SLIDERSOUND: TRANSFERRING CONTINUOUS SEMANTIC CONTROLS FROM PROCEDURAL TO PRETRAINED NEURAL SOUND EFFECTS MODELS

Yisu Zong, Joshua Reiss

Queen Mary University of London
Centre for Digital Music

ABSTRACT

Sound effects design involves creating and modifying sounds to meet production requirements and creative intentions. Text-to-audio models can support this process by generating high-quality material from natural language descriptions, but precise and continuous control over individual sound properties remains challenging. Procedural audio models provide such control through explicit and meaningful parameters, although their simplified synthesis methods can limit sound realism. To address these limitations, this paper presents a method for transferring continuous semantic controls from procedural models to a pretrained neural sound effects model, enabling parameter-based adjustment of sounds generated from text. We train a control adapter using supervision from paired procedural sounds to learn the changes caused by parameter adjustments and apply corresponding changes to neural outputs. Our evaluations across multiple sound effect categories demonstrate the effectiveness of the proposed method.

Index Terms— Sound effects generation, Controllable sound synthesis, Procedural audio, Adapter learning

1. INTRODUCTION

Sound designers commonly work with recordings captured for a production or from existing sound libraries, then edit and process them to suit the intended context. These decisions reflect both production requirements and the designer's aesthetic preferences, allowing different sonic interpretations of the same event. A production may require several variations, such as footsteps at different walking speeds or explosions with different decay times. Preparing suitable material through recording and manual processing can be time-consuming. Direct sound synthesis offers a complementary way to produce new material and variations. However, its practical value for sound design depends on both the sound quality and the ability to shape their characteristics.

Neural sound effects models have shown promising performance in generating high-quality and varied sounds. Text-conditioned sound generation systems [1, 2, 3] allow users to describe sound sources and their characteristics through natural language. However, plain text provides a discrete, descriptive interface without directly exposing continuously adjustable parameters for individual sound properties, which limits its practical application. For example, requesting "longer explosion decay" expresses the desired change but does not provide a direct control over the extent of that change. Several methods provide more explicit control through temporal envelopes and time-varying acoustic features, which can be extracted from vocal imitations [4] or specified directly as control curves [5]. Sound morphing methods explore intermediate and hybrid sounds through interpolation [6] and prompt manipulation [7, 8].

Procedural models (PM) offer a different way to control sound generation through explicit parameters [9], allowing users to generate variations through physically or perceptually meaningful controls, but the design's simplifying assumptions can limit sound realism. One approach to improving model performance is to expose more parameters, but this comes at the expense of reduced ease of control [10]. Subsequent work [11] trained a neural synthesizer on procedural examples and adapted it to real recordings. However, the resulting sound quality and control ability can be sensitive to recording quantity and diversity, sound-matching accuracy, and parameter coverage. These dependencies can make it difficult to extend this approach across a broader range of sound effects.

In this paper, we propose SliderSound, a method for transferring continuous semantic controls from PM to a pretrained text-to-audio model, allowing the control to build on existing synthesis capabilities rather than learning realistic sound generation from limited recordings. We formulate control learning through the acoustic changes caused by parameter adjustments. Differences between procedural sounds before and after an adjustment supervise corresponding changes relative to the original neural generation. This formulation uses procedural audio to specify the effect of a control without requiring corresponding real recordings or the neural model to reproduce the procedural sound. A control adapter learns these adjustments while the pretrained generator remains frozen. We evaluate the method in three sound effect categories: explosion, footsteps, and engine, assessing the accuracy and consistency of the learned controls and their effect on sound

quality. Audio samples are provided at `https://zys711.github.io/SliderSound/`.

2. PROPOSED METHOD

SliderSound learns relative control by transferring the changes produced by a PM to text-generated sounds. As illustrated in Fig. 1(a), we represent the acoustic change of a parameter adjustment as the difference between the CLAP [12] embeddings of the procedural sounds before and after the adjustment, motivated by evidence that CLAP embeddings retain perceptually relevant attributes [13]. This change is used to construct a target relative to the text-only neural output. A control adapter conditioned on the same adjustment is trained to move the neural output's embedding towards this target, while the pretrained generator remains frozen. At inference as shown in Fig. 1(c), the generator and learned adapter generate sounds conditioned on text and the signed parameter adjustments.

