# DIFFVOX: A DIFFERENTIABLE MODEL FOR CAPTURING AND ANALYSING VOCAL EFFECTS DISTRIBUTIONS

*Chin-Yun Yu*[♭*] , *Marco A. Martínez-Ramírez*[♯], *Junghyun Koo*[♯], *Ben Hayes*[♭], *Wei-Hsiang Liao*[♯], *György Fazekas*[♭], *and Yuki Mitsufuji*[♯♮]

[♭]Centre for Digital Music, Queen Mary University of London, London, UK
[♯]Sony AI, Tokyo, Japan
[♮]Sony Group Corporation, Tokyo, Japan
chin-yun.yu@qmul.ac.uk

## ABSTRACT

This study introduces a novel and interpretable model, DiffVox, for matching vocal effects in music production. DiffVox, short for "**Diff**erentiable **Vo**cal F**x**", integrates parametric equalisation, dynamic range control, delay, and reverb with efficient differentiable implementations to enable gradient-based optimisation for parameter estimation. Vocal presets are retrieved from two datasets, comprising 70 tracks from MedleyDB and 365 tracks from a private collection. Analysis of parameter correlations reveals strong relationships between effects and parameters, such as the high-pass and low-shelf filters often working together to shape the low end, and the delay time correlating with the intensity of the delayed signals. Principal component analysis reveals connections to McAdams' timbre dimensions, where the most crucial component modulates the perceived spaciousness while the secondary components influence spectral brightness. Statistical testing confirms the non-Gaussian nature of the parameter distribution, highlighting the complexity of the vocal effects space. These initial findings on the parameter distributions set the foundation for future research in vocal effects modelling and automatic mixing.

## 1. INTRODUCTION

Audio effects are essential in music production. They enable audio engineers to shape a sound's timbre and spatial characteristics, such as stereo width. Understanding how these effects (their settings) are used in real-world audio is valuable for developing automatic audio processing tools to create realistic music mixes. Yet, this knowledge is based on decades of experience and often remains untracked systematically. Since the distribution of effect parameters is unknown, we often approximate it with non-informative priors, such as uniform or Gaussian distributions, to ease downstream tasks. This is often used for generating synthetic training data for a classifier model that identifies the applied processing, including tasks such as effects detection [1], music mixing style transfer systems [2, 3, 4], or pretraining audio representations [5]. This forms a biased and weighted training objective, where the effect settings that are less likely to occur in real-world mixes are overrepresented, and vice versa, thus cancelling out the prior. The influence of weighting has been found in filter design using neural networks [6], where different sampling strategies of the filter coefficients affect the generalisation results on real-world impulse responses (IRs).

Unfortunately, due to the complexity of the music production process, it is challenging to collect real-world data with annotations on the effects parameters. Reverse-engineering the mix is more feasible. Directly optimising the effects parameters end-to-end using gradient descent is effective in equaliser matching [7], fitting IRs with filters [8] or feedback delay networks (FDN) [9], learning compressor parameters [10], or even capturing the whole mixing process graph [11]. We thus adopt differentiable sound matching as a proxy to capture real-world effects configurations. For simplicity and feasibility, we focus on a **single track, mono-in-stereo-out** scenario, independent of broader mix interactions between tracks. We choose vocals because they are often the most prominent element in a mix and are carefully processed. The resulting parameters can be considered as *presets* sampled from arbitrary mixes. The order and routing of the effects are fixed, resulting in fixed dimensionality, which makes later analysis tractable. We follow [11] in using static parameters to control the effects, which is sufficient to represent complete mixes.

Our contributions are as follows: Firstly, we propose an effects model that reflects professional music production practices while being efficient to train in a differentiable manner. We incorporate parallel algorithms running on graphics processing units (GPUs) to accelerate fitting the recursive filters used in the equaliser and dynamic range controller. We implement a differentiable ping-pong delay, an FDN reverb with frequency-dependent attenuation, and a dynamic range controller with look-ahead. Secondly, we propose a loss function that matches the signals' microdynamics in a multi-resolution fashion, similar to common spectral losses, to capture the features of interest in different scales. Thirdly, we fit the effects to hundreds of vocal tracks and analyse the collected presets. Our analysis reveals the importance of spatial effects for sound matching, highlights strong correlations of specific parameters, and demonstrates that the most explainable components of the parameter distributions are related to spaciousness and spectral brightness. Lastly, we publicise our experiments' source code and the vocal presets dataset to foster further research on audio effects prior [1].

## 2. THE EFFECTS MODEL

Our chosen effects are based on standard practices in music production [2]. The mono input is first treated with a six-band parametric equaliser, followed by a compressor and an expander as a dynamic range controller. Then, the signal is split into two paths: one for the dry signal and the other for the wet signal. A ping-pong delay and an FDN reverb process the wet signal. A panner processes

[*] Work done during an internship at Sony AI.

[1] github.com/SonyResearch/diffvox

[2] www.soundonsound.com/techniques/vocal-production

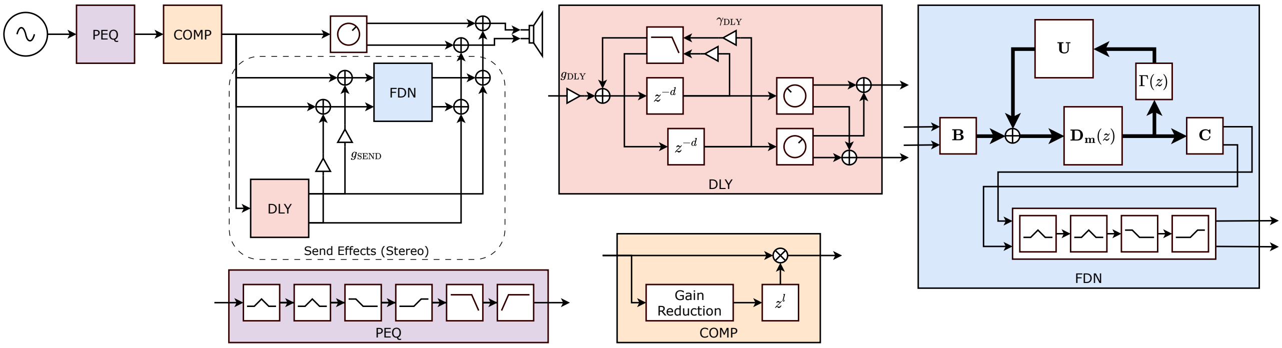

Figure 1: The proposed model (upper left) and individual effects for vocal effects processing.

the dry signal and then mixes it with the wet signal. The exact routes are shown in Fig. 1. We pick effect implementations with the fewest parameters possible to reduce the dimensionality so it does not exceed the number of vocal tracks we collected. We approximate the effects only when necessary to reduce fitting time. In the following sections, we describe each effect in detail.

### 2.1. The Parametric Equaliser (PEQ)

The PEQ sequentially applies the following six filters: two peak filters (PK1 and PK2), a low-shelf filter (LS), a high-shelf filter (HS), a low-pass filter (LP), and a high-pass filter (HP). We reference the T-RackS Classic Equaliser plugin by IK Multimedia[3] to set the ranges of the parameters. We follow the Audio EQ Cookbook[4] to implement the filters as Biquad filters since they are commonly used in digital audio effects. We fix the shelf filters' Q factor to 0.707, resulting in four gains, four Q factors, and six frequencies to be optimised.

