A Plugin for Neural Audio Synthesis of Impact Sound Effects

Zih-Syuan Yang
Sound and Music Analysis Group
Digital Media Technology Lab
School of Computing and Digital Technology
Birmingham City University
Birmingham, UK
zih-syuan.yang@mail.bcu.ac.uk

Jason Hockman
Sound and Music Analysis Group
Digital Media Technology Lab
School of Computing and Digital Technology
Birmingham City University
Birmingham, UK
jason.hockman@bcu.ac.uk

ABSTRACT

The term *impact sound* as referred to in this paper, can be broadly defined as the sudden burst of short-lasting impulsive noise generated by the collision of two objects. This type of sound effect is prevalent in multimedia productions. However, conventional methods of sourcing these materials are often costly in time and resources. This paper explores the potential of neural audio synthesis for generating realistic impact sound effects, targeted for use in multimedia such as films, games, and AR/VR. The designed system uses a Realtime Audio Variational autoEncoder (RAVE) [2] model trained on a dataset of over 3,000 impact sound samples for inference in a Digital Audio Workstation (DAW), with latent representations exposed as user controls. The performance of the trained model is assessed using various objective evaluation metrics, revealing both the prospects and limitations of this approach. The results and contributions of this paper are discussed, with audio examples and source code made available.

KEYWORDS

neural audio synthesis, sound effect synthesis, deep generative learning, impact sound

ACM Reference Format:
Zih-Syuan Yang and Jason Hockman. 2023. A Plugin for Neural Audio Synthesis of Impact Sound Effects. In *Audio Mostly 2023 (AM '23), August 30–September 01, 2023, Edinburgh, United Kingdom.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3616195.3616221

Figure 1: Synchronisation of customised impact sound effect in a digital audio workstation.

1 INTRODUCTION

When it comes to sourcing sound materials for multimedia such as games, films and AR/VR technologies, the use of large databases consisting of numerous audio recordings and sound effects is a common practice. Generally, this involves seeking through a library and auditioning one sample after another before fine-tuning and aligning to asynchronous visual cues. Although this could be regarded as a standard workflow in the industry, it is often a time-consuming process, especially when the desired sound possesses unique acoustic or timbral characteristics [5]. Aside from pre-recorded samples, audio synthesis is a popular technique for generating tailor-made sounds. While versatile, different synthesis methods have their sets of advantages and limitations; a plugin implementation of an algorithm will typically only target one type of sound [11], and the resultant product can often be audibly synthetic and lack realism [9]. Traditionally, in films and TV shows, sound effects are dubbed over by professional Foley artists, who use props to generate and record the idea of everyday sounds [13]. In the blooming industries of games and VR/AR technologies, the demand for realistic audio with subtle tonal quality differences in each action instance is even more crucial to provide context-specific and immersive soundscapes. Dependency on largely repetitive sample libraries or data augmentation with signal processing techniques is not ideal and usually lacks semantic controls.

With the high cost of resources and time for high-fidelity samples and professional recordings, this paper aims to provide an alternative by offering a way to generate and manipulate realistic impact sound effects with minimal and intuitive tonal controls. The proposed plugin is designed to support audio production for multimedia, ensuring smooth integration with existing workflows through DAWs, and providing a seamless solution to aforementioned challenges.

AM '23, August 30–September 01, 2023, Edinburgh, United Kingdom

ACM ISBN 979-8-4007-0818-3/23/08…$15.00
https://doi.org/10.1145/3616195.3616221

1.1 Background

Advances in deep learning algorithms have led to a substantial increase in applications across various domains; one of the most noteworthy being photorealistic image generation [12]. This technology has been adopted in the area of audio generation, primarily focusing on speech and pitched instrument synthesis. Several studies have explored the use of neural synthesis for broadband impulsive sounds, particularly in the context of drum samples, which share tonal similarities with impact sounds. Past research on the synthesis of footstep and knocking sound effects using both neural networks [1, 3] and signal processing techniques [4, 7, 14] provided great inspiration for this project. However, recent developments in deep learning have yet to be fully explored in the field of sound effect synthesis for multimedia. While past research in impact sound generation often uses modal synthesis, novel architectures in deep learning introduce the possibility of creating sounds using alternative methods [1]. This paper aims to contribute to the less-researched field of neural audio synthesis of sound effects by investigating the potential of state-of-the-art generative deep learning technology for impact sound generation.