Let G_θ denote the pretrained text-to-audio generator with frozen parameters θ, and let ϕ denote the trainable control adapter parameters. Given a text prompt s, initial latent noise ξ, and a signed parameter adjustment $\Delta\mathbf{c} \in R^K$ with the control parameter number K, the original and controlled outputs are $x_0 = G_\theta(s, \xi)$, $x_\Delta = G_{\theta,\phi}(s, \xi, \Delta\mathbf{c})$. Each coordinate of $\Delta\mathbf{c}$ corresponds to a procedural control, with its sign and magnitude specifying the requested adjustment. The adjustment is relative to the original neural generation; it does not specify an absolute procedural configuration for that sound.

2.1. Procedural Supervision

Let $S(\mathbf{c}, \eta)$ denote a PM output sound with normalized parameters $\mathbf{c}$ and randomness seed η. In order to learn the relative control instead of forcing the model to learn PM sounds, we construct a bank of procedural sounds $\mathcal{B}$ with parameter sets $\mathcal{C} = \{(\mathbf{c}_i, \eta_i)\}_{i=1}^N$ and encode them using a frozen LAION-CLAP [14] audio encoder with unit-normalized output $q(S(\mathbf{c}, \eta))$. For a random sound effect, not all parameter adjustments are practically meaningful. For example, a large portion of rumble decay cannot be removed from a short explosion, since this component barely exists in the original signal. Therefore, for each original neural output x_0, we retrieve a procedural sound with similar acoustic characteristics by maximizing cosine similarity:

$$(\mathbf{c}_*, \eta_*) = \underset{(\mathbf{c},\eta)\in\mathcal{C}}{\arg\max}\, sim(q(x_0), q(S(\mathbf{c}, \eta))). \tag{1}$$

The selected source remains fixed across all adjustments associated with that neural reference.

For a requested adjustment $\Delta\mathbf{c}$, the matched source supplies an original and controlled pair: $a = S(\mathbf{c}_*, \eta_*)$, $b = S(\mathbf{c}_* + \Delta\mathbf{c}, \eta_*)$, The procedural embedding change defines the target embedding $\mathbf{q}^*_\Delta$ for the controlled neural output:

$$\begin{aligned} \mathbf{r}^q &= q(b) - q(a), \\ \mathbf{q}^*_\Delta &= \frac{q(x_0) + \mathbf{r}^q}{\|q(x_0) + \mathbf{r}^q\|_2}. \end{aligned} \tag{2}$$

As a result, the training objective is

$$\mathcal{L} = \|q(x_\Delta) - \mathbf{q}^*_\Delta\|_2^2. \tag{3}$$

2.2. Control Adapter

We select Woosh-Flow [3] as our generator backbone, and build the adapter upon it. Woosh-Flow is trained with flow matching in the latent space of Woosh-AE and conditioned on text representations from Woosh-CLAP. As shown in Fig. 1(b), we insert a residual adapter A_ℓ after each of its 12 transformer blocks. A shared bias-free linear control encoder W_c maps $\Delta\mathbf{c}$ to a 64-dimensional control embedding. Each adapter projects transformer block representations $\mathbf{h}_\ell$ to 32 dimensions, applies Swish [15], and multiplies the features element-wise by a bias-free projection of the control embedding, broadcast across sequence positions. A bias-free up-projection restores the original representation dimension before residual addition. Each block's representation $\mathbf{h}_\ell$ is updated as

$$\mathbf{h}'_\ell = \mathbf{h}_\ell + A_\ell(\mathbf{h}_\ell, W_c\Delta\mathbf{c}), \tag{4}$$

The bias-free control pathway ensures that an all-zero adjustment yields zero residuals throughout training, so zero-parameter inputs maintain generation conditioned solely on the input text.

2.3. Training

We follow the differentiable optimization strategy of DRaFT [16] to train the adapter through the complete generation process. For each training example, x_Δ is generated using the same text prompt and initial noise as the precomputed x_0. Gradients propagate through the frozen CLAP, Woosh-AE decoder, and all sampling steps to update only the adapters and shared control encoder.