Recently, differentiable time-domain evaluation of time-invariant recursive filters has been made possible by specialised kernels enabling efficient backpropagation [12]. However, these kernels do not employ any parallelisation along the time axis, meaning they do not fully utilise parallel processors. Blelloch's parallel prefix sum algorithm [13] illustrates that recursive expressions can be parallelised if the recursion can be expressed in terms of an associative operation. This reduces time complexity from $O(N)$ to $\approx O(\frac{N}{p})$, where $p$ is the number of parallel processors. Thus, we seek to express the Biquad filter recursion using an associative operation, allowing us to apply the parallel scan algorithm [14].

A Biquad filter consists of five parameters $\{b_0, b_1, b_2, a_1, a_2\}$. The state-space realisation [15] of a Biquad filter in Direct-Form-II is:

$$\begin{aligned} \tilde{\mathbf{x}}[n+1] &= \mathbf{A}_{\mathrm{BQ}}\tilde{\mathbf{x}}[n] + \begin{bmatrix} x[n] \\ 0 \end{bmatrix} \\ y[n] &= \mathbf{C}_{\mathrm{BQ}}\tilde{\mathbf{x}}[n] + b_0 x[n], \end{aligned} \tag{1}$$

where $\mathbf{A}_{\mathrm{BQ}} = \begin{bmatrix} -a_1 & -a_2 \\ 1 & 0 \end{bmatrix}$ and $\mathbf{C}_{\mathrm{BQ}} = \begin{bmatrix} b_1 - b_0 a_1 & b_2 - b_0 a_2 \end{bmatrix}$. Noting that the first line of Eq. (1) can be rewritten without recursion:

$$\tilde{\mathbf{x}}[n] = \sum_{k=1}^{n} \mathbf{A}_{\mathrm{BQ}}^{n-k} \begin{bmatrix} x[k-1] \\ 0 \end{bmatrix}, \tag{2}$$

we define an associative binary operation $\oplus$ acting on tuples $(\mathbf{U}_1, \mathbf{v}_1) \oplus (\mathbf{U}_2, \mathbf{v}_2) \mapsto (\mathbf{U}_2\mathbf{U}_1, \mathbf{U}_2\mathbf{v}_1 + \mathbf{v}_2)$. By setting $\mathbf{U}_n = \mathbf{A}_{\mathrm{BQ}}$, $\mathbf{v}_n = \begin{bmatrix} x[n-1] & 0 \end{bmatrix}^\top$, and $s_n = (\mathbf{U}_n, \mathbf{v}_n)$, we can express the recursion step of our filter associatively:

$$\tilde{s}_n = s_1 \oplus s_2 \oplus \cdots \oplus s_n = \bigoplus_{k=1}^{n} \left( \mathbf{A}_{\mathrm{BQ}}, \begin{bmatrix} x[k-1] \\ 0 \end{bmatrix} \right) \quad , \tag{3}$$

where $\tilde{s}_n^{(2)}$, the second entry of the tuple $\tilde{s}_n$, gives us $\tilde{\mathbf{x}}[n]$. Moreover, if $\mathbf{A}_{\mathrm{BQ}}$ is diagonalisable, applying $\oplus$ can be simplified further, reducing matrix multiplications to scalar multiplications. If the poles of the filter $\lambda_1, \lambda_2$ are distinct, $\mathbf{A}_{\mathrm{BQ}}$ can be diagonalised as $\mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}$ and $\mathbf{\Lambda} = \mathrm{diag}(\begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix})$. Using the fact that $\mathbf{A}_{\mathrm{BQ}}^n = \mathbf{P}\mathbf{\Lambda}^n\mathbf{P}^{-1}$ and altering our associative representation such that $\mathbf{v}_n = \mathbf{P}^{-1} \begin{bmatrix} x[n-1] & 0 \end{bmatrix}^\top$ and, accordingly, $\mathbf{U}_n = \mathbf{\Lambda}$, we recover our filtered signal $\tilde{\mathbf{x}}[n] = \mathbf{P}\tilde{s}_n^{(2)}$.

To ensure distinct poles, we restrict the Q factor of the HP and LP filters to be no smaller than 0.5. In the case of real poles, we set $\mathbf{P} = \begin{bmatrix} \begin{bmatrix} 1 & {\lambda_1}^{-1} \end{bmatrix}^\top & \begin{bmatrix} 1 & {\lambda_2}^{-1} \end{bmatrix}^\top \end{bmatrix}$. For complex conjugate poles ($\lambda_1 = \lambda_2^*$), we utilise the coupled form state-space model [16]. In the coupled form, the transition matrix is a rotation matrix. Applying the rotation matrix to a two-dimensional vector is equivalent to complex multiplication where the multiplier is the pole $\lambda_1$. In other words, we can run just one complex one-pole filter instead of two. The rotation and Direct-Form-II transition matrices are interchangeable by the following equations:

$$\mathbf{A}_{\mathrm{BQ}} = \mathbf{P} \begin{bmatrix} \Re(\lambda_1) & -\Im(\lambda_1) \\ \Im(\lambda_1) & \Re(\lambda_1) \end{bmatrix} \mathbf{P}^{-1}, \; \mathbf{P}^{-1} = \begin{bmatrix} 0 & \Im(\lambda_1) \\ -1 & \Re(\lambda_1) \end{bmatrix}. \tag{4}$$

Plug Eq. (4) into the first line of Eq. (1), multiply $\mathbf{P}^{-1}$ on both sides, and convert every vector into a complex number, we get the following recursion:

$$\bar{x}[n+1] = \lambda_1 \bar{x}[n] - i x[n]. \tag{5}$$

Let $\mathbf{v}_n = -ix[n-1]$ and $\mathbf{U}_n = \lambda_1$ in the associative operation, then the filtered signal $\tilde{\mathbf{x}}[n]$ is $\mathbf{P} \begin{bmatrix} \Re(\tilde{s}_n^{(2)}) & \Im(\tilde{s}_n^{(2)}) \end{bmatrix}^\top$. The computational cost is the same as in Eq. (1), but the implementation is more straightforward since we can treat it as a one-pole filter.

### 2.2. The Feed-Forward Compressor and Expander (COMP)

We adopt the compressor and expander model from the DAFx textbook [17]. The effect is controlled by the following parame-

[3] www.ikmultimedia.com/products/trclasseq

[4] https://www.w3.org/TR/audio-eq-cookbook/

ters: the compressor/expander thresholds $CT/ET$, the compressor/expander ratios $CR/ER$, the attack/release/RMS smoothing factors $\alpha_{\text{at}}/\alpha_{\text{rt}}/\alpha_{\text{rms}}$, and the make-up gain. We use the differentiable implementation `torchcomp` by Yu et al. [18]. The RMS level detector and the backpropagation of the attack/release ballistic filter are also one-pole filters; thus, we accelerate them using the parallel scan.

