DDSP-SFX: ACOUSTICALLY-GUIDED SOUND EFFECTS GENERATION WITH DIFFERENTIABLE DIGITAL SIGNAL PROCESSING

Yunyi Liu [*]
Electrical and Information Engineering
University of Sydney
Sydney, Australia
yunyi.liu@sydney.edu.au

Craig Jin
Electrical and Information Engineering
University of Sydney
Sydney, Australia
craig.jin@sydney.edu.au

David Gunawan [‡]
Dolby Laboratories
Sydney, Australia
dguna@dolby.com

ABSTRACT

Controlling the variations of sound effects using neural audio synthesis models has been a challenging task. Differentiable digital signal processing (DDSP) provides a lightweight solution that achieves high-quality sound synthesis while enabling deterministic acoustic attribute control by incorporating pre-processed audio features and digital synthesizers. In this research, we introduce DDSP-SFX, a model based on the DDSP architecture capable of synthesizing high-quality sound effects while enabling users to control the timbre variations easily. We integrate a transient modelling algorithm in DDSP that achieves higher objective evaluation scores and subjective ratings over impulsive signals (footsteps, gunshots). We propose a novel method that achieves frame-level timbre variation control while also allowing deterministic attribute control. We further qualitatively show the timbre transfer performance using voice as the guiding sound.

1. INTRODUCTION

Sound effects refer to natural or synthetic sounds different from speech or music. They play an important part in digital media such as film and games, but typically require intensive labour in recording. In recent years, deep generative models have demonstrated their synthesis capability in the audio domain, especially for modelling speech [1] and music [2]. However, neural audio synthesis (NAS) methods typically model the audio waveforms or time-frequency representations directly and require a large amount of data and computation power. To model sound effects with varying acoustic characteristics, a large, even vast, set of descriptive labels [3, 4], may be required, which is often impractical.

Sound effects are diverse and usually difficult to describe using words. To this end, many synthesis algorithms focus on using vocalizations to guide sound generation because voice can be an intuitive control interface for end-users. The supervised phoneme-based control-synthesis approach, which converts human-produced phonemes directly to sound effects, has been widely explored [5, 6, 7, 8]. However, such methods typically require ground-truth labelling between the phonemes and the corresponding sound effects. The unsupervised approach, which entails extracting acoustic features from the guiding sounds and applying such features to the generated sound effects, has also proven to be very effective in musical sound modelling. For example, differentiable digital signal processing [9] is a popular NAS architecture introduced to take advantage of pre-built digital synthesizers for waveform generation. Neural networks are then used to estimate the parameters for the corresponding synthesizers conditioned on pre-processed acoustic features. Owing to the modular approach and conditioned features, DDSP is lightweight to train and use while offering controllable audio synthesis over pre-defined audio features such as pitch and loudness. Although Lundberg [10] has applied DDSP to model motor engine sounds, there have been few attempts at modelling other sound effects, especially impulsive sounds such as footsteps or gunshots, two commonly used sound effects in games. To properly model the transient information, Barahona & Collins [11] proposed an alternative technique to model the inharmonic sounds through a controllable filterbank synthesizer under a similar DDSP framework. However, due to the fixed number of filterbanks design, it remains questionable whether it is capable of synthesizing highly time-varying pitched sounds such as motors accelerating, which constantly changes its pitch and frequency distributions over time. Additionally, it remains a critical issue as to how to effectively control the subtle timbre variations of a particular sound effect driven by vocalizations, given the differing acoustic characteristics between voice and various desired sound effects.

In this research, we explore sound effects modelling driven by acoustic features of target sounds utilizing the DDSP architecture. We are particularly interested in controlling the subtle timbre characteristics of the target sounds using a guiding sound, such as varying the type of engines for a motor sound with fixed pitch and loudness. To this end, we propose a novel approach for frame-level timbre variation independent of other acoustic attributes. Our method only requires a trained decoder to transfer the pre-defined acoustic envelopes and timbre envelopes from one type to another type of sound, without relying on an additional encoder. We further introduce an architecture targeted at the synthesis of impulsive sound effects based on the DDSP framework.

