SCRAPL: SCATTERING TRANSFORM WITH RANDOM PATHS FOR MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The Euclidean distance between differentiable wavelet scattering transform coefficients (known as *paths*) provides informative gradients for perceptual quality assessment of deep inverse problems in computer vision, speech, and audio processing. However, these transforms are computationally expensive when employed as differentiable loss functions for stochastic gradient descent due to their numerous paths, which significantly limits their use in neural network training. Against this problem, we propose "**Sc**attering transform with **Ra**ndom **P**aths for machine **L**earning" (SCRAPL): a stochastic optimization scheme for efficient evaluation of multivariable scattering transforms. We implement SCRAPL for the joint time–frequency scattering transform (JTFS) which demodulates spectrotemporal patterns at multiple scales and rates, allowing a fine characterization of intermittent auditory textures. We apply SCRAPL to differentiable digital signal processing (DDSP), specifically, unsupervised sound matching of a granular synthesizer and the Roland TR-808 drum machine. We also propose an initialization heuristic based on importance sampling, which adapts SCRAPL to the perceptual content of the dataset, improving neural network convergence and evaluation performance. We make our audio samples available and provide SCRAPL as a Python package.

1 INTRODUCTION

A scattering transform (ST) is a wavelet-based nonlinear operator which decomposes a high-resolution input $\boldsymbol{x}$ into a collection $\Phi\boldsymbol{x}$ of low-resolution coefficients, known as *paths* (Mallat, 2012). Without loss of generality, let us consider a two-layer multivariable ST of a time-domain signal $\boldsymbol{x}(t)$:

$$\Phi\boldsymbol{x}(p,t,\lambda) = \rho\Big(\Big(\Big||\mathbf{W}\boldsymbol{x}| \circledast \boldsymbol{\Psi}_p\Big| \circledast \boldsymbol{\Psi}_0\Big)(t,\lambda)\Big). \tag{1}$$

In the equation above, $\mathbf{W}$ is a wavelet transform; the vertical bars denote complex modulus; the circled asterisk $\circledast$ denotes a multivariable convolution over time t and wavelet scale λ; $\boldsymbol{\Psi}$ is a multivariable wavelet filterbank which is indexed by path p; $\boldsymbol{\Psi}_0$, i.e., $\boldsymbol{\Psi}_p$ with $p = 0$ is a multivariable low-pass filter; and ρ is a pointwise nonlinearity, e.g., path normalization and logarithmic transformation.

The design of the filterbank $\boldsymbol{\Psi}$ aims at a tradeoff between three properties: invariance to rigid motion, stability to small deformations, and separation of sparse patterns (Mallat, 2016). In speech and audio processing, examples of such $\boldsymbol{\Psi}$ include "plain" time ST (Andén & Mallat, 2014); joint time–frequency scattering (JTFS) (Andén & Mallat, 2014); and spiral ST (Lostanlen & Mallat, 2016). In computer vision, examples include "plain" 2-D ST (Bruna & Mallat, 2013); joint roto-translation ST Sifre & Mallat (2013); and scalo-roto-translation ST (Oyallon et al., 2014).

The squared Euclidean distance between scattering coefficients, or *ST distance* for short, is:

$$d_\Phi(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \sum_{p=0}^{P-1}\sum_{t=0}^{T-1}\sum_{\lambda=0}^{\Lambda-1}\Big|\Phi\boldsymbol{x}(p,t,\lambda) - \Phi\tilde{\boldsymbol{x}}(p,t,\lambda)\Big|^2, \tag{2}$$

where P is the number of paths; T is the number of time samples; and Λ is the number of scales. Behavioral studies suggest that ST distance is a good predictor of dissimilarity judgments between isolated sounds, for suitably chosen $\boldsymbol{\Psi}$ and ρ (Patil et al., 2012; Lostanlen et al., 2021; Tian et al., 2025). Relatedly, neurophysiology studies suggest that JTFS is a suitable idealized model of spectrotemporal receptive fields in the auditory cortex of humans (Norman-Haignere & McDermott, 2018) and nonhuman mammals (Kowalski et al., 1996). These findings motivate the use of JTFS as part of a differentiable loss function for neural audio models (Vahidi et al., 2023).

