us to search for directions in the latent space as in [22] or to directly hardcode conditional features in the architecture as in [23]. This easily controllable latent space permits sound design applications. One downside of score-based models compared to GANs is their higher inference times to generate new samples.

In this work, we extend the approach of [16] and propose CRASH (Controllable Raw Audio Synthesis with High-resolution), a score-based generative model adapted to the waveform domain. On a drum sound dataset, the numerous capabilities offered by this architecture allows for musically-relevant sound design applications. Our contributions are the following:

- A score-based model for unconditional generation that can achieve high fidelity 44.1 kHz drum sounds directly in the waveform domain,
- The use of a noise-conditioned U-Net to estimate the score function,
- A novel *class-mixing* sampling scheme to generate "hybrid" sounds.
- Experimental and practical insights about the choice of the stochastic differential equation used to corrupt the data.

2. BACKGROUND

2.1 Score Based Modelling through Stochastic Differential Equations

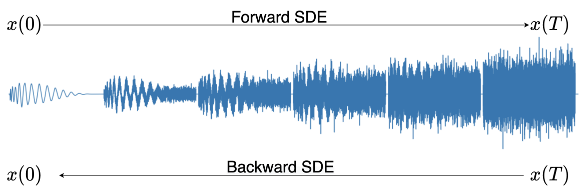


Figure 1. Illustration of the noising and denoising processes of a kick sound with a VP schedule

2.1.1 *Forward Process*

Let p_{data} be a data distribution. Diffusion models consist in progressively adding noise to the data distribution to transform it into a known distribution from which we can sample from as shown in Fig. 1. In [16], the authors formalize this noising process as the following **forward** Stochastic Differential Equation (SDE):

$$d\mathbf{x} = f(t)\mathbf{x}dt + g(t)d\mathbf{w} \tag{1}$$

where $f(t)$ is a continuous negative function from $[0, T] \to \mathbb{R}^-$, $g(t)$ a continuous positive function from $[0, T] \to \mathbb{R}^+$, and $\mathbf{w}$ is a standard Wiener process. Such approach can be understood as a continuous-time generalization of Denoising Diffusion Probabilistic Models (DDPMs) [14, 20] and denoising Score Matching with Langevin Dynamics (SMLD) [15]. For $\mathbf{x}(0) \sim p_{\text{data}}$, the transition kernel of Eq. 1 is given by a normal distribution:

$$p_t(\mathbf{x}(t) \mid \mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); m(t)\mathbf{x}(0), \sigma^2(t)\mathbf{I}), \tag{2}$$

where $m(t)$ and $\sigma(t)$ follow the system:

$$\begin{cases} \frac{dm}{dt} = f(t)m(t) \\ \frac{d\sigma^2(t)}{dt} = 2f(t)\sigma^2(t) + g^2(t) \end{cases} \tag{3}$$

with the following initial conditions $m(0) = 1$ and $\sigma^2(0) = 0$. The solutions for $m(t)$ and $\sigma(t)$ are :

$$\begin{cases} m(t) = e^{\int_0^t f(s)ds} \\ \sigma^2(t) = e^{\int_0^t 2f(s)ds} \int_0^t g^2(u) e^{\int_0^u -2f(s)ds} du. \end{cases} \tag{4}$$

In [16], the authors define three types of SDEs which are presented in Tab. 1. For the Variance Preserving (VP)

	$f(t)$	$g(t)$
VP	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}$
VE	0	$\sqrt{\frac{d[\sigma^2(t)]}{dt}}$
sub-VP	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)(1-e^{-2\int_0^t \beta(s)ds})}$

Table 1. Functions used in the VP, VE and sub-VP SDEs

and sub-Variance Preserving (sub-VP) schedules, $m(T) \approx 0$ and $\sigma(T) \approx 1$ which means that the original data distribution is transformed into a distribution close to a standard normal distribution i.e. $p_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. For the Variance Exploding (VE), $\sigma^2(T) \gg m \approx 1$ which means that the original data is not discernable at $t = T$ and that $p_T \approx \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$.

2.1.2 *Generation with the Reverse Process*

In order to sample from the data distribution, we can sample $\mathbf{x}_T \sim p_T$ and apply the associated reverse time SDE [24] given by:

$$d\mathbf{x} = [f(t)\mathbf{x} - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\tilde{\mathbf{w}} \tag{5}$$

where $\tilde{\mathbf{w}}$ is a standard Wiener process running backwards from T to 0 and dt is an infinitesimal negative timestep.