2. PROPOSED METHOD

We base our model on the vanilla DDSP architecture [9], which uses a sinusoidal model and a subtractive noise model as synthesizers. To better adapt the DDSP architecture to synthesize high-quality sound effects, we integrate a separate transient modelling method in the synthesizer and introduce a technique to avoid harmonic artifacts. To be able to control the timbre variations effectively, we propose an encoder structure that allows for frame-level timbre control. Our complete model architecture is depicted in

[*] Work done while interning at Dolby Laboratories.
[‡] This work has been supported by Dolby Laboratories.

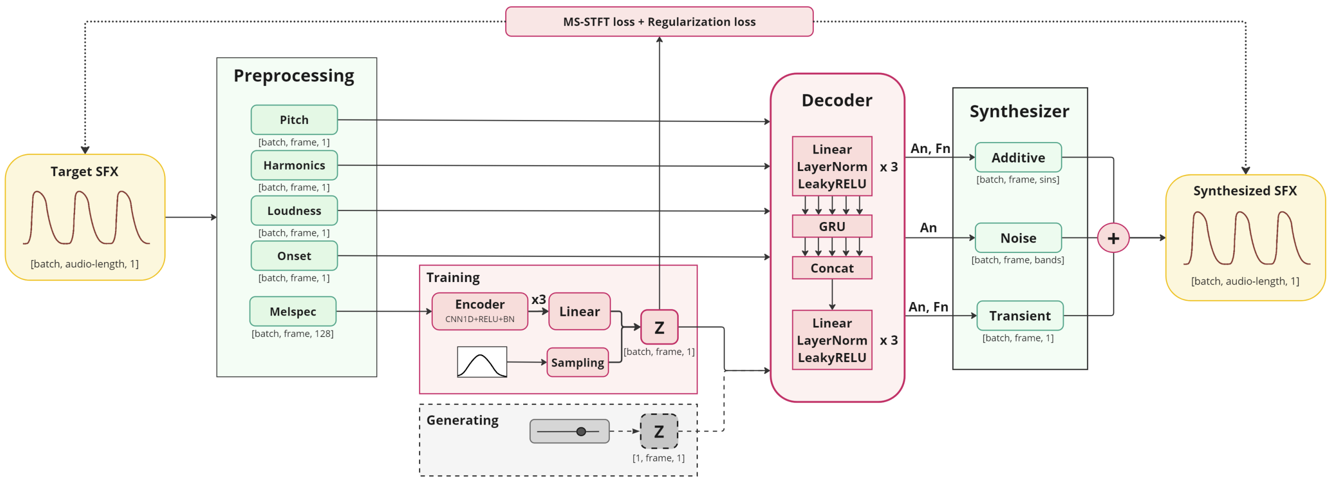


Figure 1: Proposed model architecture. Blocks in green are definitive features or algorithms. Blocks in pink are composed of learnable neural networks. Blocks in grey indicate they are used during the inference phase to replace the trained encoder.

Figure 1. Before the training stage, in addition to the pitch and loudness vectors described in DDSP [9], we also need to preprocess the audio to extract a few acoustic features. The acoustic features and the latent variable output from the encoder are concatenated together to be processed by the decoder, which is composed of several Multi-layer Perceptrons (MLPs) and Gated Recurrent Units (GRUs) as illustrated in the architecture. The decoder finally outputs the required synthesis parameters for the synthesizer, namely, frequencies and amplitudes of the sinusoids for the additive synthesizer, amplitude vector for the subtractive noise synthesizer, and finally, frequencies and amplitudes for the transient model. Lastly, all the synthesized outputs are combined together as the synthesized out. In the following sections, we elaborate on the proposed techniques.