As an illustration, let $\boldsymbol{x}$ be a fixed reference and $\tilde{\boldsymbol{x}} = F_{\boldsymbol{x}}(\boldsymbol{w})$ be its reconstruction by an autoencoder F with trainable weights $\boldsymbol{w}$. Denoting the set of path indices by $\mathscr{P} = \{0, \ldots, P-1\}$ and the vector of all time–frequency entries $\Phi\boldsymbol{x}(p,t,\lambda)$ for each path $p \in \mathscr{P}$ by $\phi_p(\boldsymbol{x})$, the ST loss function writes as:

$$\mathscr{L}_{\boldsymbol{x}}^{\Phi}(\tilde{\boldsymbol{x}}) = \frac{1}{P}\sum_{p=0}^{P-1} \mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}}) \quad \text{where} \quad \forall p \in \mathscr{P},\ \mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}}) = P\big\|\phi_p(\boldsymbol{x}) - \phi_p(\tilde{\boldsymbol{x}})\big\|^2. \tag{3}$$

Unfortunately, $\mathscr{L}_{\boldsymbol{x}}^{\Phi}(\tilde{\boldsymbol{x}})$ and its gradient $\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\Phi}(\tilde{\boldsymbol{x}})$ are expensive in memory and in operations. Certainly, algorithmic refinements such as FFT-based filtering, multirate processing, and depth-first search can reduce the cost of an ST path (Oyallon et al., 2018). Yet, the need to traverse all P paths remains an obstacle to the applicability of multivariable ST for gradient-based learning at scale.

In this article, we aim to accelerate the training of an autoencoder F whose loss is ST distance between reference and reconstruction, and so over a finite corpus $\mathscr{X} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\}$. Formally:

$$\boldsymbol{w}^{\star} = \arg\min_{\boldsymbol{w}} \frac{1}{N}\sum_{n=0}^{N-1} \left(\mathscr{L}_{\boldsymbol{x}_n}^{\Phi} \circ F_{\boldsymbol{x}_n}\right)(\boldsymbol{w}). \tag{4}$$

Given the decomposition in Equation 3, a naïve idea would be to replace each term $\mathscr{L}_{\boldsymbol{x}_n}^{\Phi}$ in the equation above by some per-path loss $\mathscr{L}_{\boldsymbol{x}_n}^{\phi_p}$, where the p's would be drawn independently and uniformly at random in the path set $\mathscr{P}$. This is a crude form of stochastic approximation (Benveniste et al., 2012) which is motivated by the tree-like structure of ST: neglecting the overhead of the first layer ($|\mathbf{W}\boldsymbol{x}|$), the computation of single-path gradient $\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}_n}^{\phi_p}$ is roughly P times more efficient than that of a full ST gradient $\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}_n}^{\Phi}$. However, this speedup comes at the detriment of numerical precision: a deterministic quantity has been replaced by an estimator whose variance may be impractically large.

"**Sc**attering transform with **Ra**ndom **P**aths for machine **L**earning" (SCRAPL) is our proposed solution to this problem. Acknowledging that each single-path gradient makes for an inexpensive but noisy learning signal, we stabilize it via a combination of three stochastic optimization techniques. Our contributions are:

1. **Stochastic approximation of scattering transform**: through uniform sampling of paths.
2. **Path-wise adaptive moment estimation** ($\mathscr{P}$-Adam for short): an extension of the Adam algorithm (Kingma & Ba, 2014) which accounts for the non-i.i.d. nature of ST paths.
3. **Path-wise stochastic average gradient with acceleration** ($\mathscr{P}$-SAGA for short): a variant of the SAGA algorithm (Defazio et al., 2014) which keeps a memory of previous gradient values across all paths p.
4. **θ-importance sampling** supplies auxiliary information to the stochastic optimizer by sampling paths p in proportion to the typical rate of change of the gradient in the optimization landscape.

Our main empirical finding is that SCRAPL accomplishes a favorable tradeoff between goodness of fit and computational efficiency on unsupervised sound matching, i.e., a nonlinear inverse problem in which the forward operator implements an audio synthesizer. In the context of differentiable digital signal processing (DDSP), the state-of-the-art perceptual loss function for this task is multiscale spectral loss (MSS, Engel et al. (2020)). However, the gradient of MSS is uninformative when input and reconstruction are misaligned or when the synthesizer controls involve spectrotemporal modulations (Vahidi et al., 2023). Taking advantage from the stability guarantees of JTFS, SCRAPL expands the class of synthesizers which can be effectively decoded via DDSP.