Due to the causal smoothing effect of the RMS level detector, the gain reduction signal $g[n]$ is slightly delayed. Modern digital compressors often have a look-ahead feature to compensate for this delay. To learn the continuous delay time $l \in \mathbb{R}_+$, we approximate the look-ahead by truncated sinc interpolation:

$$g(n+l) \approx \sum_{k=-L_1}^{L_2+1} g[n+k]\operatorname{sinc}(k-l), \tag{6}$$

where $l \in [0, L_2]$ and $L_1, L_2$ are the truncation lengths. The sinc function is defined as $x \mapsto \frac{\sin(\pi x)}{\pi x}$. The output of the compressor is then $y_{\text{COMP}}[n] = g(n+l)x[n]$.

### 2.3. The Ping-Pong Delay (DLY: $\mathbb{R} \to \mathbb{R}^2$)

Ping-Pong delay is a stereo delay effect where the delays alternate between the left and right channels. It is implemented by two delay lines whose outputs are each other's inputs. In modern music production, the panning of the delay is more flexible and not limited to hard left and right. We thus add two separate panners to control the panning of two delay lines. We add an LP filter in the feedback path to simulate the decaying echoes. The LP filter is the same Biquad filter as in Section 2.1.

We adopt the damped sinusoidal approach to learn the delay time using frequency-sampling (FS) [19, 20], representing the delay effects as convolutions with a truncated finite impulse response (FIR) $\mathbf{h}_{\text{DLY}}[n]$ with a length of $N_{\text{DLY}}$. The following equations approximate the transfer functions of the delays:

$$\begin{aligned} H_{\text{odd}}(z) &= \frac{z^{-d}}{1-\gamma_{\text{DLY}}H_{\text{LP}}(z)z^{-2d}} \\ &\approx \gamma_{\text{DLY}}^{-1} \sum_{k=1}^{\left\lfloor \frac{N_{\text{DLY}}-d}{2d} \right\rfloor} \left(\gamma_{\text{DLY}}H_{\text{LP}}(z)\eta^{\frac{\angle z}{2\pi}N_{\text{DLY}}}z^{-d}\right)^{2k-1}, \end{aligned} \tag{7}$$

$$\begin{aligned} H_{\text{even}}(z) &= \frac{\gamma_{\text{DLY}}H_{\text{LP}}(z)z^{-2d}}{1-\gamma_{\text{DLY}}H_{\text{LP}}(z)z^{-2d}} \\ &\approx \sum_{k=1}^{\left\lfloor \frac{N_{\text{DLY}}}{2d} \right\rfloor - 1} \left(\gamma_{\text{DLY}}H_{\text{LP}}(z)\eta^{\frac{\angle z}{2\pi}N_{\text{DLY}}}z^{-d}\right)^{2k}, \end{aligned} \tag{8}$$

where $\eta \in [0,1]$ is the surrogate variable, $\gamma_{\text{DLY}} \in [0,1]$ is the decay factor, $d$ is the delay time, and $H_{\text{LP}}(z)$ is the transfer function of the LP filter. $\eta^{\frac{\angle z}{2\pi}N_{\text{DLY}}}z^{-d}$ forms a damped sinusoidal in the frequency domain, a surrogate way to learn the true delay operator $z^{-d}$. We set $\eta = 1$ during inference, assuming the variable always converges to one. We explicitly compute each delayed impulse within the range $0 \le n < N_{\text{DLY}}$ to reduce the time aliasing effect when FS the original transfer function [21].

The two impulses are then joined together as

$$\mathbf{h}_{\text{DLY}}[n] = \text{PAN}_{\text{odd}}(h_{\text{odd}}[n]) + \text{PAN}_{\text{even}}(h_{\text{even}}[n]) \tag{9}$$

where PAN : $\mathbb{R} \to \mathbb{R}^2$ is the panning function. Following [20], we use the straight-through estimator to backpropagate the gradients to the damped sinusoidal. The final output of the ping-pong delay is the convolution of the input signal with the IR $\mathbf{y}_{\text{DLY}}[n] = g_{\text{DLY}}(y_{\text{COMP}}[n] * \mathbf{h}_{\text{DLY}}[n])$ and $g_{\text{DLY}} \in [0,1]$ controls the volume of the delayed signal.

### 2.4. The Feedback Delay Network Reverb (FDN: $\mathbb{R}^2 \to \mathbb{R}^2$)

FDN is an artificial reverberation algorithm that uses a network of delay lines with feedback to create dense reverberations [22]. In this work, we use a stereo FDN with six delay lines. It is best described in state-space form as:

$$\begin{aligned} \begin{bmatrix} \tilde{x}_1[n+m_1] \\ \tilde{x}_2[n+m_2] \\ \vdots \\ \tilde{x}_6[n+m_6] \end{bmatrix} &= \mathbf{A}\tilde{\mathbf{x}}[n] + \mathbf{B}\mathbf{x}[n], \\ \mathbf{y}_{\text{FDN}}[n] &= \mathbf{C}\tilde{\mathbf{x}}[n], \end{aligned} \tag{10}$$

where $\mathbf{A} \in \mathbb{R}^{6\times6}$, $\mathbf{B} \in \mathbb{R}^{6\times2}$, $\mathbf{C} \in \mathbb{R}^{2\times6}$, and $m_i$ are the delay times. $\mathbf{A}$ controls how the energies spread across delay lines. The delay times are co-primes to increase echo density. We set the delay times to $\mathbf{m}$ = [997, 1153, 1327, 1559, 1801, 2099] proposed in [23].

Computing the recursion of Eq. (10) directly in automatic differentiation frameworks is time-consuming due to the overhead of an enormous amount of function calls to register computational nodes, and each contributes little computation [12]. Furthermore, there are no specialised kernels for Eq. (10) (in contrast to the PEQ in Section 2.1); thus, we use the FS method to approximate the IR of the FDN. The transfer function $\mathbf{H}_{\text{FDN}}(z) = \frac{\mathbf{Y}_{\text{FDN}}(z)}{\mathbf{X}(z)}$ is given by the following equation:

$$\mathbf{H}_{\text{FDN}}(z) = \mathbf{C}\left(\mathbf{D}_{\mathbf{m}}^{-1}(z) - \mathbf{A}(z)\right)^{-1}\mathbf{B}, \tag{11}$$

$$\mathbf{D}_{\mathbf{m}}(z) = \operatorname{diag}\left(\begin{bmatrix} z^{-m_1} & z^{-m_2} & \dots & z^{-m_6} \end{bmatrix}\right). \tag{12}$$

We parametrise $\mathbf{A}(z) = \mathbf{U}\Gamma(z)$ where $\mathbf{U}$ is an orthogonal matrix and $\Gamma(z) = \operatorname{diag}([\gamma^{m_1}(z)\dots\gamma^{m_6}(z)]), \gamma(z) \in [0,1]$. We follow Mezza et al. [9] to parametrise $\mathbf{U}$ to be unitary, thus satisfying the unilosslessness condition [24]. $\gamma(z)$ is the attenuation filter that controls the decay rate of the reverb.