2 METHOD

2.1 RAVE

The designed application builds upon a RAVE model as proposed by Caillon and Esling [2], which overcomes the common limitation of low synthesis quality in variational autoencoders (VAEs) by introducing a novel training procedure consisting of two stages: *representation learning* and *adversarial fine-tuning*. The model first undergoes regular VAE training to retrieve high-level attributes of the dataset before being fine-tuned through an adversarial generation objective, attaining high-quality audio synthesis. In addition, by incorporating a multi-band decomposition step of raw audio in conjunction with classical synthesis techniques, RAVE is able to achieve a sampling rate of 48kHz with performance 20 times faster than real-time on a standard CPU and no substantial increase in computational complexity. This addresses the processing-intensive nature of raw waveform representation and enables efficient real-time synthesis. Lastly, the dimensionality of the learnt representation is restricted to a minimum using singular value decomposition in order to find the informative parts and aid in the compactness and manipulation of latent trajectories.

In the first stage, a multiscale spectral loss $S(\cdot,\cdot)$ introduced by Engel et al. [6] is employed to measure the distance between synthesised and real samples, which encourages the model to learn key perceptual qualities of the data without penalising for inaccurately reconstructed phase. This is used in conjunction with the loss derived from ELBO for the encoder and decoder training until convergence.

$$\mathcal{L}_{VAE} = \mathbb{E}_{\hat{x}\sim p(x|z)}[S(x,\hat{x})] + \beta \times \mathcal{D}_{KL}[q_\phi(z|x) \parallel p(z)] \quad (1)$$

In the above equation, x, $\hat{x}$ and z represent the real sample, generated sample and latent variable respectively. $q_\phi(z|x)$ and $p(z)$ each denote the approximate posterior distribution of z given x and the prior distribution of z. The symbol $\parallel$ signifies that the Kullback-Leibler (KL) divergence, $\mathcal{D}_{KL}$, measures the difference between the two distributions. Lastly, β is a weighting factor used to balance the multiscale spectral loss and KL divergence term in the overall loss function. The second stage focuses on enhancing the quality and naturalness of the synthesised audio. This is achieved by freezing the learnt latent representation of the encoder and using it as the base distribution for GANs. The decoder then samples from this and generates new data in adversary to discriminator D. The hinge loss version of the GAN objective is used for training, which is defined as:

$$\begin{aligned}\mathcal{L}_{dis}(x,z) &= \max(0, 1 - D(x)) + \mathbb{E}_{\hat{x}\sim p(x|z)}[\max(0, 1 + D(\hat{x}))] \\ \mathcal{L}_{gen}(z) &= -\mathbb{E}_{\hat{x}\sim p(x|z)}[D(\hat{x})]\end{aligned} \quad (2)$$

In addition to the adversarial objective, the aforementioned spectral distance and a feature matching loss are added to obtain the final objective for the decoder:

$$\mathcal{L}_{total}(x,z) = \mathcal{L}_{gen}(z) + \mathbb{E}_{\hat{x}\sim p(x|z)}[S(x,\hat{x}) + \mathcal{L}_{FM}(x,\hat{x})] \quad (3)$$

Proposed by Kumar et al. [10], the feature matching loss $\mathcal{L}_{FM}$ can be considered as a learned similarity metric with the objective to minimise the $\mathcal{L}_1$ distance between the discriminator feature maps of real and synthetic audio. Similarly, the discriminator D detailed in [10] is used. This features three Markovian window-based discriminators with identical network structures operating on different audio scales, creating a multiscale framework. The first discriminator, D_1, focuses on the raw audio, while the second and third discriminators, D_2 and D_3, target the same signal but downsampled by factors of 2 and 4. By employing this multiscale approach, each discriminator is able to learn features specific to different ranges of audio, which in turn facilitates the acquisition of an inductive bias related to the overall audio structure.