2.1. Harmonic Indicator

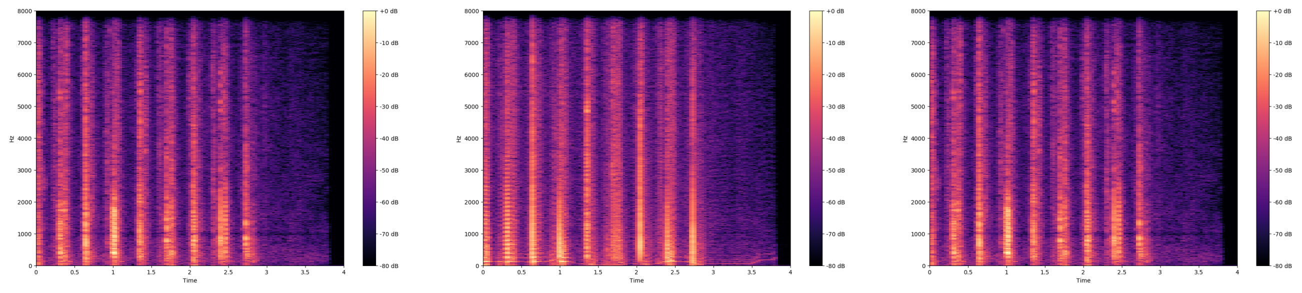
A harmonic plus noise model [12] performs well when the target sound is harmonic. However, when integrated into DDSP, it could lead to harmonic artifacts or distortion for sounds with little to no harmonic components. This is because both the harmonic synthesizer and the subtractive noise synthesizer are treated equally in the synthesis. As the multi-scale STFT loss only considers the summed energy across each frequency bin, it does not penalize the excessive harmonics, especially when the target sound contains mainly noise, with flat energies across each frequency. In figure 2, we show an example harmonic artifact generated by DDSP of a footstep sound. When modelling impact sounds such as footsteps and gunshots, the harmonic plus noise synthesizer tends to assign too many harmonics in its synthesizer. Therefore, we decided to employ a harmonic detector to determine the degree of harmonic components present in the sound and then train the model to attenuate when few harmonics are detected. Specifically, during preprocessing, we scan the entire sound with a pitch detector [13] across every frame and obtain a confidence score C (0-100%) for the pitch estimation of the modelled sound. Rather than using the scores directly as guiding information, we wish to obtain a smoother and flatter curve to 'activate/deactivate' the harmonic synthesizer. To this end, we input C to a custom Sigmoid function to smooth out the output value as shown below:

$$H = 1/(1 + e^{(-a(C-b))})\,, \tag{1}$$

where a determines the steepness of the curve, and b determines the horizontal shift for the Sigmoid function. As both a and b are hyper-parameters, we later found setting $a = 10$ and $b = 0.7$ effective in training on our dataset. Here, we call the output value H from the Sigmoid function as the harmonic indicator. This indicator is further passed as input to the decoder as conditioning information for the presence of harmonic components. When H returns a low value (indicating low confidence of harmonic information present in a frame), the decoder learns to attenuate the harmonic synthesizer and vice versa.

2.2. Transient modelling

From an analysis perspective, a signal in the time domain could be separated into three parts [14], harmonic components, transients (which are modelled as impulses [15, 16], and residual noise. A transient signal (e.g., gunshots, footsteps) has a sharp attack and short sustain, which can be difficult for sinusoidal modelling or subtractive noise modelling. To synthesize transient signals, we employ a similar modelling approach as [15, 10], i.e., synthesizing sinusoids in the discrete cosine domain and converting to the time domain using an inverse Discrete Cosine Transform (IDCT). Due to the nature of time-frequency transforms, this results in impulses in the time domain. Instead of placing transients equal-distantly across the time frames, we instead chose to model them within each frame, as we wish to treat each transient differently and be able to control its timbre. A sinusoid in the DCT domain would translate to a single pulse in the converted time domain through IDCT. The frequency in the DCT domain controls the time location of the pulse, from the start 0ms to 10ms across 160 samples in a frame. Please note that even though the time is extremely short (10ms), changing the time location of the pulse would change the timbre of the impulse clip, as humans are very sensitive to phase changes in transient sounds. For each of the 400 time frames of the 4s long signal, a decoder network will learn and output the parameters of the sinusoids used for transient modelling: i.e., the



(a) Reference footstep (b) DDSP (c) DDSP-SFX

Figure 2: Harmonic artifacts with DDSP. Notice the horizontal lines prevalent in DDSP synthesized footsteps, especially in the first two events. By using a harmonic indicator, we are able to attenuate the harmonic synthesizer and thereby rely mostly on our subtractive noise and transient model for synthesis. To listen to how these sound like, please refer to our accompaniment website: `https://reinliu.github.io/DDSP-SFX/`.