For footsteps training, we additionally add one aligned temporal loss, because CLAP embeddings provide weak supervision for brief changes in individual footfall envelopes. We align a (defined in Section 2.1) with x_0 during data preparation by triggering the footfalls in a using reviewed timestamps from x_0. When generating b, non-"pace" adjustments preserve this schedule, while "pace" adjustments modify event timing and allow footfalls to enter or leave the fixed-duration window. Using a framewise log-RMS extractor E, we compare the neural and procedural envelope changes:

$$\begin{aligned} r &= E(x_\Delta) - E(x_0), \\ r^* &= E(b) - E(a), \\ \mathcal{L}_{\text{align}} &= \frac{\lambda}{T}\sum_{n=1}^{T} |r[n] - r^*[n]|, \end{aligned} \tag{5}$$

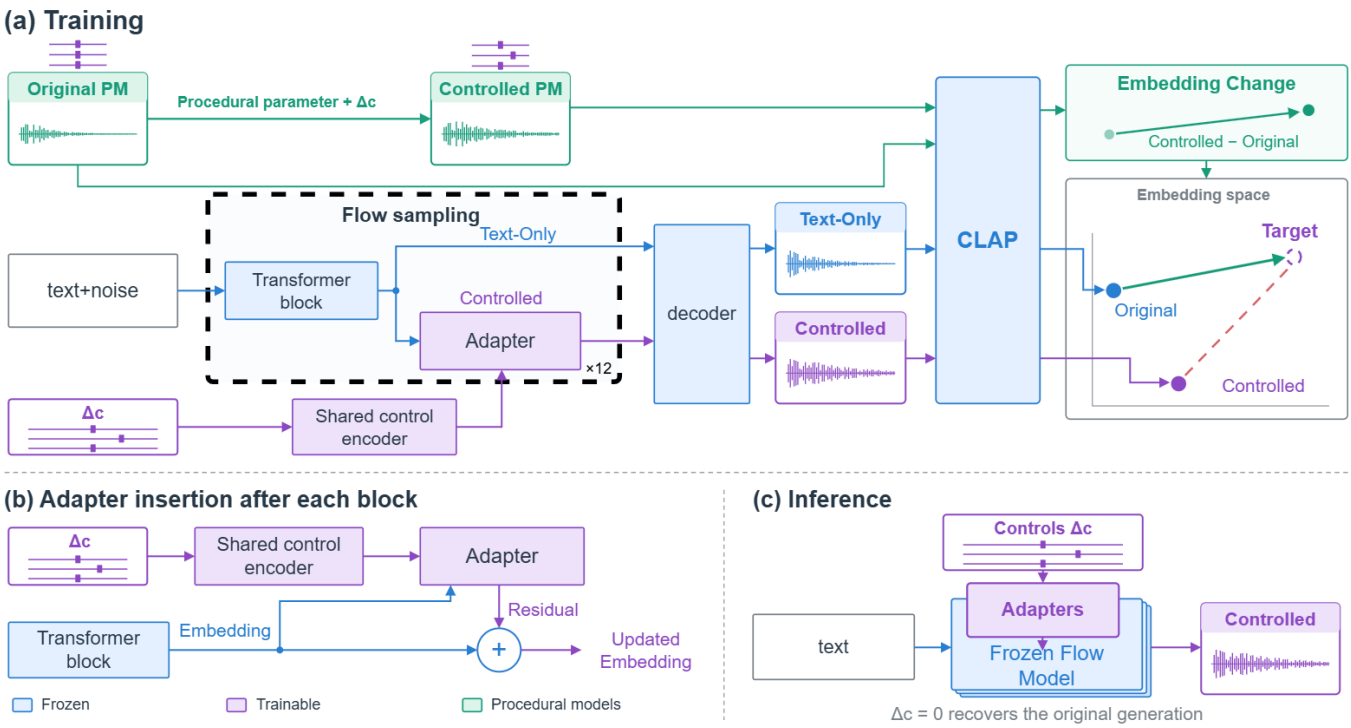

Fig. 1. Overview of SliderSound. (a) Procedural CLAP embedding changes define training targets relative to text-only outputs. (b) Adapters add control-dependent residuals to transformer audio-token representations. (c) Inference uses text and signed parameter adjustments.

where n indexes temporal frames, T is their total number, λ is the weighting coefficient.

3. EXPERIMENTS

3.1. Procedural Audio Models

We evaluate the method on three sound categories and the corresponding PMs: explosion [1], footsteps [2], and engine [3]. These models are physically inspired signal processing models, where the design concepts are derived from the implementation by Andy Farnell [17]. We select 8, 13, and 14 exposed continuous control parameters, respectively, from the explosion, footsteps and engine models. We train a separate adapter for each sound category while retaining the same pretrained backbone.

3.2. Data

Training Data: For each category, we construct a procedural sound bank that contains 20,000 procedural sounds by sampling normalized parameter settings and generating 48 neural output variations using frozen Woosh-Flow with category-specific prompts. All audio is 5 seconds long at 48 kHz. Following Section 2.1, each neural output is associated with one procedural sound, which remains fixed across its parameter adjustments.