Figure 1 illustrates one of our experiments: unsupervised sound matching in a nondeterministic granular synthesizer. On one hand, models based on MSS and other state-of-the-art perceptual losses are computationally efficient but inaccurate. On the other hand, JTFS-based models are five times more accurate but one hundred times more costly. SCRAPL is a new point on this Pareto front: it is within a factor two of JTFS in terms of accuracy while being within a factor three of MSS in terms of runtime, making it suitable for large-scale DDSP. Relatedly, SCRAPL is also more memory-efficient than JTFS, thus reducing overhead between cores and allowing for a larger batch size.

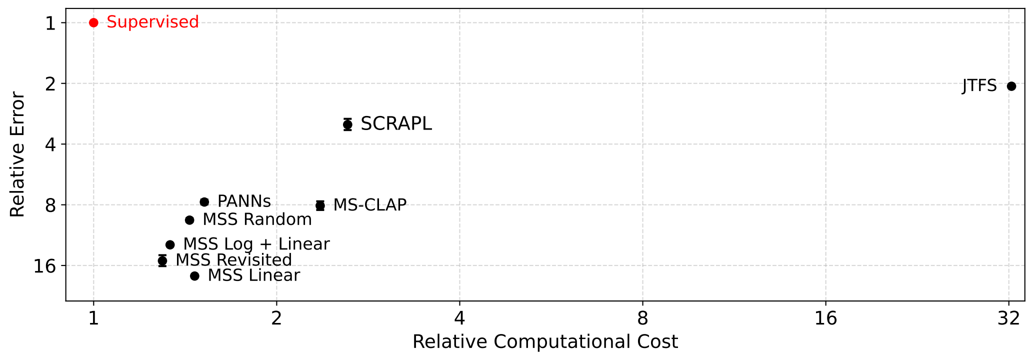

Figure 1: Mean average error (y-axis) versus computational cost (x-axis) of unsupervised sound matching models for the granular synthesis task. Both axes are rescaled by the performance of a supervised model with same number of parameters. Whiskers denote 95% CI, estimated over 20 random seeds. Due to computational limitations, JTFS-based sound matching is evaluated only once.

2 Related Work

The guiding intuition behind SCRAPL is that natural signals and images exhibit strong correlations across ST paths. This fact has been observed empirically since the onset of ST research (Bruna & Mallat, 2013; Andén & Mallat, 2011) and aligns with earlier work on texture modeling based on pairwise correlations between wavelet modulus coefficients (Portilla & Simoncelli, 2000).

Visual and auditory textures, understood as stationary random fields, play a key role in applied ST research. ST features outperform short-term Fourier features (e.g., MSS) in their ability to characterize intermittency in non-Gaussian textures (Muzy et al., 2015). Texture resynthesis by gradient descent of ST loss has been applied to such diverse settings as computer music creation (Lostanlen et al., 2019) and the study of the cosmic microwave background (Delouis et al., 2022).

The democratization of differentiable programming toolkits (e.g., TensorFlow, PyTorch, JAX) have greatly advanced the flexibility of gradient backpropagation in "hybrid" scattering–neural networks involving learnable and non-learnable modules. Angles & Mallat (2018) have built a hybrid scattering–GAN model for image generation, in which ST distance plays the role of a discriminator.

To our knowledge, the closest prior work to SCRAPL is the pruned graph scattering transform (pGST) of Ioannidis et al. (2020), a method which reduces the complexity of ST by pruning down the path set $\mathscr{P}$ down to a proper subset $\mathscr{P}' \subset \mathscr{P}$, based on a graph-spectrum-inspired criterion. Although both pGST and SCRAPL share a similar overarching goal, let us point out that pGST is a feature selection method: the cardinality of $\mathscr{P}'$ is typically $\sim 10\%$ that of $\mathscr{P}$ and $\mathscr{P}'$ is kept fixed across training examples and across epochs. In comparison, SCRAPL performs a more radical pruning, down to a single path (card $\mathscr{P}' = 1$), while harnessing dedicated techniques in stochastic optimization ($\mathscr{P}$-Adam and $\mathscr{P}$-SAGA) to reduce the variance of ST loss during gradient backpropagation.

3 Methods

3.1 Stochastic Approximation of Scattering Transform Loss Gradient

The proposition below, proven in Appendix A, shows that if paths are drawn uniformly at random, the stochastic approximation in SCRAPL is unbiased: in other words, the expected value of the stochastic gradient of per-path loss is equal to the gradient of full ST loss.