In most of the previous works on differentiable FDN [9, 23, 25], frequency-independent $\gamma$ is used to parametrise $\mathbf{A}$, which limits the flexibility of the model. The difficulty is that we usually want the decay time to be delay-independent, and designing such filters is non-trivial. Mezza et al. [26] tackle this problem indirectly by learning separate FIRs for each delay line and training them with a frequency-dependent objective. Here, we adopt a more straightforward approach that aligns with the FS method. We sample 49 points [5] of $\gamma(z)$ with equal spacing from 0 to $\pi$. The attenuation coefficients are then upsampled to the desired length of the FFT during FS. After fitting, we can approximate the FDN by calculating the linear-phase filter from the magnitude response of $\gamma^{m_i}(z)$ and then applying Mezza's model for real-time purposes. In addition, the decaying time of the reverb in Eq. (11) is frequency-dependent, while the initial gain of the reverb is not. To correct this, we add a PEQ after the reverb, which contains two peak filters and two shelf filters from Section 2.1. In practice, we apply PEQ on the impulse response of the reverb $\mathbf{h}_{\text{FDN}}[n]$ before convolving it with the input signal for efficiency.

[5] The number was decided empirically based on early experiments by reducing the number of points until it loses spectrum details.

### 2.5. The Effect Sends (SEND) and Parametrisation

Both the reverb (Section 2.4) and the delay (Section 2.3) only model the wet signal. To further increase the model's flexibility, we also send the delayed signal to the reverb to colourise the delays to have similar acoustic characteristics to the direct signal, controlled by the send level $g_{\text{SEND}} \in [0, 1]$. The total number of parameters in our effect chain is 152[6]. Compared to GRAFx [11, 20], a package that provides differentiable effects modelling, a similar effects signal chain requires at least 264 more parameters [7], since in GRAFx the goal is to represent the mix faithfully, thus their delay and reverbs are over-parametrised to be expressive enough for various mixing materials. In contrast, our model specialises in vocals, and we prioritise having a compact representation of parameters for analysis over expressiveness. Since many effects' parameter ranges are bounded, we apply different parametrisation to the parameters, which are summarised in Table 1.

## 3. EXPERIMENTS

### 3.1. Datasets

We apply our effects model to two datasets: 1) the `MedleyDB` [27, 28] and 2) our private multi-track dataset `Internal` with paired dry and wet stems [11]. The latter consists mainly of modern mainstream Western music. Both datasets are sampled at 44.1 kHz. We use the official metadata of `MedleyDB` to pick the vocal tracks. For `Internal`, the pairing information between raw tracks and processed stems is missing. We calculate the cross-correlations between each song's dry tracks and wet stems and use these correlations to recover their mapping. We then drop non-vocal stems based on their filenames.

Stems processed from only one raw track are selected to fit our problem setting (*mono-in-stereo-out*). Some input tracks are stereo, possibly due to the exporting process from the DAW. For these tracks, we first peak-normalise both channels and then calculate their difference (side channel). We drop the track if the maximum side energy exceeds $-10\,\text{dB}$. We then take the average of the two channels to form a mono source. Since the raw tracks and processed stems are not always aligned in time, we time-align the raw tracks so their cross-correlation with the processed stems is maximised.

### 3.2. Optimisation

The loss functions we use are 1) the multi-scale STFT (MSS) loss, 2) the multi-scale Loudness Dynamic Range (MLDR) loss, and 3) the regularisation loss on the surrogate variable $\eta$. The MSS loss is defined as

$$\mathcal{L}_{\text{MSS}}(\hat{y}[n], y[n]) = \frac{1}{3} \sum_{N \in \{128, 512, 2048\}} \frac{\|\hat{\mathbf{Y}}_N - \mathbf{Y}_N\|_2}{\|\mathbf{Y}_N\|_2} + \frac{\|\log(\hat{\mathbf{Y}}_N) - \log(\mathbf{Y}_N)\|_1}{N M_N}, \quad (13)$$

where $\hat{\mathbf{Y}}_N$ and $\mathbf{Y}_N$ are the magnitude spectrograms of the predicted and ground-truth signals, respectively, computed with FFT size $N$ and hop size $\frac{N}{4}$. $M_N$ is the number of frames in the spectrogram. Similar to [11], we use `auraloss`[8] to compute the MSS loss, with A-weighting applied before the STFT [29]. This loss minimises the distance in the spectral domain.

Inspired by previous work on differentiable microdynamics metrics [30], we propose the MLDR loss to match the dynamics of the predicted and ground-truth signals, thus better guiding the fitting of the compressor. Given a signal $x[n]$, its LDR is defined as

$$\begin{aligned}&\text{LDR}(x[n], t_{\text{short}}, t_{\text{long}}) \\ &\quad = \log\left(\frac{\text{RMS}\left(x^2[n], t_{\text{short}}\right)}{\text{RMS}\left(x^2[n + \lfloor\frac{t_{\text{long}} - t_{\text{short}}}{2T_s}\rfloor], t_{\text{long}}\right)}\right), \quad (14)\end{aligned}$$

where RMS calculates the energy envelopes of the signal, $t_{\text{short}}$ and $t_{\text{long}}$ are the integration times in second, and $T_s$ is the sampling period. The longer the integration time, the smoother the RMS envelope. For more calculation details, please refer to [30]. $t_{\text{long}} \gg t_{\text{short}}$, so the LDR describes how the loudness varies locally in a scale proportional to $t_{\text{long}}^{-1}$. Similar to MSS loss, we want to match the LDR from coarse to fine scales. Thus, we propose the following $L_1$ loss:

$$\begin{aligned}&\mathcal{L}_{\text{MLDR}}(\hat{y}[n], y[n]) \\ &= \frac{1}{N} \sum_{t \in \{1,2\}} \sum_{n=0}^{N-1} \left| \text{LDR}\left(\hat{y}[n], \frac{t}{20}, t\right) - \text{LDR}\left(y[n], \frac{t}{20}, t\right) \right| \\ &\qquad (15)\end{aligned}$$

where $N$ is the length of the signal.

The final loss function is the weighted sum of the MSS and MLDR losses on the left, right, mid, and side channels, plus the regularisation loss on $\eta$:

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{y}}[n], \mathbf{y}[n]) &= \frac{1}{2} \sum_{i=1}^{2} \Big[\mathcal{L}_{\text{MSS}}(\hat{y}_i[n], y_i[n]) \\ &+ \frac{1}{2}\mathcal{L}_{\text{MSS}}(\sqrt{2}\mathcal{H}(\hat{\mathbf{y}}[n])_i, \sqrt{2}\mathcal{H}(\mathbf{y}[n])_i) + \frac{1}{2}\mathcal{L}_{\text{MLDR}}(\hat{y}_i[n], y_i[n]) \\ &+ \frac{1}{4}\mathcal{L}_{\text{MLDR}}(\mathcal{H}(\hat{\mathbf{y}}[n])_i, \mathcal{H}(\mathbf{y}[n])_i)\Big] + (1 - \eta)^2, \quad (16)\end{aligned}$$

where $\mathcal{H} : \begin{bmatrix} x & y \end{bmatrix}^\top \mapsto \frac{1}{\sqrt{2}} \begin{bmatrix} x + y & x - y \end{bmatrix}^\top$ is the Hadamard transform that converts left/right to mid/side channels. The last term encourages the damped sinusoidal to be on the unit circle. The weights are set empirically to balance the initial magnitude of the individual losses.