It means that by knowing $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can use a discretization of Eq. 5 to sample $\mathbf{x}(0)$ from $p_0 = p_{\text{data}}$.

In practice, the score function $s(\mathbf{x}(t), \sigma(t)) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is intractable and it is approximated by a neural network $s_\theta(\mathbf{x}(t), \sigma(t))$ parameterized by θ. In order to train the network, [13] shows that for any t, minimizing

$$\mathbb{E}_{p_t(\mathbf{x})} \|s_\theta(\mathbf{x}, \sigma(t)) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2 \tag{6}$$

is equivalent to minimizing

$$\mathbb{E}\|s_\theta(\mathbf{x}, \sigma(t)) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2 \tag{7}$$

where the expectation is over $\mathbf{x}(0) \sim p_{\text{data}}$, $\mathbf{x}(t) \sim p_t(\mathbf{x}(t) \mid \mathbf{x}(0))$, and the latter distribution is given by Eq. 2.

Now, in order to train the network for all $t \in [0, T]$ we consider the following mixture of Eq. 7 losses over all noise levels:

$$L(\theta) = \mathbb{E}\lambda(t)\|s_\theta(\mathbf{x}(t), \sigma(t)) - \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2 \tag{8}$$

where we sample $t \sim [0, T]$, $\mathbf{x}(0) \sim p_{\text{data}}$, $\mathbf{x}(t) \sim p_t(\mathbf{x}(t) \mid \mathbf{x}(0))$ and where $\lambda(t)$ is a weighting function.

In [14–16], $\lambda(t)$ is empirically set such that $\lambda(t)^{-1} \propto \mathbb{E}\left\|\nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t) \mid \mathbf{x}(0))\right\|_2^2 \propto \sigma^2(t)^{-1}$ while in [25] the authors show that the maximum likelihood estimator is obtained with $\lambda(t) = g^2(t)$ in $L(\theta)$.

The training procedure is described in Alg. 1, where we reparameterize our neural network as $\epsilon_\theta(\mathbf{x}(t), \sigma(t)) := -\sigma(t)s_\theta(\mathbf{x}(t), \sigma(t))$ in order to estimate ϵ.

Algorithm 1 Training procedure

```
while Training do
    Sample $t \sim \mathcal{U}([0,T]), \mathbf{x}(0) \sim p_{\text{data}}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Compute $\mathbf{x}(t) = m(t)\mathbf{x}(0) + \sigma(t)\epsilon$
    Gradient descent on $\nabla_\theta \left\| \frac{\sqrt{\lambda(t)}}{\sigma(t)} [\epsilon_\theta(\mathbf{x}(t), \sigma(t)) - \epsilon] \right\|_2^2$
end while
```

Once the network is trained, a N-step discretization of the **backward** SDE is done in order to unconditionally generate samples. This process is described in Alg. 2, it is non-deterministic since we obtain various sounds by starting from the same sample $\mathbf{x}(T)$.

Algorithm 2 Sampling via SDE

```
Choose $N$, sample $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$
for $i = N-1, ..., 0$ do
    $t_i = T\frac{i}{N}, f_i = f(t_i), g_i = g(t_i), \sigma_i = \sigma(t_i)$
    $\mathbf{x}_i = (1 - \frac{f_{i+1}}{N})\mathbf{x}_{i+1} - \frac{g_{i+1}^2}{N\sigma_{i+1}}\epsilon_\theta(\mathbf{x}_{i+1}, \sigma_{i+1})$
    if $i > 0$ then
        Sample $\mathbf{z}_{i+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
        $\mathbf{x}_i = \mathbf{x}_i + \frac{g_{i+1}}{\sqrt{N}}\mathbf{z}_{i+1}$
    end if
end for
```

2.2 Deterministic Sampling via Score based Ordinary Differential Equation

As mentioned in [16], for any SDE, there exists a corresponding deterministic process which satisfies an ordinary differential equation (ODE):

$$d\mathbf{x} = [f(t)\mathbf{x} - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt \tag{9}$$

This defines a flow ϕ^t such that the marginal distributions $\phi_*^t(p_{\text{data}})$ are identical to the ones obtained by applying the SDE of Eq. 1. This mapping is interesting because it provides a latent representation for each $\mathbf{x} \sim p_{\text{data}}$.

The procedure of sampling via the N-step discretization of the ODE is described in Alg. 3.