Proposition 3.1. Let $\Phi = (\phi_p)_0^{P-1}$ be a scattering transform with P paths. Given a signal or image $\boldsymbol{x}$, let $F_{\boldsymbol{x}}$ be an autoencoder operating on $\boldsymbol{x}$ and let $\mathscr{L}_{\boldsymbol{x}}^{\Phi}$ be the associated ST reconstruction loss. Let $\mathscr{U}_P$ be the uniform distribution over $\mathscr{P} = \{0, \ldots, P-1\}$. One has, for every weight vector $\boldsymbol{w}$:

$$\mathbb{E}_{z \sim \mathscr{U}_P}\left[\boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\phi_z} \circ F_{\boldsymbol{x}})(\boldsymbol{w})\right] = \boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\Phi} \circ F_{\boldsymbol{x}})(\boldsymbol{w}). \tag{5}$$

Although a uniform sampling of paths matches the intuition of approximating the ST gradient in expectation, we will see that this may be suboptimal. The θ-importance sampling method, which we will present in Section 3.4, does not satisfy the hypothesis of Proposition 3.1; yet, it consistently outperforms uniform sampling as part of SCRAPL. The design of biased stochastic approximation schemes is an active topic in machine learning research (Dieuleveut et al., 2023).

3.2 $\mathscr{P}$-ADAM: PATH-WISE ADAPTIVE MOMENT ESTIMATION

The key idea behind the Adam optimizer is to smooth the successive realizations of the stochastic gradient, here denoted by $\boldsymbol{g}$, via autoregressive estimates of its first- and second-order element-wise moments, denoted by $\boldsymbol{m}$ and $\boldsymbol{v}$ (Kingma & Ba, 2014). However, the smoothing technique in Adam is ineffective for SCRAPL because the gradients of path-wise ST losses are not identically distributed. Against this problem, we propose to maintain P estimates of path-wise moments ($\mathscr{P}$-Adam):

$$\boldsymbol{m}_p \leftarrow \beta_1^{(k-\tau_p)/P}\boldsymbol{m}_p + (1-\beta_1^{(k-\tau_p)/P})\boldsymbol{g} \tag{6}$$

$$\boldsymbol{v}_p \leftarrow \beta_2^{(k-\tau_p)/P}\boldsymbol{v}_p + (1-\beta_2^{(k-\tau_p)/P})(\boldsymbol{g}\odot\boldsymbol{g}), \tag{7}$$

where k is the current iteration number, τ_p is the iteration when path p was last drawn; β_1 and β_2 are hyperparameters; and the circled dot denotes element-wise multiplication of vectors. The exponent $(k-\tau_p)/P$ adapts the time constant of smoothing to the recency of the previous estimate.

The second step in $\mathscr{P}$-Adam, following classical Adam, consists in bias correction and ratio of debiased first-order moment to stable square root of debiased second-order moments:

$$\boldsymbol{g}_{\text{current}} = \frac{\dfrac{\boldsymbol{m}_p}{1-\beta_1^{k/P}}}{\sqrt{\varepsilon + \dfrac{\boldsymbol{v}_p}{1-\beta_2^{k/P}}}}, \tag{8}$$

where we have adapted the original exponents of Adam (β_1^k, β_2^k) to account for the number of paths.

3.3 $\mathscr{P}$-SAGA: PATH-WISE STOCHASTIC AVERAGE GRADIENT WITH ACCELERATION

The stochastic average gradient (SAG) algorithm has the potential to accelerate stochastic gradient descent in the context of the minimization of finite sums (Schmidt et al., 2017). Although this sum is typically over training examples in neural network training, in SCRAPL, Equation 3 is a sum over paths for a given example $\boldsymbol{x}$. With this observation in mind, we propose $\mathscr{P}$-SAGA, a path-wise version of SAG with acceleration (SAGA, Defazio et al. (2014)). We maintain a memory of the last $\mathscr{P}$-Adam updates over each path, denoted by $(\hat{\boldsymbol{g}}_p)_0^{P-1}$; and the set of paths previously visited, denoted by Γ. Given a learning rate α_k at iteration k, the $\mathscr{P}$-SAGA update is:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha_k\left(\boldsymbol{g}_{\text{current}} - \hat{\boldsymbol{g}}_p + \frac{\sum_{\gamma\in\Gamma}\hat{\boldsymbol{g}}_\gamma}{\max(1,\operatorname{card}\Gamma)}\right). \tag{9}$$

Algorithm 1 in Appendix B summarizes SCRAPL with both $\mathscr{P}$-Adam and $\mathscr{P}$-SAGA enabled.