### 3.3. Fitting Details

We normalise the input and target vocals to have $-18\,\text{dB}$ LUFS [31] using `pyloudnorm`[9]. Each track is then split into twelve-second segments with five seconds of overlap. The overlapped region is used as a warm-up, and the loss is calculated only for the last seven seconds, similar to [11]. We drop silent segments and select up to 35 segments in each training step to form a batch. We train the effects on each track for 2k steps using the Adam optimiser with 0.01 learning rate and pick the checkpoint with the lowest loss. We use the CUDA implementation of parallel scan by Martin et al. [32]. The fitting time of each track ranges from 20 to 40 minutes on a single RTX 3090 GPU.

[6] 14 for PEQ, 9 for the compressor and expander, 8 for the delay, 119 for the FDN reverb, one for the panning, and one for the send level.

[7] This number is based on replacing our delay and reverbs with theirs.

[8] github.com/csteinmetz1/auraloss

[9] github.com/csteinmetz1/pyloudnorm

Table 1: The parametrisation of the effects. $\text{tri}(\mathbf{X})$ is the upper triangular part of the matrix $\mathbf{X}$. For details on the bounds, please refer to our code repository.

| Condition ($\mathbb{P}$) | Parametrisation ($\mathbb{R} \to \mathbb{P}$) | Parameters ($\theta \in \mathbb{R}$) |
|---|---|---|
| $x \in \mathbb{R}$ | $\theta \mapsto \theta$ | Equaliser's/make-up gain, $CT, ET, \mathbf{B}, \mathbf{C}$ |
| $0 \leq x \leq 1$ | $\sigma : \theta \mapsto \frac{1}{1+e^{-\theta}}$ | Panning, $\alpha_{\text{rms}}, \alpha_{\text{at}}, \alpha_{\text{rt}}, ER, \gamma_{\text{DLY}}, g_{\text{DLY}}, g_{\text{SEND}}$ |
| $a \leq x \leq b$ | $\theta \mapsto a + \sigma(\theta)(b-a)$ | Equaliser's Q and frequency, $CR, d, \gamma(z)$ |
| $0 \leq x \leq a$ | $\theta \mapsto \lvert\theta\rvert \mod a$ | $l$ |
| $x \leq 0$ | $\theta \mapsto -\log(1+e^{\theta})$ | $\log(\eta)$ |
| $\mathbf{X}^\top\mathbf{X} = \mathbf{I}$ | $\boldsymbol{\Theta} \mapsto e^{\text{tri}(\boldsymbol{\Theta})-\text{tri}(\boldsymbol{\Theta})^\top}$ | $\mathbf{U}$ |

The PK and LS/HS filters in the PEQ are initialised with zero gains. The PK filters' cut-off frequencies are bounded differently, so their parameters are ordered and not permutation-invariant. The cut-off frequencies for LP and HP filters are initialised to 17.5 kHz and 200 Hz, respectively. The dynamic range controls are initialised with $CR = 2, ER = \frac{1}{2}, CT = -18, ET = -48$, and make-up gain 0 dB. We initialise the delay with $\gamma_{\text{DLY}} = g_{\text{DLY}} = 0.1$ and delay time to 400 ms. We initialise the FDN with $\mathbf{B} = \mathbf{1}, \mathbf{C} = \mathbf{0}, \gamma(z) \sim \mathcal{U}(0.4, 0.6)$. The send level is initialised to 0.01. The model is initialised very close to an identity function, which is the upper bound for the loss, so we can easily detect problematic runs. We set the impulse response length to four seconds for the delay and 12 seconds for the FDN reverb. We bound the damping factor $\gamma(z)$ to have a maximum of nine seconds T60 time to reduce the aliasing effect, but still be long enough to capture most of the reverb tail.

A few tracks have non-linear effects, such as distortion and modulations, that are not modelled in our effects. To exclude them, we drop fitting runs that 1) have a minimum loss above a certain threshold, 2) have a loss that fluctuates heavily during fitting, or 3) have a loss that is not decreasing. The thresholds are set empirically after checking runs with apparent outlier fitting losses. Out of 76 tracks from `MedleyDB`, 6 tracks are excluded ($\approx$ 8%); out of 370 tracks from `Internal`, 5 tracks are excluded ($\approx$ 1.3%).

## 4. RESULTS AND DISCUSSIONS

### 4.1. Sound Matching Performance

We fit different configurations to test the benefits of having spatial effects. We denote the complete model as DiffVox. The evaluation metrics are the fitting losses on the left/right and mid/side channels. The averaged scores across tracks are shown in Table 2.

Table 2: Matching performance with different configurations.

| **Dataset** | **Configuration** | **MSS** ↓ | | **MLDR** ↓ | |
|---|---|---|---|---|---|
| | | l/r | m/s | l/r | m/s |
| `Internal` | No processing | 1.44 | 2.39 | 1.82 | 2.08 |
| | DiffVox | **0.76** | **0.94** | **0.34** | **0.41** |
| | ⊢ w/o Approx. | 0.78 | 0.95 | 0.38 | 0.44 |
| `MedleyDB` | No processing | 1.27 | 2.16 | 1.00 | 1.35 |
| | DiffVox | 0.75 | 0.98 | **0.39** | **0.45** |
| | ⊢ w/o Approx. | 0.77 | 1.00 | 0.42 | 0.48 |
| | w/o FDN | 0.79 | 1.14 | 0.48 | 0.62 |
| | w/o DLY | 0.76 | 0.99 | 0.40 | 0.47 |
| | w/o DLY&FDN | **0.61** | **0.90** | 0.82 | 1.17 |

From Table 2, we can see that with just PEQ and compressor, the rendered audio matches well regarding spectral content, as indicated by the MSS on `MedleyDB`. However, it fails to match the microdynamics indicated by the MLDR loss. Adding the delay or the FDN improves the matching of the microdynamics. DiffVox achieves the best matching performance in MLDR and a lower MSS than with just Delay or FDN, proving its ability to match the sound in spectral and dynamic aspects. After removing the approximation, the performance drops slightly, which is expected since we swap truncated FIRs of reverb and delay with infinite IRs during inference, allowing the longer tails in the IRs to introduce a mismatch. Nevertheless, it is still better than without FDN.

### 4.2. Parameter Correlation Analysis

We analyse the correlation between the parameters in both datasets. Specifically, on the minimum set of parameters to reproduce the effects $\boldsymbol{\theta} \in \mathbb{R}^{130}$, which excludes the surrogate variable $\eta$ and the lower triangular part of the logits of $\mathbf{U}$. Since the parameters are unlikely to be normally distributed, we compute their Spearman correlation coefficient (SCC) instead of regular correlations[10]. We exclude the correlations of multi-dimensional parameters $\gamma(z)$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{U}$ since interpreting them is not straightforward. We observe high correlations between the sampled $\gamma(z)$, suggesting that the attenuation coefficients can be decomposed into fewer parameters.