Algorithm 3 Sampling via ODE

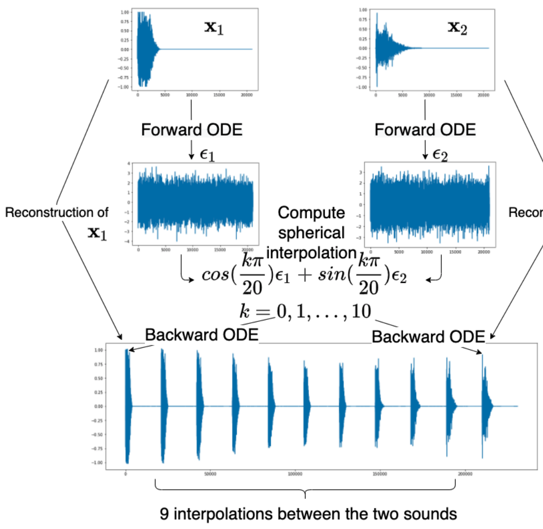
```
Choose $N$, sample $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$
for $i = N-1, ..., 0$ do
    $t_i = T\frac{i}{N}, f_i = f(t_i), g_i = g(t_i), \sigma_i = \sigma(t_i)$
    $\mathbf{x}_i = (1 - \frac{f_{i+1}}{N})\mathbf{x}_{i+1} - \frac{g_{i+1}^2}{2N\sigma_{i+1}}\epsilon_\theta(\mathbf{x}_{i+1}, \sigma_{i+1})$
end for
```

2.3 Inpainting

Let's imagine that we don't like the attack of a kick (or any other part of a sound), the method of inpainting permits us to regenerate the desired part. In order to do that, we apply a reverse-time SDE or ODE discretization to an isotropic Gaussian and fix the part that we want to keep (with the associated noise corruption) after each denoising timestep. As presented in section 6, we obtain very diverse and coherent results.

2.4 Interpolations

The flexibility of SDEs and ODEs allows to compute interpolations between sounds. In fact, there exists an infinity of latent spaces indexed by $t \in [0, T]$. We present here two types of interpolations: ODE interpolation in the latent space of isotropic Gaussians and SDE interpolation in any t-indexed latent space.



$\mathbf{x}_1$
$\mathbf{x}_2$
Forward ODE
ϵ_1
Forward ODE
ϵ_2
Reconstruction of $\mathbf{x}_1$
Compute spherical interpolation
Reconstruction of $\mathbf{x}_2$
$cos(\frac{k\pi}{20})\epsilon_1 + sin(\frac{k\pi}{20})\epsilon_2$
$k = 0, 1, \ldots, 10$
Backward ODE
Backward ODE
9 interpolations between the two sounds

Figure 2. Interpolation of two sounds via Forward and Backward ODE

2.4.1 *ODE interpolation in the latent space of isotropic Gaussians*

Let ϵ_1 and ϵ_2 be two samples from a standard normal distribution of $\mathbb{R}^L$ where L is our space dimension and $0 \leq \lambda \leq 1$. We consider the spherical interpolation $\epsilon_\lambda = \lambda\epsilon_1 + \sqrt{1-\lambda^2}\epsilon_2$ and then apply the ODE sampling to it. We choose a spherical interpolation in order to preserve a variance close to 1 for ϵ_λ.

Morever, if we want to interpolate two sounds $\mathbf{x}_1$ and $\mathbf{x}_2$, we can apply the Forward ODE in order to obtain the corresponding latent codes ϵ_1 and ϵ_2, apply the desired spherical interpolation and then apply an ODE sampling.

2.4.2 *ODE interpolation in a t-indexed latent space*

In [17], the authors perform a linear interpolation between two sounds at a corresponding intermediate t-indexed latent space before applying Denoising Diffusion Probabilistic Model (DDPM is the discrete equivalent of a VP SDE). We adapt the method to the continuous framework with

SDE and ODE. Here again, the interpolation can be done between two t-indexed latent codes or between sounds corrupted using the transition kernel of Eq. 2.