3.4 θ-IMPORTANCE SAMPLING

We now consider the important special case of differentiable digital signal processing (DDSP, see Section 1), in which the autoencoder composes a non-learnable decoder with a learned encoder: i.e., $F_{\boldsymbol{x}} = (D\circ E_{\boldsymbol{x}})$ (Engel et al., 2020). We assume both D and $E_{\boldsymbol{x}}$ to be differentiable with respect to their inputs, but D is not necessarily deterministic. We denote by U the dimension of the parameter space; i.e., the output space of $E_{\boldsymbol{x}}$ and input space of D.

A known drawback of DDSP is that the optimization landscape of spectral loss in parameter space (i.e., of $\mathscr{L}_{\boldsymbol{x}}^{\Phi}\circ D$) may not coincide with that of P-loss (i.e., Euclidean distance to θ) (Hayes et al., 2024). Against this drawback, we propose a method named θ-importance sampling (θ-IS), which constructs a categorical distribution π over the path space $\mathscr{P}$. The key idea behind θ-IS is to introduce bias in the stochastic approximation of spectral loss so as to bring it closer to P-loss. For lack of supervision, we are unable to construct the optimal distribution π but provide a heuristic of this form:

$$\pi_p = \frac{1}{U}\sum_{u=0}^{U-1}\frac{C_{u,p}}{\sum_{p=0}^{P-1} C_{u,p'}}, \tag{10}$$

where, intuitively, $C_{u,p}$ represents the importance of parameter dimension θ_u upon path p. We rescale this importance relative to all paths and average uniformly across parameters u, yielding an importance-weighted categorical distribution π over paths.

Let $E_{\boldsymbol{x},u}(\boldsymbol{w})$ denote the u^{th} coordinate of $E_{\boldsymbol{x}}(\boldsymbol{w})$. Given $\boldsymbol{w}$, we measure the sensitivity of each ST path p to the parameter control u around the input $\boldsymbol{x}$ in terms of the following partial derivative:

$$\mathbf{s}_{\boldsymbol{x},u,p} : \boldsymbol{w} \longmapsto \frac{\partial\left(\mathscr{L}_{\boldsymbol{x}}^{\phi_p} \circ D\right)}{\partial \theta_u}\Big(E_{\boldsymbol{x},u}(\boldsymbol{w})\Big) \tag{11}$$

To convert the sensitivity function $\mathbf{s}_{\boldsymbol{x},u,p}$ into relative importance $C_{u,p}$, we multiply it by the transposed gradient of $E_{\boldsymbol{x},u}$, yielding a vector field mapping neural network parameters to synth parameters. We evaluate the gradient of this vector field at $\boldsymbol{w}$, yielding a square matrix; compute its largest eigenvalue; and repeat the process over a representative dataset $\mathscr{X}$ of unlabeled signals. Formally:

$$C_{u,p} = \mathbb{E}_{\boldsymbol{x}\sim\mathscr{X}}\left[\lambda_{\max}\left(\boldsymbol{\nabla}_{\boldsymbol{w}}\left(s_{\boldsymbol{x},u,p}(\boldsymbol{w})\boldsymbol{\nabla} E_{\boldsymbol{x},u}(\boldsymbol{w})^{\top}\right)\right)\right], \tag{12}$$

where $\lambda_{\max}(\mathbf{M})$ is the largest eigenvalue of a square matrix $\mathbf{M}$; $\boldsymbol{\nabla}_{\boldsymbol{w}}$ is the gradient with respect to $\boldsymbol{w}$. In practice, we compute $\lambda_{\max}(\mathbf{M})$ using a stochastic power iteration with deflation[1] and the Hessian vector product (HVP), which has the same asymptotic runtime complexity as a backpropagation step.

This measure is inspired by Schmidt et al. (2017), who propose a variant of the SAG algorithm in which mini-batches are sampled non-uniformly; more precisely, in proportion to the Lipschitz constant of the gradients. This heuristic relies on the argument that gradients which change quickly should be regarded as more important than gradients which change slowly.