The following discussions are based on the top ten most correlated pairs of parameters. `MedleyDB`'s most correlated pairs are not aligned with `Internal`, showing that the two datasets have different characteristics. Since `MedleyDB` has less data compared to `Internal` (70 vs. 365 tracks), the correlation we see is also less reliable. Thus, we primarily focus on the `Internal` dataset.

High correlations are observed between the delay effect parameters. The negative correlations between the delay time and the feedback gain (-.58) and the delay gain (-.51) imply that when a longer delay time is used, the delay effects diminish. The positive correlation between the feedback gain and the cut-off frequency of the delay's LP filter (.49) indicates that when a darker delay is used, it also fades out faster, and vice versa. In addition, high correlations are observed between the gain and the Q factor of the PEQ's PK2 filter (-.60) and the gain and cut-off frequency of the FDN's PK2 filter (-.46). The counter-interaction between the compressor threshold and the make-up gain is also observed (-.55), which is expected, as the two parameters are usually adjusted together to ensure consistency in loudness before and after the compressor.

[10] For convenience, the analysis is performed on the parameter logits before the parametrisation. This does not affect SCC for most of the scalar parameters (excludes $\mathbf{U}$ and $l$) since their parametrisation in Table 1 is monotonic.

We see that the attenuation coefficients above 19.7 kHz are highly correlated (ranging from .53 to .60) with the LP filter cut-off frequency. The reverb tends to compensate for the high-frequency loss by reducing the decay rate because we set the maximum cut-off frequency to 18 kHz according to the T-RackS EQ. A possible solution is to set a higher bound for the cut-off frequency or incorporate a wet/dry mix ratio on LP similar to GRAFx [20].

To analyse the correlation from a broader perspective, we measure the correlations between individual effects in the effects model on `Internal`. We define the effect-wise correlation as the average of the absolute SCCs between the parameters of the two effects. For correlations within the same effect, the diagonal elements are excluded. We see that most of the effects have high autocorrelation, except for the PEQ's LS filter (.011). The LS filter has a high correlation with the HP filter (.198). The LS filter only has two parameters: gain and frequency. A low correlation means that the two parameters operate nearly independently. This implies the two filters are used collaboratively to shape the low end. Based on the correlations, the hierarchical clustering using Ward's method [33] reveals three main clusters: all the spatial effects, the HP and LS filters, and the remaining effects. These clusters could be used to inform a simpler design of vocal effects.

### 4.3. Principal Component Analysis

Inspired by a similar dataset work [34], we perform principal component analysis (PCA) [35] on the parameters' logits to analyse the distribution of the parameters. Although PCA is limited to linear relationships, its simplicity and interpretability make it a good starting point for understanding the effects prior. Given the sample covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\theta}} \in \mathbb{R}^{130\times130}$ of the logits $\boldsymbol{\theta}$, we compute the eigenvalues $\boldsymbol{\lambda}_{\boldsymbol{\theta}} \in \mathbb{R}^r$ and the eigenvectors $\mathbf{V}_{\boldsymbol{\theta}} \in \mathbb{R}^{130\times r}$ such that

$$\begin{aligned}\mathbf{\Sigma}_{\boldsymbol{\theta}} = \mathbf{V}_{\boldsymbol{\theta}}\text{diag}(\begin{bmatrix}\lambda_{\boldsymbol{\theta}_1} & \lambda_{\boldsymbol{\theta}_2} & \ldots & \lambda_{\boldsymbol{\theta}_r}\end{bmatrix})\mathbf{V}_{\boldsymbol{\theta}}^\top, \\ \lambda_{\boldsymbol{\theta}_1} \geq \lambda_{\boldsymbol{\theta}_2} \geq \ldots \geq \lambda_{\boldsymbol{\theta}_r} > 0.\end{aligned} \tag{17}$$

$\mathbf{V}_{\boldsymbol{\theta}}$ is an orthogonal matrix, and the eigenvectors are the principal components (PCs) of the parameters. The eigenvalues represent the variance of the parameters in the direction of the PCs. $r$ is the rank of $\mathbf{\Sigma}_{\boldsymbol{\theta}}$ and equals the number of non-zero eigenvalues.

The cumulative percentage of total variance (CPV, [35]) is a simple method for evaluating the PCA model's capacity. The CPV given $p$ number of components to retain is defined as $100 \sum_{i=1}^{p} \lambda_{\boldsymbol{\theta}_i}$ / $\sum_{i=1}^{r} \lambda_{\boldsymbol{\theta}_i}$. The CPV of both models is shown in Fig. 2. `Internal` has more variance explained by the first $p$ components than `MedleyDB`, indicated by the larger area under the curve (AUC = 91.0 % vs 88.5 %). In other words, the parameters in `Internal` are more densely distributed than `MedleyDB`. To see how `MedleyDB` is explained by the `Internal`'s PCA model, we also compute another CPV by replacing the eigenvalues with the sum of squared projections of `MedleyDB`'s parameters onto each `Internal`'s PC[11] and plot it in Fig. 2. It shows that ≈ 65 % of the variances in `MedleyDB` can be captured by `Internal`'s first ten PCs, and the CPV trends are similar, but for higher PCs, the discrepancy increases.

One application of having a PCA model is using it as a generative model, drawing samples from the distribution $\mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\lambda}_{\boldsymbol{\theta}}))$. To test this assumption, we perform two multivariate normality tests, Royston's [36] and Henze-Zirkler's [37] tests, on the PC weights of `Internal`. The p-values of the tests are nearly zero given the first 75 PCs (≈ 99 % of the total variance), indicating that the logits are unlikely to be drawn from a multivariate normal distribution.

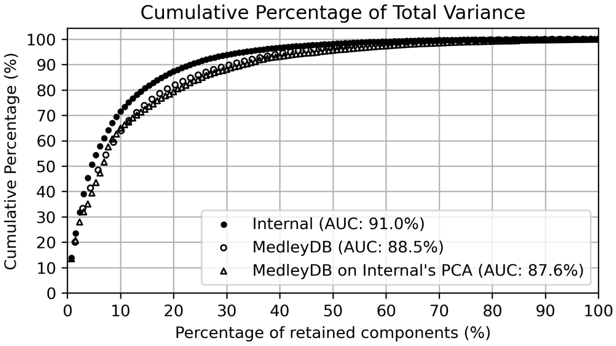

Figure 2: Cumulative total variance as a function of the percentage of retained PCs ($100p/r$) from both PCA models.

We plot the frequency responses of the PEQ, the delay, the tone correction PEQ, and the reverb decay time in Fig. 3. The mean parameters (first column) show a reasonable configuration that we could expect for a professionally processed vocal. Both the high and low ends are boosted, and the mid-range close to 1 kHz is slightly attenuated with narrow Q. An adequate amount of delay and reverb is added, with the high frequencies in the reverb attenuated. The reverb's decay time is around 2 seconds, starting at low frequencies and gradually decreasing to zero. The compression ratio is set to 3.5:1, with a threshold of $-22$ dB and a make-up gain of 2.4 dB. The vocals are polished and professionally processed.