We sample 512 adjustment requests for each neural output, each changing one parameter with a non-zero magnitude of up to 0.5 within available range. Requests are balanced across parameters and positive and negative adjustments. Applying these requests to the neural outputs gives 24,576 candidate training examples.

Evaluation Data: For the purpose of sound quality evaluation, we curate 45 explosion recordings from Pro Sound Effects [4], 40 footsteps recordings from ESC-50 [18], and 48 recordings from a subset of HL-CEAD [19].

3.3. Baselines and Evaluation Metrics

We compare our method with the original Woosh-Flow denoted as Text-only; NeuralPM Supervised(interpolation) version [11], a neural network that learns control from PM and trains on real recording-PM sound pairs, where the valid parameter range depends on the training pairs.

We evaluated our model's generated sound quality and control ability. For sound quality evaluation, we adopted Fréchet Audio Distance (FAD) [20] and Maximum Mean Discrepancy (MMD) [21]. To assess the control capability, we followed [11] and employed Spearman's rank correlation coefficient to evaluate the relationship between high-level

[1] https://nemisindo.com/models/explosion
[2] https://nemisindo.com/models/footsteps
[3] https://nemisindo.com/models/engine-advanced
[4] https://www.prosoundeffects.com/

Table 1. Control correlations across three sound effect categories.

	Explosion					Footsteps					Engine				
Method	mean↑	Boominess	Brightness	Roughness	Depth	mean↑	Hardness	Brightness	Depth	Sharpness	mean↑	Brightness	Depth	Roughness	Warmth
SliderSound	**0.817**	0.781	0.925	0.725	0.836	**0.453**	0.320	0.889	-0.246	0.849	**0.570**	0.866	0.456	0.053	0.906
Text-only	0.136	-0.358	0.782	0.329	-0.208	0.338	0.451	0.515	0.455	-0.067	-0.072	0.329	-0.115	-0.289	-0.212
NeuralPM	0.659	0.557	0.496	0.695	0.887	0.188	0.248	0.361	-0.246	0.388	0.441	0.294	0.418	0.422	0.631

Table 2. Sound quality across three sound effect categories.

	Explosion		Footsteps		Engine	
Method	FAD ↓	MMD ↓	FAD ↓	MMD ↓	FAD ↓	MMD ↓
SliderSound	**7.943**	**75.818**	**9.030**	**48.104**	34.097	198.969
Text-only	8.051	75.882	9.039	48.418	**33.195**	**190.464**
NeuralPM	9.992	87.653	27.275	150.747	42.336	213.160

audio features in the original PM outputs and model outputs. The selected features for each sound category listed in Table 1 are from Audio Commons project [5].

4. RESULTS AND DISCUSSION

4.1. Sound Quality

For quality evaluation, the test set contains 128 sounds per model and category, where random relevant description prompts are used for Text-only, and SliderSound has the same prompts and random valid parameters; NeuralPM has random valid parameters only.

The sound quality results in Table 2 show that SliderSound achieves lower FAD and MMD than NeuralPM across all three sound categories, supporting the use of a pretrained generator as the foundation for control learning. For explosions and footsteps, both distances are also slightly lower than those of Text-only. These results suggest that the adapter can possibly build on the original model's synthesis capabilities while refining its output distribution toward real recordings. One possible explanation is that procedural supervision provides plausible directions of variation, allowing parameter adjustments to improve distributional agreement in generated sounds. However, the higher distances for engines indicate that this benefit is category-dependent and that the learned adjustments do not consistently improve sound quality.

4.2. Control Ability

For control evaluation, we vary each parameter over 20 equally spaced values within the valid range while keeping the remaining parameters fixed for SliderSound and NeuralPM, where SliderSound is the relative normalized range [-0.5, 0.5], and NeuralPM is the absolute normalized range [0,1]. Since Text-only does not accept numerical parameters, we evaluate its control ability by appending 10 ordered descriptor phrases to the prompt, such as "very short rumble decay" to "very long rumble decay" for "rumble decay", while keeping the generation seed fixed.

For each audio feature, Table 1 reports the mean correlation across all control parameters, together with the overall mean across features. SliderSound achieves higher overall means than Text-only and NeuralPM across all three categories, suggesting that procedural supervision provides a more consistent relationship between requested adjustments and acoustic variations than ordered text descriptions alone. However, the higher mean conceals uneven performance: footstep brightness and sharpness show strong correspondence, while hardness performs below Text-only and depth is negatively correlated.