frequency F_n and amplitude A_n. If no transient is present in a given time frame, then A_n should be 0. This provides a convenient means for controlling the transient signals. The transient modelling equation is defined as:

$$x[n] = \sum_{n=0}^{N-1} A_n \text{IDCT} \sin\left(2\pi F_n \frac{t}{f}\right), \tag{2}$$

where $x[n]$ is the modelled transient signal, t is the total time samples, f is the frame size and N is the total number of frames. In order to provide guiding information to the decoder, we extract peak onsets from each sound. The onset amplitude vector is constructed from the signal spectrogram using a margin-based Harmonic Percussive Source Separation (HPSS) method [17] with a conservative and large margin parameter value of 8 during the preprocessing stage. Local maxima (peak estimation) is performed to obtain the onset amplitude vector where the percussive events take place.

2.3. Frame-level timbre control

In addition to guiding the sound synthesis with explicit acoustic features such as pitch and loudness, we also wish to encode the subtle timbre variations of the generated sound into a controllable latent space. To this end, we employ a similar encoder structure introduced by Devis et al. [18] and train DDSP as a Variational Autoencoder (VAE) [19]. We first compute the mel-spectrograms (128 mel-frequency bands, 400 time frames) of our input sounds in the pre-processing stage. They become the input to our encoder, comprising three stacks of convolutional 1-D layers, RELU activation, and batch normalization layers followed by a linear layer. The mean and log-variance output from the encoder are then reparameterized to produce the sampled latent vector z:

$$z = \mu + \epsilon \cdot \sigma^2, \tag{3}$$

where ϵ is a random value sampled from a unit-Gaussian distribution, μ is the mean output from the encoder, $\sigma = e^{0.5\cdot\text{logvar}}$, and logvar is the logarithm of the variance. In this way, we obtain a continuous one-dimensional latent vector along the time axis. The distribution of z is regularized to be close to a unit-Gaussian $\mathcal{N}(0,1)$. After the model has been trained, we can deliberately modify the value of z within ±3 as 99.7% of the values are within three standard deviations. The most "typical" timbres appearing in the data set are encoded with z close to 0, whereas "rare" timbres are encoded with z values farther away from 0. Once the model has been trained, we can completely abandon the encoder structure. In the inference stage, apart from explicit control over pitch and amplitude, we can also output different timbres of the sound (eg. transforming from a BMW car engine sound to a Mercedes) by creating a control variable at hand within the same range ±3 to replace z for the decoder. A global slider is used to adjust this pseudo latent variable and thus vary the timber of the output sound. In this way, we are able to create interesting sound effects by providing explicit and implicit control variables along the time axis. To demonstrate how it works, in section 4, we use human voice mimicking sound effects as an example to guide the sound synthesis.

2.4. Loss Function

Our loss function contains a regularization loss component and a reconstruction loss component. We use the same multi-scale STFT loss as used in DDSP for our reconstruction loss (FFT sizes: 2048, 1024, 512, 256, 128, 64). For the regularization loss, we apply a scaling variable β on the regularization loss term to prevent the reconstruction loss from overtaking the total loss [20]. The total loss function in our model becomes:

$$\mathcal{L} = \mathcal{L}_{rec} + \beta \cdot \mathcal{L}_{reg}, \tag{4}$$

where the $\mathcal{L}$ indicates the total loss, $\mathcal{L}_{rec}$ is the reconstruction loss, and $\mathcal{L}_{reg}$ is the regularization loss.

3. EXPERIMENTS

3.1. Dataset

We use a publicly available sound effects dataset [21]. It includes 7 categories of sound effects, all of which are sampled at 22.5 kHz and have a length of around 4 seconds. Each category contains 581-800 audio samples. The data used for all of the experiments described in this paper consists only of footsteps, gunshots and motor sounds. This is because we found these three categories contain the most variations and they suit the best for our goal as to effectively control the timbre of the sounds. We refer to these three sound types as our three data sets. All sounds are trimmed to 4 seconds duration and down-sampled at 16 kHz to be consistent with our neural network architecture. We split the data 90% for training and 10% for testing.