To see how the PCs affect the parameters, we add the $i^{\text{th}}$ PC to $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ with scales in $\{3, 1, -1, -3\} \times \sqrt{\lambda_{\boldsymbol{\theta}_i}}$. The first PC (second column) mainly affects the spaciousness of the acoustic property where the vocals are placed. The feedback gain and overall volume of the delay are increased. The decay time of the reverb is also significantly increased, especially in frequencies above 4 kHz, which creates a very shimmery and spacious sound. The second PC (third column) creates a band-pass effect similar to that of a telephone, which can be seen from the drastic changes in the HS and LP filters. This aligns with McAdams' timbre space [38], where their second dimension is related to the spectral centroid. Long reverberations also smooth the attack time, which is related to McAdams' first dimension. The dynamic range compression gets slightly heavier in the fourth PC but does not change much in the first three.

## 5. CONCLUSIONS

This paper presents an expressive differentiable model for vocal effects processing and a method for capturing the effects parameters from professional mixes. Spatial effects are crucial for achieving good matching performance, as indicated by lower microdynamics losses. The parameter correlation analysis reveals meaningful relationships between effect parameters, such as the interaction between delay time and feedback gain and between compressor threshold and make-up gain. The first two principal components of the PCA model on our private dataset reveal primary directions that alter the vocals, including control of spaciousness and frequency bandwidth manipulation (telephone-like effects), which have some connections to McAdams' timbre space.

[11] The PCA projection is defined as $\mathbf{x} \mapsto \mathbf{V}_{\boldsymbol{\theta}}^\top(\mathbf{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}})$ where $\mu_{\boldsymbol{\theta}}$ is the sample mean of the parameters.

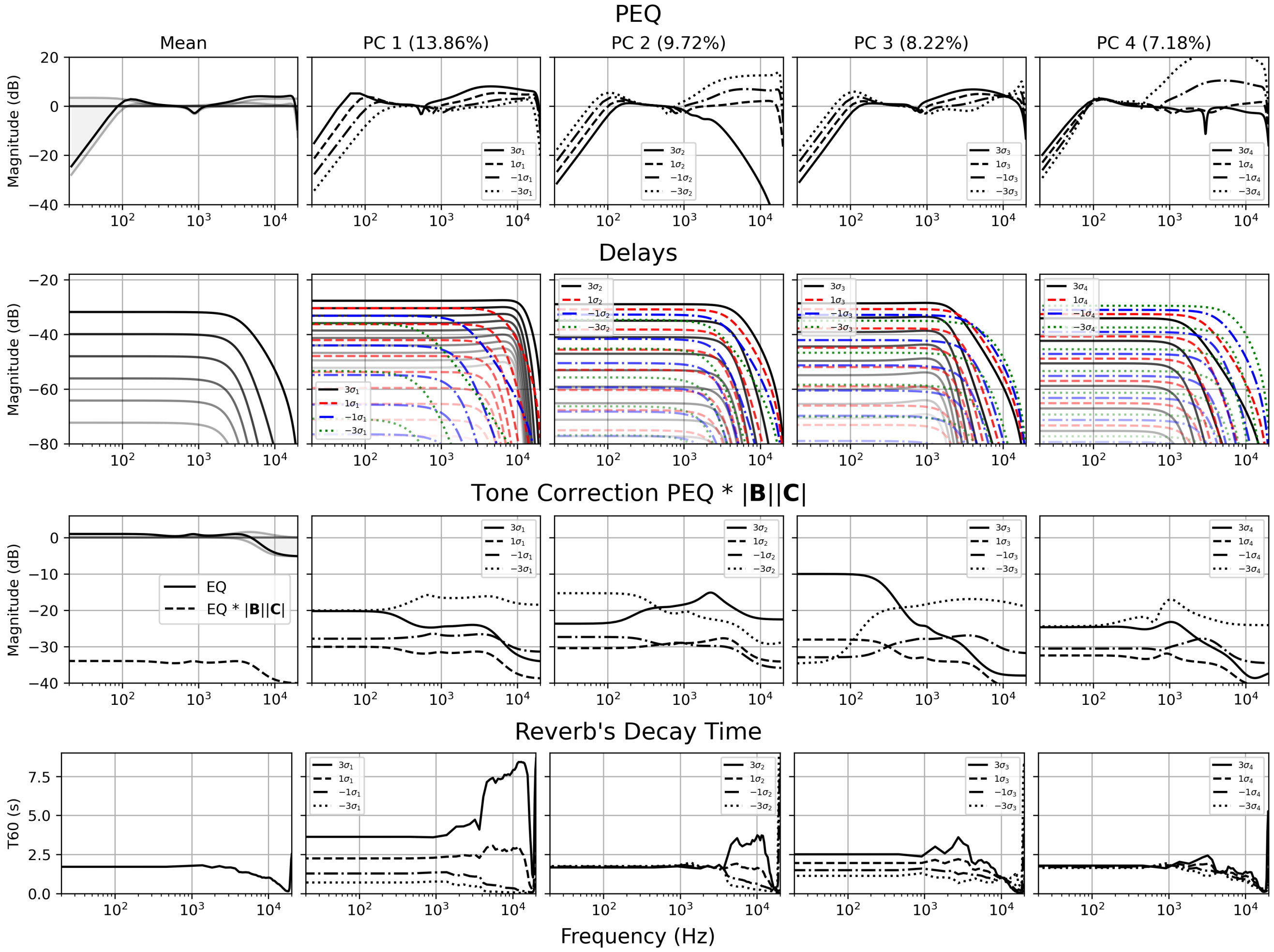

Figure 3: The mean (column one) and the first four principal components (column two to five with their percentage of explained variance) of the `Internal` dataset. The first and third rows show the frequency response of the PEQ and the tone correction filter. The second row shows the frequency response of the delayed signals, with colour intensity proportional to the delay time. The fourth row shows the frequency-dependent decay of the FDN reverb.

While our approach successfully captures effect parameters for most tracks, limitations remain in handling vocals with other effects we did not model or heavy automation. Addressing these limits and extending the model to multi-track scenarios is left for future work. The non-normality of the parameter distribution suggests that a more sophisticated generative model is needed to capture the actual distribution. We release the dataset of 435 vocal preset parameter logits produced in this study, plus a test set of 20 additional presets from the `Internal` dataset. The dataset is accompanied by the DiffVox model implemented in PyTorch, scripts to reproduce the results on the `MedleyDB` dataset, and the PCA models. We hope these resources can advance the development of future automatic mixing tools or neural audio effect models.

## 6. ACKNOWLEDGMENTS

We thank Sungho Lee for providing the cross-correlation data of the `Internal` dataset. This research is supported jointly by UKRI (grant number EP/S022694/1) and QMUL and utilised Queen Mary's Apocrita HPC facility, supported by QMUL Research-IT. doi: 10.5281/zenodo.438045

## 7. REFERENCES

[1] Matthew Rice, Christian J Steinmetz, George Fazekas, and Joshua D Reiss, "General purpose audio effect removal," in *IEEE WASPAA*, 2023, pp. 1–5.