Additionally, the RAVE model provides novel post-training analysis of the latent space which enables direct control over the trade-off between representation compactness and reconstruction fidelity using fidelity parameter f [2]. This works by employing singular value decomposition (SVD) on a centred matrix derived from the latent representation, separating the informative and uninformative parts of the latent space. By defining f for use with the associated rank of the SVD, latent dimensionality is able to be reduced by projecting the concatenation of a low-rank representation and noise sampled from the prior distribution.

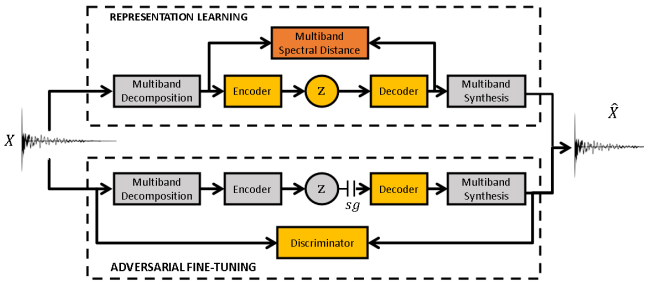

Figure 2: RAVE architecture (Callion and Esling [2]). The yellow blocks are optimised and the grey blocks are frozen during the training stages.

2.2 Dataset

The collected dataset consists of just over 3000 single impact recordings taken from Zapsplat,[1] a website which hosts a large variety of free sound effects. The collected samples consist of professional Foley recordings. Several key search terms, including *single footstep, hit, impact, tapping*, and *knocking*, are used in the *Foley* category on the site. From these, 122 samples are separated from the training set for evaluation, ensuring fair results. These samples feature individual footstep recordings of sneakers on various surfaces, with different levels of force or actions.

2.3 Training

Using the official RAVE source code provided on GitHub,[2] the original implementation of the model as described in [2] is obtained, and training is conducted using the collected samples. The model is trained for a total of 1.5 million steps with 1 million steps for the first stage and 0.5 million for the second on a Tesla T4 GPU, running over a span of 3 days and 17 hours. The Adam optimisation algorithm [8] is deployed using the PyTorch library,[3] with a learning rate α of 10^{-4}, β of $(0.5, 0.9)$, and a batch size of 8, in accordance with the default RAVE configuration. Upon completion of training, the model is exported as a TorchScript file, which can be loaded for inference purposes. The trained model is exported using two different values of f for comparison: 0.9 and 0.95, which result in representation lengths of 16 and 32 respectively in the second dimension.

2.4 Audio Plugin

The plugin utilises the encoder of the trained model and allows the user to load their own audio samples into the latent space. The encoded representation can be manipulated as desired with exposed latent controls, altering its key tonal qualities. For every modification to the representation, the signal is decoded and can be played back for evaluation and further adjustments. Sample playback is triggered via MIDI note-on messages, which allows for automated playback with the use of a piano roll in a DAW. This feature closely resembles a sampler and enables users to place MIDI notes corresponding to visual cues on a timeline. Additionally, as real-world impact sounds generated by even identical objects commonly possess slightly different qualities every time, a randomisation feature was added to simulate this natural variation. This feature sits at the end of the signal chain and alters the pitch and volume of each sample playback. The degree of this alteration depends on a user-defined parameter, which specifies the range of the randomisation effect. Upon receiving a MIDI note-on message, the plugin generates two random numbers for pitch and volume modification before multiplying them by the random control parameter. These values are then used respectively for calculating the resampling ratio and gain. The signal flow diagram including this feature can be seen in Figure 3.

The plugin is written in C++ with the use of the JUCE framework.[4] JUCE provides the basis and tools for the development of audio and MIDI processing, graphical user interface, as well as the handling of various operating systems for the application. LibTorch[5] is employed to interface with the pre-trained PyTorch model. It is a binary distribution of the PyTorch library that provides an API for integrating PyTorch models in C++ applications. The plugin is developed in consideration of functional cross-platform operation.