4 EXPERIMENTS

We apply SCRAPL to a differentiable implementation of the joint time–frequency scattering transform (Muradeli et al., 2022). We conduct three unsupervised sound matching experiments under the DDSP paradigm. The encoder, $E_{\boldsymbol{x}}$, is a convolutional neural network which operates on a learnable constant-Q transform (Cheuk et al., 2020).

To highlight the new kinds of perceptual quality assessment tasks SCRAPL enables, all three experiments investigate nondeterministic decoders that introduce random time shifts into the resulting reconstructed audio. While our experiments are for a discriminative and generative audio processing task, it is important to emphasize that SCRAPL is a general algorithm for scattering transforms and can be equally applied to deep inverse problems in other domains like computer vision.

4.1 JOINT TIME–FREQUENCY SCATTERING TRANSFORM (JTFS)

The joint time–frequency scattering transform (JTFS) is a nonlinear convolutional operator which extracts spectrotemporal modulations over the constant-Q spectrogram (Andén et al., 2019). The multivariable filter $\boldsymbol{\Psi}_p$ comprises two stages: temporal scattering, i.e., 1-D band-pass filtering with Morlet wavelets over the time axis; and frequential scattering, i.e., idem over the log-frequency axis. The center frequencies of band-pass filters for temporal scattering, called *rates*, are measured in Hertz. The center frequencies for frequential scattering, called *scales*, are measured in cycles per octave. Thus, in the case of JTFS, the path index p is a rate–scale multiindex.

4.2 GRANULAR SYNTH SOUND MATCHING

Granular synthesis is an example of a new class of synths that can be effectively sound matched with SCRAPL and the JTFS, due to its inherently stochastic audio generation process with individual grains

[1] https://github.com/noahgolmant/pytorch-hessian-eigenthings

being misaligned in time at the micro-level, but still being perceived as a single texture. It has been extensively used in the production of electronic music since the late 1950s [2] and played a fundamental role in the creation of contemporary music genres such as future bass. Our differentiable granular synth produces textures of chirplet grains with random temporal positions, center frequencies, and chirp rates, and has two parameters: density (θ_{density}) which controls how many grains are produced, and slope (θ_{slope}) which controls their rate of frequency modulation.

We compare four MSS-based losses: linear, log + linear (Engel et al., 2020), random (Steinmetz & Reiss, 2020), and a SOTA hyperparameter-tuned revisited MSS loss (Schwär & Müller, 2023). Given their correlation with human perception (Kilgour et al., 2019; Tailleur et al., 2024), we also include the Euclidean distance of MS-CLAP (Elizalde et al., 2023) and PANNs Wavegram Logmel embeddings (Kong et al., 2020). In addition, we train with ordinary (i.e., full-tree) JTFS so as to put the speed and accuracy of SCRAPL into context. Lastly, as an estimate of best achievable performance with this neural network architecture, we run a supervised version of sound matching, under the name of "parameter loss" or P-loss for short. See Appendix F for implementation details.

4.3 Chirplet Synth Sound Matching

Similar to the unsupervised granular synth sound matching experiment, we evaluate our θ-importance sampling initialization heuristic for SCRAPL on a differentiable chirplet synth (based on the implementation by Vahidi et al. (2023)) with two parameters: θ_{AM} which controls the rate of amplitude modulation (Hz) and θ_{FM} which controls the rate of frequency modulation (oct/s). Since the paths in the JTFS correspond to specific wavelet AM and FM center frequencies, given a chirplet synth configuration with bounded θ_{AM} and θ_{FM} ranges, we know which paths of the JTFS should provide the most informative gradients for the synth parameters. After computing our initialization heuristic, we can analyze the resulting path probabilities and verify that the paths within the parameter ranges of the synth have been assigned a probability greater than uniform.

We evaluate four different synth configurations:

1. Slow AM ($\theta_{\text{AM}} \in [1.0, 2.0]$ Hz), slow FM ($\theta_{\text{FM}} \in [0.5, 1.0]$ oct/s);
2. Slow AM ($\theta_{\text{AM}} \in [1.0, 2.0]$ Hz), moderate FM ($\theta_{\text{FM}} \in [2.0, 4.0]$ oct/s);
3. Fast AM ($\theta_{\text{AM}} \in [2.8, 8.4]$ Hz), moderate FM ($\theta_{\text{FM}} \in [2.0, 4.0]$ oct/s);
4. Fast AM ($\theta_{\text{AM}} \in [2.8, 8.4]$ Hz), fast FM ($\theta_{\text{FM}} \in [4.0, 12.0]$ oct/s).