2.5 Class-Conditional sampling with a classifier

For any class y, we can train a noise-conditioned classifier on corrupted data $\mathbf{x}(t)$. As a consequence, the output of the classifier gives us $p_t(y \mid \mathbf{x}(t))$ for each class y. We can use automatic-differentiation to differentiate this quantity and by the Bayes Formula, since $p(y)$ is constant for each class y, we have the following formula:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid y) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(y \mid \mathbf{x}) \quad (10)$$

As a consequence, we can generate samples of one class by solving this reverse time SDE:

$$d\mathbf{x} = [f(t)\mathbf{x} - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid y)]dt + g(t)d\tilde{\mathbf{w}} \quad (11)$$

This approach is flexible since it only requires to train a noise-conditioned classifier: there is no need to design and train a class-conditional score-based model as done in [17].

3. A DISCUSSION ABOUT CHOOSING THE RIGHT SDE: A GENERALIZATION OF THE SUB-VP SDE

In this section $T = 1$.

3.1 About the relation relation between $m(t)$ and $\sigma(t)$

The VP SDE is the continuous version of the Denoising Diffusion Probabilistic Model (DDPM) used in [14] [17] [18]. One of the main features of this model is that the mean coefficient $m(t)$ of the perturbation kernel is linked to the standard deviation $\sigma(t)$ (or noise-level) by the following equation $m(t) = \sqrt{1 - \sigma^2(t)}$. Because the decrease of m is relatively small on a large range of values for σ, this means that a (very-noisy) drum sound audio must begin to appear after only a few steps of denoising during the sampling algorithm. (For instance if $\sigma = 0.8$, $m = 0.6$). We believe that this fast decay of the signal-to-noise ratio can be detrimental when sampling with Alg. 2 and 3.

Moreover, without mentioning this fact, in [16] the authors introduce the sub-VP SDE which is characterized by the following formula $m(t) = \sqrt{1 - \sigma(t)}$. The authors obtained state of the art results on the image generation task which corroborates our intuition that m might be too large for values of σ near 1 tends to be right.

In this work, we explore the four relations between m and σ plotted in Fig. 3. The blue one corresponds to the VP SDE, the yellow is the sub-VP SDE and the green and red ones corresponds to a generalization of the sub-VP schedule. In Tab. 2 we write the functions $f(t)$ and $g(t)$ for each of these 4 relation. For the rest of the paper we take the convention $f(t) := -\frac{1}{2}\beta(t)$ in order to compare the VP and sub-VP schedules with ours.

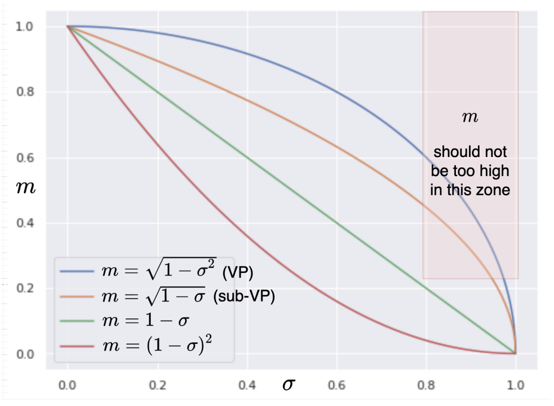


Figure 3. Different relations between m and σ

m-σ relation	$f(t)$	$g(t)$
$m = \sqrt{1-\sigma^2}$ (VP)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}$
$m = \sqrt{1-\sigma}$ (sub-VP)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)ds})}$
$m = 1-\sigma$ (sub-VP 1-1)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)(1 - e^{-\frac{1}{2}\int_0^t \beta(s)ds})}$
$m = (1-\sigma)^2$ (sub-VP 1-2)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)(1 - \frac{3}{2}e^{-\frac{1}{2}\int_0^t \beta(s)ds} + \frac{1}{2}e^{-\int_0^t \beta(s)ds})}$

Table 2. Functions used in the VP, sub-VP and generalized sub-VP SDEs.

3.2 Choosing the right functions for the SDE

Choosing a particular relation between m and σ, imposes a relation between g and β. The remaining free parameter is the function β, needed to fully define the SDE. In [16], the authors use a linear schedule for $\beta(t)$ because it is the continuous generalization of DDPMs. As presented in Fig. 4, this choice leads to a $\sigma(t)$ function that rapidly grows to its maximum. In [26], the authors mention this fast growing σ function as a potential shortcoming and propose a smoother function (the green one in Fig. 4).