[2] Christian J. Steinmetz, Nicholas J. Bryan, and Joshua D. Reiss, "Style transfer of audio effects with differentiable signal processing," *JAES*, vol. 70, no. 9, pp. 708–721, 2022.

[3] J. Koo, M. A. Martínez-Ramírez, W.-H. Liao, S. Uhlich, K. Lee, and Y. Mitsufuji, "Music mixing style transfer: A contrastive learning approach to disentangle audio effects," in *IEEE ICASSP*, 2023, pp. 1–5.

[4] J. Koo, M. A. Martinez-Ramirez, W.-H. Liao, G. Fabbro, M. Mancusi, and Y. Mitsufuji, "ITO-Master: Inference-time

optimization for audio effects modeling of music mastering processors," in *ISMIR*, 2025.

[5] C. J. Steinmetz, S. Singh, M. Comunità, I. Ibnyahya, S. Yuan, E. Benetos, and J. D. Reiss, "ST-ITO: Controlling audio effects for style transfer with inference-time optimization," in *ISMIR*, Nov. 2024, pp. 661–668.

[6] Joseph T Colonel, Christian J Steinmetz, Marcus Michelen, and Joshua D Reiss, "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *IEEE ICASSP*, 2022, pp. 3104–3108.

[7] Shahan Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *DAFx*, 2020, pp. 265–272.

[8] P. Bhattacharya, P. Nowak, and U. Zölzer, "Optimization of cascaded parametric peak and shelving filters with backpropagation algorithm," in *DAFx*, 2020, pp. 101–108.

[9] A. I. Mezza, R. Giampiccolo, E. De Sena, and A. Bernardini, "Data-driven room acoustic modeling via differentiable feedback delay networks with learnable delay lines," *EURASIP JASM*, vol. 2024, no. 1, pp. 51, 2024.

[10] Alec Wright and Vesa Valimaki, "Grey-box modelling of dynamic range compression," in *DAFx*, 2022, pp. 304–311.

[11] S. Lee, M. A. Martínez-Ramírez, W.-H. Liao, S. Uhlich, G. Fabbro, K. Lee, and Y. Mitsufuji, "Reverse engineering of music mixing graphs with differentiable processors and iterative pruning," *JAES*, vol. 73, pp. 344–365, June 2025.

[12] Chin-Yun Yu and György Fazekas, "GOLF: A Singing Voice Synthesiser with Glottal Flow Wavetables and LPC Filters," *TISMIR*, Dec 2024.

[13] Guy E. Blelloch, "Prefix sums and their applications," Tech. Rep. CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, Nov. 1990.

[14] Mark Harris, Shubhabrata Sengupta, and John D Owens, "Parallel prefix sum (scan) with CUDA," *GPU gems*, vol. 3, no. 39, pp. 851–876, 2007.

[15] Julius O. Smith, *Introduction to Digital Filters with Audio Applications*, W3K Publishing, 2007.

[16] Jean Laroche, "On the stability of time-varying recursive filters," *JAES*, vol. 55, pp. 460–471, 2007.

[17] Udo Zölzer, *DAFX: Digital Audio Effects*, chapter Nonlinear Processing, pp. 110–112, John Wiley & Sons, 2011.

[18] Chin-Yun Yu, Christopher Mitcheltree, Alistair Carson, Stefan Bilbao, Joshua D. Reiss, and György Fazekas, "Differentiable all-pole filters for time-varying audio systems," in *DAFx*, 2024, pp. 345–352.

[19] Ben Hayes, Charalampos Saitis, and György Fazekas, "Sinusoidal frequency estimation by gradient descent," in *IEEE ICASSP*, 2023, pp. 1–5.

[20] S. Lee, M. A. Martínez-Ramírez, W.-H. Liao, S. Uhlich, G. Fabbro, K. Lee, and Y. Mitsufuji, "GRAFX: An Open-Source Library for Audio Processing Graphs in PyTorch," in *DAFx Demo Papers*, 2024, pp. 475–478.

[21] Sungho Lee, Hyeong-Seok Choi, and Kyogu Lee, "Differentiable artificial reverberation," *IEEE/ACM TASLP*, vol. 30, pp. 2541–2556, 2022.

[22] John Stautner and Miller Puckette, "Designing multi-channel reverberators," *Computer Music Journal*, vol. 6, no. 1, pp. 52–65, 1982.

[23] Gloria Dal Santo, Karolina Prawda, Sebastian Schlecht, and Vesa Välimäki, "Differentiable feedback delay network for colorless reverberation," in *DAFx*, 2023, pp. 244–251.

[24] S. J. Schlecht and E. A. P. Habets, "On lossless feedback delay networks," *IEEE TSP*, vol. 65, no. 6, pp. 1554–1564, 2016.

[25] R. Giampiccolo, A. I. Mezza, and A. Bernardini, "Differentiable MIMO feedback delay networks for multichannel room impulse response modeling," in *DAFx*, 2024, pp. 278–285.

[26] A. I. Mezza, R. Giampiccolo, and A. Bernardini, "Modeling the frequency-dependent sound energy decay of acoustic environments with differentiable feedback delay networks," in *DAFx*, 2024, pp. 238–245.

[27] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive mir research.," in *ISMIR*, 2014, pp. 155–160.

[28] R. M. Bittner, J. Wilkins, H. Yip, and J. P. Bello, "MedleyDB 2.0: New data and a system for sustainable data collection," *ISMIR Late Breaking and Demo Papers*, 2016.

[29] Alec Wright and Vesa Välimäki, "Perceptual loss function for neural modeling of audio systems," in *IEEE ICASSP*, 2020, pp. 251–255.

[30] S. Nercessian, R. McClellan, and A. Lukin, "A direct microdynamics adjusting processor with matching paradigm and differentiable implementation," in *DAFx*, 2022, pp. 248–255.

[31] R. ITU-R, "ITU-R BS. 1770-4: Algorithms to measure audio programme loudness and true-peak audio level," *International Telecommunication Union*, 2015.

[32] Eric Martin and Chris Cundy, "Parallelizing linear recurrent neural nets over sequence length," in *ICLR*, Vancouver, Canada, 2018.

[33] Joe H. Ward Jr., "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[34] Corentin Guezenoc and Renaud Seguier, "A Wide Dataset of Ear Shapes and Pinna-Related Transfer Functions Generated by Random Ear Drawings," *The Journal of the Acoustical Society of America*, vol. 147, no. 6, pp. 4087–4096, 2020.

[35] I. T. Jolliffe, *Principal Component Analysis*, Springer, 2002.

[36] J. P. Royston, "Some techniques for assessing multivarate normality based on the Shapiro-Wilk W," *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 32, no. 2, pp. 121–133, 6 1983.

[37] N. Henze and B. Zirkler and, "A class of invariant consistent tests for multivariate normality," *Commun. Stat. Theory Methods*, vol. 19, no. 10, pp. 3595–3617, 1990.

[38] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes," *Psychological research*, vol. 58, pp. 177–192, 1995.