At the parameter level, some controls also behave unexpectedly. For example, for footsteps, varying the pace parameter did not change the number of footfalls within the fixed-duration output, despite supervision from both the CLAP and aligned temporal losses. Text-only successfully changes stepping pace, although precisely specifying the amount of change through text remains difficult. Modifying the prompt may also affect other sound characteristics. One significant limitation is that CLAP embeddings may not adequately represent all acoustic and temporal changes relevant to control. Audio representations more sensitive to these changes may improve control learning. Furthermore, strong correspondence with the PM over the high-level audio features does not establish whether the model learns the intended sound variation or reproduces changes associated with artifacts introduced by the PM's design defects. Distinguishing meaningful control effects from such artifacts remains an important direction for improving the reliability of procedural control transfer.

5. CONCLUSION

We presented SliderSound, a method that combines meaningful controls from procedural audio models with the generation capabilities of a pretrained text-to-audio model. A control adapter learns relative acoustic changes from paired procedural sounds, enabling continuous parameter adjustments to text-generated outputs. Experiments on multiple sound effects demonstrate improved average control correspondence compared with Text-only and NeuralPM while largely retaining the original generation quality. This integration of procedural control and pretrained neural generation offers a promising approach to achieving both control and realism in sound effects design.

[5] https://audiocommons.github.io/

6. REFERENCES

[1] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley, "AudioLDM: Text-to-audio generation with latent diffusion models," in *Proceedings of the 40th International Conference on Machine Learning*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, Eds., 23–29 Jul 2023, vol. 202 of *Proceedings of Machine Learning Research*, pp. 21450–21474.

[2] Zach Evans, Julian D Parker, Matthew Rice, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons, "Stable audio 3," *arXiv preprint arXiv:2605.17991*, 2026.

[3] Gaëtan Hadjeres, Marc Ferras, Khaled Koutini, Benno Weck, Alexandre Bittar, Thomas Hummel, Zineb Lahrichi, Hakim Missoum, Joan Serrà, and Yuki Mitsufuji, "Woosh: A sound effects foundation model," *arXiv preprint arXiv:2604.01929*, 2026.

[4] Hugo Flores García, Oriol Nieto, Justin Salamon, Bryan Pardo, and Prem Seetharaman, "Sketch2sound: Controllable audio generation via time-varying signals and sonic imitations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

[5] Yoonjin Chung, Junwon Lee, and Juhan Nam, "T-foley: A controllable waveform-domain diffusion model for temporal-event-guided foley sound synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 6820–6824.

[6] Xinlei Niu, Jing Zhang, and Charles Patrick Martin, "Soundmorpher: Perceptually-uniform sound morphing with diffusion model," *arXiv preprint arXiv:2410.02144*, 2024.

[7] Purnima Kamath, Chitralekha Gupta, and Suranga Nanayakkara, "Morphfader: Enabling fine-grained controllable morphing with text-to-audio models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.

[8] Annie Chu, Hugo Flores-García, Oriol Nieto, Justin Salamon, Bryan Pardo, and Prem Seetharaman, "Mix2morph: Learning sound morphing from noisy mixes," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2026, pp. 15472–15476.

[9] Dimitris Menexopoulos, Pedro Pestana, and Joshua Reiss, "The state of the art in procedural audio," *Journal of the Audio Engineering Society*, , no. 12, pp. 826–848, 2023.

[10] Yisu Zong, Nelly Garcia-Sihuay, and Joshua Reiss, "A machine learning method to evaluate and improve sound effects synthesis model design," in *Audio Engineering Society Conference: AES International Audio for Games Conference*, 2024.

[11] Yisu Zong and Joshua Reiss, "Learning control of neural sound effects synthesis from physically inspired models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

[12] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang, "Clap: learning audio concepts from natural language supervision," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[13] Héctor Martel, Joe Hennessy-Priest, and Taemin Cho, "Probing low-level acoustic attribute encoding in clap audio embeddings," in *Proceedings of the 29th International Conference on Digital Audio Effects*, 2026.

[14] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[15] Prajit Ramachandran, Barret Zoph, and Quoc V Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[16] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet, "Directly fine-tuning diffusion models on differentiable rewards," in *International Conference on Learning Representations (ICLR)*, 2024.

[17] Andy Farnell, *Designing sound*, Mit Press, 2010.

[18] Karol J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. pp. 1015–1018, ACM Press.

[19] George K Sidiropoulos and George A Papakostas, "Machine biometrics-towards identifying machines in a smart city environment," in *2021 IEEE World AI IoT Congress (AIIoT)*, 2021, pp. 0197–0201.

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

[21] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.