3.2. Training

We train our model and the vanilla DDSP model on the same three data sets separately, resulting in three trained models for both the vanilla DDSP and our proposed method. Each model is trained with a batch size of 16 on an RTX 6000 GPU for exactly 100,000 training steps. We chose a size of 1024 for the recursive hidden units of DDSP, 100 sinusoids for the harmonic synthesizer, 100 bands for the noise synthesizer, and a frame size of 160 samples. We use an ADAM [22] optimizer with a starting learning rate of $1e^{-4}$, which gradually decayed to $1e^{-5}$ after 80% of the steps. To maintain a stable regularization loss and balance it with the reconstruction loss, we initialize β with 0, and activate it only after 10% of the training steps with $\beta = 1$, and then scale it linearly to $1e^{3}$ until reaching 80% of the training steps.

3.3. Evaluation

To show that our latent space is capable of outputting discernable timbre variations, we performed a small-scale listening test on 26 participants. To guide the synthesis, we recorded three human vocalizations emulating the sound effect for each category of our data set as our out-of-domain guiding sounds. We then use the extracted acoustic features from these guiding sounds to perform timbre transfer. For our latent vector, we manually set z as 0, 0.1, 0.5, 1, 2, and 3 for each sound clip and then generate the sound effects correspondingly. We use $z = 0$ as our reference track and ask the participants to do a forced comparison test to see whether the individual tracks with z set as different values sound identical to or different from the reference. The result is shown in Section 4.3. We evaluate our model in terms of synthesis quality. The timbre encoding performance is demonstrated qualitatively. For each model, we pair a reference sound with the generated sound. Each model is tasked to synthesize waveforms similar to those of the reference by taking in the extracted acoustic features from the reference. We compare the synthesis performance through a series of objective metrics and a subjective listening test. We conduct a second subjective listening test to understand the effectiveness of our timbre encoding.

3.3.1. Synthesis performance

Statistical similarity. Frechet Audio Distance (FAD) [23] is an audio quality evaluation method that compares the feature representations through an embedding layer of a pre-trained audio classification model between the generated sounds and the reference sounds. We compute the FAD score with the VGG model [1] to understand the statistical similarity of our synthesized sounds compared with the reference sounds.

Spectral similarity. We report the log-spectral distance (LSD) using a multi-scale STFT to measure the spectral similarity between the synthesized sound and the reference sounds. It is computed as the average distance between two power spectra over all frames in the Euclidean space. We use the same window length, hop length and FFT size as we did in the pre-processing to compute the spectra.

Audio quality. We conduct a listening test to evaluate the generated sound quality. We randomly selected ten examples per category. For each question, we asked the participants to give an absolute category rating for each soundtrack from 1 (bad) to 5 (excellent). We aggregate our results per category of sounds, meaning that we average the ten-question results and obtain the variance for the aggregated data.

4. RESULTS

4.1. Audio similarity

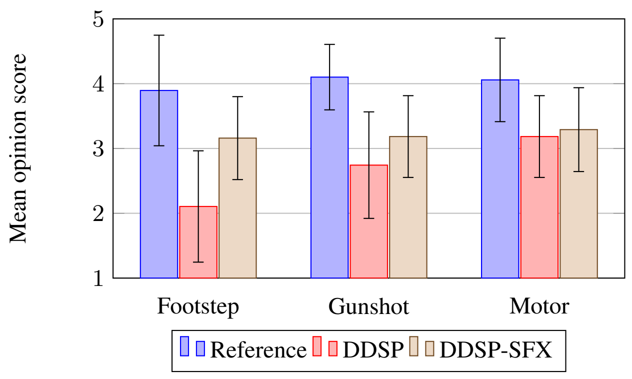
Table 1: *Objective audio synthesis performance results.*

Categories	Footstep	Gunshot	Motor
Metric	FAD ↓		
DDSP	5.356	5.213	**6.652**
DDSP-SFX	**1.529**	**2.004**	7.150
Metric	Log Spectral Distance ↓		
DDSP	0.114	0.585	**0.177**
DDSP-SFX	**0.103**	**0.446**	0.182
Metric	Multi-scale STFT ↓		
DDSP	1.63 ± 0.43	2.23 ± 0.72	**1.09 ± 0.06**
DDSP-SFX	**1.55 ± 0.44**	**2.03 ± 0.81**	1.10 ± 0.07

Referring to Table 1, our objective measures correlate well, suggesting a significant improvement in impulsive sounds (footsteps and gunshots) after we integrated our transient modelling method. For motor sounds that are steadily pitched across the entire signal, our method performs slightly worse than DDSP. This is expected because introducing the regularization term to the loss poses more challenges to the reconstruction. We also show that our transient modelling method can synthesize sharper attacks for impulsive signals in our accompanying website [2].