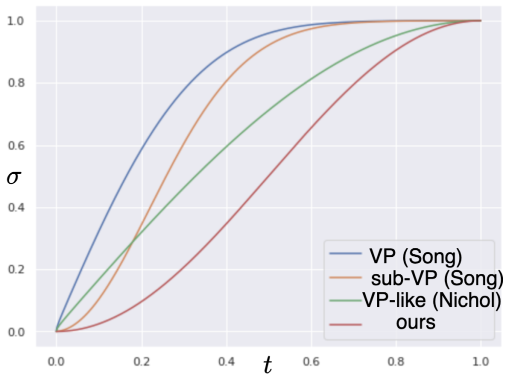


Figure 4. Different choices for the $\sigma(t)$ function

Our approach differs from [16] in that the definition of our SDE is motivated by choosing a relatively smooth increasing function $\sigma(t)$ such as $\sigma(0) = 0$ and $\sigma(1) = 1 - \epsilon$ (where ϵ is a small constant), together with a m-σ relation, from which all other quantities can be computed as shown in Tab. 2. If the two approaches are equivalent, we believe that these quantities are more interpretable. In the regime of a small number of discretization steps, a slow increasing function may induce less approximation errors. For our experiments we propose $\sigma(t) = \frac{1}{2}[1 - \cos((1-s)\pi t)]$ with $s = 0.006$ which is the red plot in Fig. 4. We also sample t in the interval $[\eta, 1]$ during the training where η is chosen such that $\sigma(\eta) = 10^{-4}$ because 10^{-4} is imperceptible.

4. CLASS-MIXING SAMPLING WITH A CLASSIFIER

Drum classes are not perfectly distinct. For instance, the dataset contains drum sounds that are percussive enough to be seen as kicks but also sufficiently brilliant to be seen as snares and some kicks are combined with a hi-hat sound. We observe that our classifier (at the noise-level $\sigma = 0$) sometimes outputs a mixed classes such as [0.3, 0.3, 0.4] and that it aligns well with our feeling when hearing the sound.

We introduce the Class-Conditional sampling to a mixture of classes: For a given noisy sound $\mathbf{x}(t)$, the vector $\nabla_{\mathbf{x}(t)} \log p_t(y_i \mid \mathbf{x}(t))$ points out to the direction of the class y_i in the noisy t-indexed latent space. Now, assuming that we have N classes $(y_i)_{i=1,...,N}$, let $(\lambda_i)_{i=1,...,N}$ be positive real numbers such as $\sum_{i=1}^{N} \lambda_i = 1$, we define a mixture of classes that we note $\{(y_i, \lambda_i)\}$ and the associated vector:

$$\nabla_{\mathbf{x}} \log p_t(\{(y_i, \lambda_i)\} \mid \mathbf{x}) := \sum_{i=1}^{N} \lambda_i \nabla_{\mathbf{x}} \log p_t(y_i \mid \mathbf{x}) \tag{12}$$

In practice, we put this term in equation 10 in replacement of the last term and use equation 11 to sample class-mixed audios. It gives us interesting results with a great palette of sounds.

5. ARCHITECTURE

5.1 Conditioned U-Net

Our model architecture is a conditioned U-Net [27], originally proposed for source separation. It takes two inputs: the noise level $\sigma(t)$ and the noisy audio $\mathbf{x}(t)$. The noise-level is encoded by Random Fourier Features [28] followed by a Multi-Layer Perceptron. The noisy audio goes into FiLM-conditioned [29] Downsampling Blocks. Then, the signal goes into Upsampling Blocks that receive skip connections from the DBlocks of same levels. The output of the network is the estimated noise $\epsilon_{\text{estimated}}$.

This bears similarities with the architecture from [17] which has a similar succession of blocks with dilated convolutions but no downsampling or upsampling layers, which makes it slow in terms of computation. The architecture from [18] has a U-Net-like shape [30], but heavily depends on the spectrogram conditioning and relies on a different noise-conditioning scheme. The σ-conditioned U-Net architecture seems to retain advantages from both approaches and is particularly suited for unconditional generation (see Fig. 5).

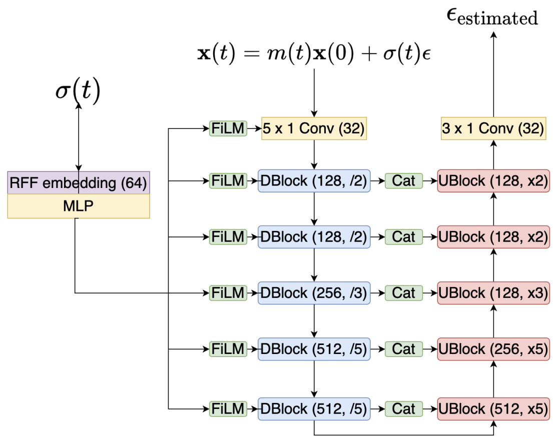


Figure 5. Architecture of the Conditioned U-Net

5.2 Noise conditioned classifier

Our noise-conditioned classifier closely mimics the architecture of our Conditioned U-Net presented in in Sect. 5.1. The classifier is composed of a succession of FiLM-conditioned DBlocks followed by a projection layer and a softmax.