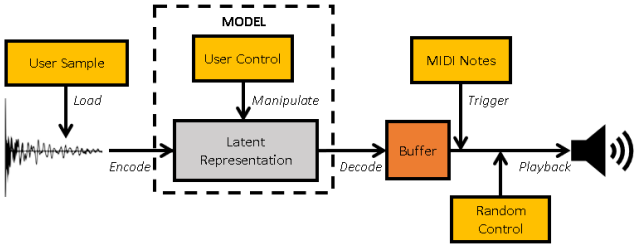

Figure 3: Audio plugin signal flow. The loaded raw signal is encoded into a latent representation that can be manipulated. This is then decoded and stored in a buffer for MIDI note triggers to playback.

3 EVALUATION

3.1 Evaluation Methodology

In order to assess the generations of the trained model, objective evaluation metrics are used. Although there are currently no standardised methods for the reliable assessment of the quality and diversity of synthesised audio, some metrics are widely adopted for the analysis and comparison of neural synthesis models [3]. For this particular model, Maximum Mean Discrepancy (MMD) and Frechet Audio Distance (FAD) are utilised to obtain the difference between synthesised and real samples by comparing their embedding distributions. The evaluations are carried out on the trained RAVE models exported with different fidelity values specified.

3.2 Latent Controls

Using a fidelity parameter of 0.9 upon export, the encoded representation consists of three dimensions, where the third dimension N is dependent on the length of the original input signal. The latent control feature is designed to provide users with the ability to modify the elements of a specified vector in the second dimension by adding or subtracting a scalar value, *delta*. The new representation is calculated using the following formula:

$$L_{new}[i] = L_{old}[i] + delta, for i = 0, 1, \cdots, N \tag{4}$$

To observe the impact of each vector on the tonal quality of a signal, a separate plugin build which exposes all latent controls is generated for exploration.

The first five vectors of the second dimension in the latent representation are selected to be added in the final plugin build, named *Splatter, Volume, Tail, Boost* and *Force*. The effect of tweaking two of the controls respectively is demonstrated in Figure 4.

[1] https://www.zapsplat.com/
[2] https://github.com/acids-ircam/RAVE
[3] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[4] https://juce.com/
[5] https://pytorch.org/cppdocs/

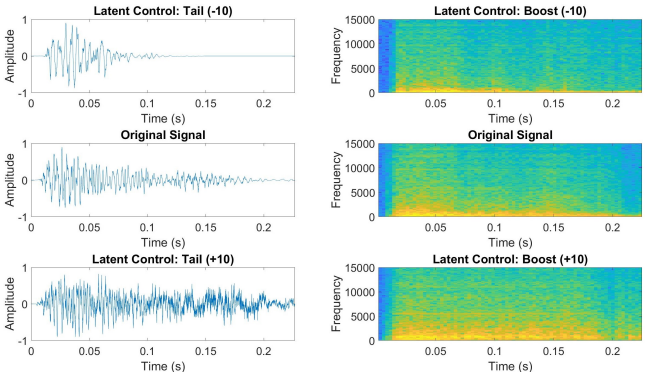

Figure 4: Latent control demonstration. The first and second columns show the effect of increasing the *Tail* and *Boost* control respectively.

Table 1: Evaluation Metrics Results. MMD and FAD scores measuring the differences between two datasets are noted.

MMD: avg / std	`Model095`	`Model090`
`Eval`	30.05 / 2.6	32.49 / 2.48
`Model095`	-	3.35 / 0.74
FAD Score	`Model095`	`Model090`
`Eval`	5.2	5.78
`Model095`	-	0.32

3.3 Results and Discussion

MMD and FAD scores are computed on the reconstructions of the evaluation set by the two models exported using an f of 0.95 and 0.9 respectively. For clarity, the former will be referred to as `Model095` and the latter as `Model090`. The evaluation set is referred to as `Eval`.