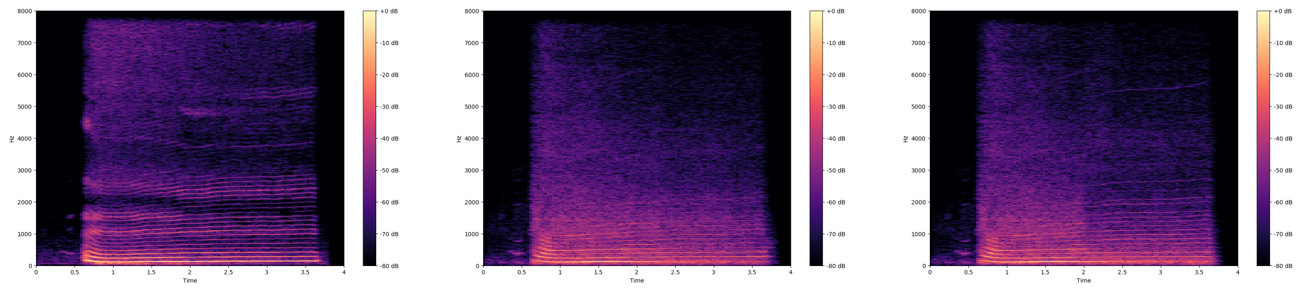
4.2. Audio quality

Figure 3: *User-rated audio quality of sound effects.*



[1]https://github.com/gudgud96/frechet-audio-distance

[2]https://reinliu.github.io/DDSP-SFX/



(a) Reference voice (b) $z = 0$ (c) $z = 1$ after 2s.

Figure 4: An example of how changing z affects the overall timbre. We use a reference voice mimicking motor sound as guidance. In figure 4b, we show what the spectrum looks like when $z = 0$ for all the time frames. In figure 4c, we show that the spectrum changes accordingly when we variate z after 2 seconds. It shows that our latent variable is able to control timbre variation in the time domain.

There were 26 participants (M/F: 19/7; ages: 23-53; audio experts/non-experts: 14/12) in our subjective listening test. Each participant was requested to use a pair of headphones for the listening tests. Each question contains a reference sound (the ones we randomly selected from the dataset), a sound generated from DDSP and a sound generated from ours. The reference tracks are expected to receive the highest scores (4-5). Therefore, we removed two outliers that rated the reference tracks below 2 for over 50% of the questions, resulting in 24 effective participants. Fig. 3 shows a bar plot of the collected mean opinion scores for the sound quality with variance as error bars. Our subjective test results are similar to our objective measures, where we see significant improvements for the impulsive sounds. The footsteps generated by DDSP were rated lower than our method, as many participants noticed the harmonic artifacts. Further, the motor sounds were rated similarly among the two methods.

4.3. Timbre encoding

Table 2: *Percentage of sounds rated as different from reference.*

z	Motors	Gunshots	Footsteps
0.1	0.255	0.235	0.216
0.5	0.451	0.431	0.510
1	0.549	0.608	0.745
2	0.686	0.804	0.843
3	0.941	0.826	/

There were 19 participants (M/F: 11/8; Age: 23-53; Audio experts/non-experts: 9/10) in our second listening subjective test. Table 2 shows the percentage of the participants who rated the synthesized sounds as different from the reference sounds when we vary the value of z. Our subjective test results indicate that most participants recorded timbre differences for $z > 1$. This follows reasonably because the encoder learns to encode the "most typical" timbres across the data set within one standard deviation. Larger values of z indicate a rarer timbre across the whole data set. Depending on the variations available within the dataset itself, the value of z for which users start to tell the timbre differences will likely change accordingly. Lastly, to demonstrate the effectiveness of our timbre encoding, we show how varying z temporally could contribute to timbre variations in Figure 4. For more examples, please refer to our supplement website.