6. EXPERIMENTS AND RESULTS

Code is available at: `https://github.com/simonrouard/CRASH`

6.1 Dataset

For this work, we use an internal non-publicly available dataset of drum sounds which has also been used in [8]. It is composed of approximately 300.000 one-shot kick, snare and cymbal sounds in equal proportions. The samples have a sample rate of 44.1kHz and are recorded in mono. We restricted and padded the audio to 21.000 time-steps because most sounds last less than 0.5 second. We used 90% of the dataset in order to train our model.

6.2 Models and process

We evaluate the influence of $\sigma(t)$ and four m-σ schedules. The training of the network is done with a learning rate of 2.10^{-4} and the Adam optimizer. In parallel, smoothed weights with exponential moving average (EMA) with a rate of 0.999 are computed and saved at each step. For each model, the network is trained for about 120 epochs and the weights are saved each 8 epochs. We generated drum sounds with the regular weights and with the EMA weights and we observed the same phenomenon as in [31]: for the regular weights the quality of the sounds is not necessarily increasing with the training time whereas the EMA weights provide better and more homogeneous Fréchet Audio Distance [32] (FAD) during training [1].

[1] We use the original implementation [32] available at `https://github.com/google-research/google-research/tree/master/frechet_audio_distance`

After generating 2700 sounds for each checkpoint of each model, we choose the best checkpoints and generate 27000 drum sounds for each. It takes 12 hours to generate 27000 drum sounds on a Nvidia RTX-3090 GPU with an ODE or SDE schedule of 400 steps and batches of 180 sounds per generation (maximum memory capacity).

6.3 Quantitative Results

We report the FAD (lower is better) between the 27000 generated drum sounds and the test set for each unconditional generation with SDE and ODE (with a discretization of 400 steps) in the table 6.3. The cos schedule refers to the function $\sigma(t) = \frac{1}{2}[1 - \cos((1-s)\pi t)]$ (the red one in Fig. 4) and the exp schedule corresponds to the function $\sigma(t) = \sqrt{1 - e^{-0.1t - 9.95t^2}}$ used in [16] (the blue one in Fig. 4).

Schedule	SDE 400 steps	ODE 400 steps
VP exp schedule (as in [16])	4.30	4.03
VP cos schedule	2.32	**1.79**
sub-VP cos schedule	2.46	2.07
sub-VP 1-1 cos schedule	2.62	**1.73**
sub-VP 1-2 cos schedule	2.56	**1.75**

Table 3. FAD comparison (lower is better)

Choosing a smoother σ function indeed improves the FAD of the generated sounds for a fixed number of discretization steps.

By using the classifier that we trained, we observe that all models generate kicks, snares and cymbals in equal proportions but the generated samples are less diverse than in the original dataset. For a fixed number of discretization steps, we think that the cos schedule performs better because it is smoother than the exp schedule.

6.4 Interactive sound design

Audio samples for all experiments described in this section can be heard on the accompanying website: `https://crash-diffusion.github.io/crash/`.

6.4.1 Interpolations

The relative lack of diversity of the unconditional generation is not dramatic since the model can still perform interactive sound design by modifying existing samples from the dataset. In order to do that, we apply the forward ODE to an existing sound and obtain its corresponding noise in the latent space of isotropic Gaussians. As presented in Fig. 7, we can perform spherical combinations on the latent codes and apply the backward ODE to obtain interpolations. Moreover the reconstructed sounds (at the left and right of the schema) are accurate.

6.4.2 Obtaining Variations of a Sound by Noising it and Denoising it via SDE

Let's take a sound $\mathbf{x}(0)$. We can noise it at a desired noise level $\sigma(t)$ via $\mathbf{x}(t) = m(t)\mathbf{x}(0) + \sigma(t)\epsilon$ and then denoise it with a SDE discretization from t to 0. We obtain then variations of the original sound.