The top half of Table 1 presents the MMD results, displaying both the average MMD score and the standard deviation for each pair of datasets. `Model095` achieved a higher average, indicating better reconstruction quality and greater similarity with the evaluation set. This is expected because the lower fidelity parameter specified for `Model090` led to a more compact latent space, but also a poorer approximation of the input due to the prior noise replacing less critical latent information. Furthermore, `Model090` has failed to reproduce some data variation present in the original data, evidenced by the lower standard deviation score. A single FAD score is generated for each dataset and shown in the bottom half of Table 1, representing the perceived quality and distance measure. As a lower score indicates better quality, `Model095` is again demonstrated to be more similar to the ground truth and superior in fidelity. This further emphasises the impact of the fidelity parameter on reconstruction quality, with `Model095` outperforming `Model090` in a manner consistent with the MMD results discussed above. This echoes the importance of selecting an appropriate fidelity parameter to achieve desired performance with regard to the balance between reconstruction quality and representation compactness as illustrated in [2].

4 CONCLUSIONS AND FUTURE WORK

In this work, we offer a novel solution using neural audio synthesis, addressing the challenges faced by a large population of audio professionals and hobbyists in sourcing and generating various impact sounds. An audio plugin is developed that enables seamless integration with existing workflows and provides intuitive tonal controls. A RAVE model is trained on over 3,000 impact sound samples, with the results evaluated using objective metrics. The evaluation results demonstrate that the model achieves excellent MMD scores, indicating high reconstruction quality. However, the FAD scores revealed improvements can still be made in enhancing the perceptual quality of generations. The trade-off between latent space compactness and reconstruction quality is also emphasised. Plugin source code and accompanying audio examples are available online. [6] Future work could focus on refining the RAVE model to be more suitable for short-lasting signals and improving its perceptual quality. Performing subjective tests can also provide insights into the usefulness and quality of generations. Increasing the duration and variety of the training dataset could also enhance generalisation to unseen data.

REFERENCES

[1] Adrián Barahona-Ríos and Sandra Pauletto. 2020. Synthesising Knocking Sound Effects Using Conditional WaveGAN.
[2] Antoine Caillon and Philippe Esling. 2021. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. https://doi.org/10.48550/arXiv.2111.05011 arXiv:2111.05011 [cs, eess].
[3] Marco Comunità, Huy Phan, and Joshua D Reiss. 2022. Neural Synthesis of Footsteps Sound Effects with Generative Adversarial Networks. (2022), 9.
[4] Perry R. Cook. 2002. Modeling Bill's Gait: Analysis and Parametric Synthesis of Walking Sounds. Audio Engineering Society. https://www.aes.org/e-lib/browse.cfm?elib=11153
[5] Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial Audio Synthesis. https://openreview.net/forum?id=ByMVTsR5KQ
[6] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. 2019. DDSP: Differentiable Digital Signal Processing. https://openreview.net/forum?id=B1x1ma4tDr
[7] Andy Farnell. 2007. Marching onwards Procedural synthetic footsteps for video games and animation. https://www.semanticscholar.org/paper/Marching-onwards-Procedural-synthetic-footsteps-for-Farnell/4364dbb2433c98da69b138245a26af4dd2aa5f5d
[8] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980 arXiv:1412.6980 [cs].
[9] Tim Kirby and Mark Sandler. 2020. Advanced Fourier Decomposition for Realistic Drum Synthesis. Vienna, Austria, 155.
[10] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. 2019. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., Vancouver, Canada. https://papers.nips.cc/paper/2019/hash/6804c9bca0a615bdb9374d00a9fcba59-Abstract.html
[11] David Moffat, Rod Selfridge, and Joshua D. Reiss. 2020. Sound Effect Synthesis. In *Foundations in Sound Design for Interactive Media* (1 ed.). Routledge, 274–299. https://doi.org/10.4324/9781315106342-13
[12] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. https://doi.org/10.48550/arXiv.2112.10752 arXiv:2112.10752 [cs].
[13] Jim Stinson. 1999. Real-time Sound Effects: The Foley Way. https://www.videomaker.com/article/7220-real-time-sound-effects-the-foley-way/
[14] Luca Turchet. 2016. Footstep sounds synthesis: Design, implementation, and evaluation of foot–floor interactions, surface materials, shoe types, and walkers' features. *Applied Acoustics* 107 (June 2016), 46–68. https://doi.org/10.1016/j.apacoust.2015.05.013

[6] https://ann-yang00.github.io/simpact/