5. DISCUSSION

In this paper, we integrated DDSP with a transient model and show that it improves the synthesis result for impulsive sound effects. We propose a simple method that achieves timbre variation of the generated sound effect while also enabling deterministic attribute transfer given a limited dataset. We further demonstrate the out-of-domain timbre transfer capability by using human vocalization as guiding sounds. We hope our method will contribute to creative sound design by allowing users to create realistic sound effects using their own voices as guiding sounds. Future work may include training our model on a larger audio dataset with more variations to enable more expressive timbre control.

6. REFERENCES

[1] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017, OpenReview.net.

[2] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," *International Conference on Machine Learning*, vol. abs/1704.01279, 2017.

[3] Marco Comunità, Huy Phan, and Joshua Reiss, "Neural synthesis of footsteps sound effects with generative adversarial networks," 10 2021.

[4] Haohe Liu, Zehua Chen, Yiitan Yuan, Xinhao Mei, Xubo Liu, Danilo P. Mandic, Wenwu Wang, and MarkD . Plumbley, "Audioldm: Text-to-audio generation with latent diffusion models," in *International Conference on Machine Learning*, 2023.

[5] Soonil Kwon, "Voice-Driven Sound Effect Manipulation," *International Journal of Human-Computer Interaction*, vol. 28, no. 6, pp. 373–382, June 2012.

[6] Yuki Okamoto, Keisuke Imoto, Shinnosuke Takamichi, Ryosuke Yamanishi, Takahiro Fukumori, and Yoichi Yamashita, "Onoma-to-wave: Environmental sound synthesis from onomatopoeic words," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, pp. –, 2022.

[7] Kohei Suzuki, Shoki Sakamoto, Tadahiro Taniguchi, and Hirokazu Kameoka, "Speak like a dog: Human to non-human creature voice conversion," in *Proceedings of 2022 APSIPA Annual Summit and Conference*, Chiang Mai, Thailand, November 2022, APSIPA.

[8] Riki Takizawa and Shigeyuki Hirai, "Synthesis of Explosion Sounds from Utterance Voice of Onomatopoeia using Transformer," in *28th International Conference on Intelligent User Interfaces*, Sydney NSW Australia, Mar. 2023, pp. 87–90, ACM.

[9] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "Ddsp: Differentiable digital signal processing," *International Conference on Learning Representations*, vol. abs/2001.04643, 2020.

[10] Anton Lundberg, "Data-Driven Procedural Audio: Procedural Engine Sounds Using Neural Audio Synthesis," .

[11] Adrián Barahona-Ríos and Tom Collins, "Noisebandnet: Controllable time-varying neural synthesis of sound effects using filterbanks," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 32, pp. 1573–1585, feb 2024.

[12] Xavier Serra and Julius Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12, 1990.

[13] Jong Kim, Justin Salamon, Peter Li, and Juan Bello, "Crepe: A convolutional representation for pitch estimation," *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, 04 2018.

[14] Leonardo Fierro and Vesa Valimaki, "Sitrano: A matlab app for sines-transients-noise decomposition of audio signals," in *2021 24th International Conference on Digital Audio Effects (DAFx)*, 2021, pp. 73–80.

[15] T.S. Verma and T.H.Y. Meng, "An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, Seattle, WA, USA, 1998, vol. 6, pp. 3573–3576, IEEE.

[16] Tony S. Verma and Teresa H. Y. Meng, "Extending Spectral Modeling Synthesis with Transient Modeling Synthesis," *Computer Music Journal*, vol. 24, no. 2, pp. 47–59, 06 2000.

[17] Jonathan Driedger, Meinard Müller, and Sascha Disch, "Extending harmonic-percussive separation of audio signals," in *International Society for Music Information Retrieval Conference*, 2014.

[18] Ninon Devis, Nils Demerlé, Sarah Nabi, David Genova, and Philippe Esling, "Continuous Descriptor-Based Control for Deep Audio Synthesis," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, June 2023, pp. 1–5, IEEE.

[19] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," *International Conference on Learning Representations*, vol. abs/1312.6114, 2013.

[20] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2016.

[21] Keunwoo Choi, Jaekwon Im, Laurie Heller, Brian McFee, Keisuke Imoto, Yuki Okamoto, Mathieu Lagrange, and Shinosuke Takamichi, "Foley sound synthesis at the dcase 2023 challenge," *In arXiv e-prints: 2304.12521*, 2023.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, vol. abs/1412.6980, 2014.

[23] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *Interspeech*, 2019.