6.4.3 Inpainting

We can also perform inpainting on a sound in order to regenerate any desired part. We show this method on Fig. 6 where we regenerate 6 endings of a snare sound.

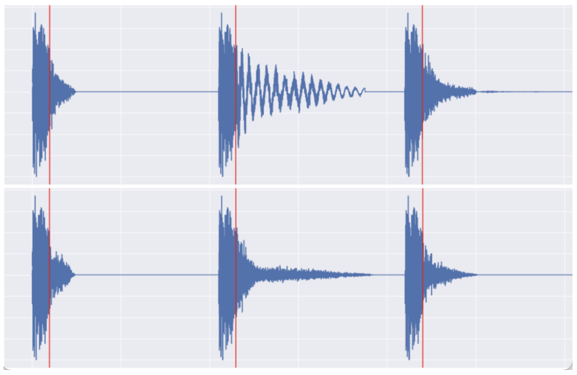


Figure 6. Six Inpaintings on the end of a snare sound

This provides an innovative way to generate a variety of plausible sounds starting with the same attack.

6.4.4 Class-Conditioning and Class-Mixing with a Classifier

We trained a noise-conditioned classifier on the 3 classes (kick, snare, cymbal) and used it to generate class-conditioned and class-mixing generation. Once again, by using the latent representation of a sound we can regenerate it (via ODE) with control on its "kickiness, snariness or cymbaliness".

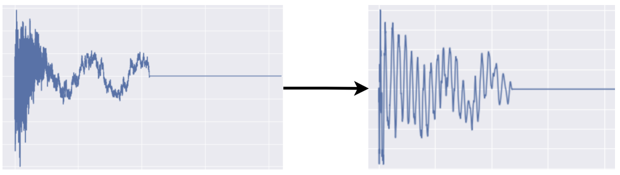


Figure 7. Transformation of a cymbal into a kick via class-conditioning ODE

7. CONCLUSION

We presented CRASH, a score-based generative model for the generation of raw audio based on the latest developments in modeling diffusion processes via SDEs. We proposed novel SDEs, well-suited to drum sound generation with high-resolution, together with an efficient architecture for estimating the score function. We showcased how the many controllable sampling schemes offered new perspectives for interactive sound design. In particular, our proposed *class-mixing* strategy allows the controllable creation of convincing "hybrid" sounds that would be hard to obtain with conventional means. We hope that these new methods will contribute to enrich the workflow of music producers.

8. REFERENCES

[1] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," 2019.

[2] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," 2019.

[3] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," 2016.

[4] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," 2017.

[5] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," 2018.

[6] A. A. Gritsenko, T. Salimans, R. van den Berg, J. Snoek, and N. Kalchbrenner, "A spectral energy distance for parallel speech synthesis," 2020.

[7] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," 2019.

[8] J. Nistal, S. Lattner, and G. Richard, "Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," 2020.

[9] J. Drysdale, M. Tomczak, and J. Hockman, "Adversarial synthesis of drum sounds," 2020.

[10] C. Aouameur, P. Esling, and G. Hadjeres, "Neural drum machine: An interactive system for real-time synthesis of drum sounds," in *International Conference on Computational Creativity*, 2019.

[11] T. Bazin, G. Hadjeres, P. Esling, and M. Malt, "Spectrogram inpainting for interactive generation of instrument sounds," *arXiv preprint arXiv:2104.07519*, 2021.

[12] A. Razavi, A. v. d. Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *NeurIPS*, 2019.

[13] P. Vincent, "A connection between score matching and denoising autoencoders," pp. 1661–1674, 2011.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[15] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 895–11 907.

[16] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2021.

[17] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," 2021.

[18] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," 2020.

[19] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[20] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," 2015.

[21] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, "Designing an encoder for stylegan image manipulation," *CoRR*, vol. abs/2102.02766, 2021. [Online]. Available: https://arxiv.org/abs/2102.02766

[22] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, "Ganspace: Discovering interpretable gan controls," 2020.

[23] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[24] B. D. O. Anderson, "Reverse-time diffusion equation models," pp. 313–326, 1982.

[25] C. Durkan and Y. Song, "On maximum likelihood training of score-based generative models," 2021.

[26] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," 2021.

[27] G. Meseguer-Brocal and G. Peeters, "Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations," *ArXiv*, vol. abs/1907.01277, 2019.

[28] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," 2020.

[29] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," 2017.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[31] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," 2020.

[32] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A metric for evaluating music enhancement algorithms," 2019.