We compare SCRAPL training runs using uniform sampling and θ-importance sampling calculated from a single training batch of 32 examples. See Appendix F for implementation details.

4.4 Roland TR-808 Sound Matching

As a real-world evaluation task, we sound match a DDSP implementation (Shier et al., 2024) of the Roland TR-808 Drum Machine, a historically meaningful synthesizer for the creation of Detroit techno, house, and hip-hop music[3]. Inharmonic transient sounds like percussion are a form of non-stationary signal that the JTFS is well suited for perceptual quality assessment (Han et al., 2024) due to its ability to extract spectrotemporal patterns at multiple scales and rates. Additionally, due to the transient nature of drum sounds, they are highly sensitive to even a few milliseconds of misalignment, thus further benefiting from the time invariance of JTFS.

We use a high fidelity, 100% analog dataset[4] of 681 bass drum, snare, tom, and hi-hat one-shot recordings of the TR-808 and repeat experiments 40 times on different train/validation/test splits and random seeds. Since the transient of analog drum recordings is rarely perfectly aligned, and no two analog TR-808 drum synths produce the same signal, we investigate both perfectly aligned sound matching (labeled *micro*) and unaligned sound matching (labeled *meso)* by up to $\pm$46 ms ($\pm$2048 samples at 44.1 kHz).

[2] https://www.iannis-xenakis.org/en/granular-synthesis/

[3] https://www.ethanhein.com/wp/2016/beatmaking-fundamentals/

[4] https://samplesfrommars.com/products/tr-808-samples

Table 1: Evaluation results for the unsupervised granular synth sound matching task. Uncertainties are 95% confidence intervals for 20 training runs using different random seeds. Due to computational limitations, the JTFS method is only evaluated once.

Method	$\boldsymbol{\theta}_{\text{synth}}$ $\boldsymbol{L_1}$ ‰ ↓	$\boldsymbol{\theta}_{\text{density}}$ $\boldsymbol{L_1}$ ‰ ↓	$\boldsymbol{\theta}_{\text{slope}}$ $\boldsymbol{L_1}$ ‰ ↓
JTFS	**42.4**	**65.8**	**19.0**
SCRAPL (no θ-IS)	73.8 ± 13	70.4 ± 8.8	77.2 ± 19
SCRAPL	65.7 ± 4.2	72.6 ± 6.3	58.7 ± 7.5
MSS Linear	370 ± 0.52	499 ± 0.84	241 ± 0.28
MSS Log + Linear	259 ± 1.7	277 ± 3.2	241 ± 0.42
MSS Revisited	311 ± 19	376 ± 40	246 ± 3.0
MSS Random	195 ± 4.2	149 ± 7.8	242 ± 1.0
MS-CLAP	166 ± 8.2	81.9 ± 9.0	250 ± 8.2
PANNs Wavegram-Logmel	159 ± 4.4	80.3 ± 4.2	238 ± 5.5
P-Loss	20.5 ± 0.20	24.7 ± 0.31	16.3 ± 0.31

We employ MSS and JTFS audio distance as evaluation metrics, as well as mean frame-by-frame perceptual loudness and loudness-weighted perceptually-scaled spectral centroid and flatness for both the transient and decay portions of reconstructed signals (eight metrics in total). Additional context for these last six metrics can be found in Shier et al. (2024). See Appendix F for all implementation details.

5 Results

5.1 Granular Synth Sound Matching

We benchmark all loss function computational costs (see Appendix C, Table 5) and plot them in Figure 1 against their evaluation accuracy (see Table 1) on $\boldsymbol{\theta}_{\text{synth}}$ L_1 relative to supervised training. We observe that SCRAPL comes within a factor of two of JTFS in terms of accuracy, and within a factor of three of MSS in terms of runtime, striking a notable balance between the two. The significant difference in runtime and convergence between JTFS and SCRAPL is further illustrated in Figure 2 where we plot validation accuracy against wall-clock time, instead of optimization steps. We also note that MSS is unable to sound match the synth at all, and the SOTA embedding losses are only able to optimize $\boldsymbol{\theta}_{\text{density}}$, albeit not as well as SCRAPL and JTFS. Validation accuracy curves for all methods are provided in Figure 2.

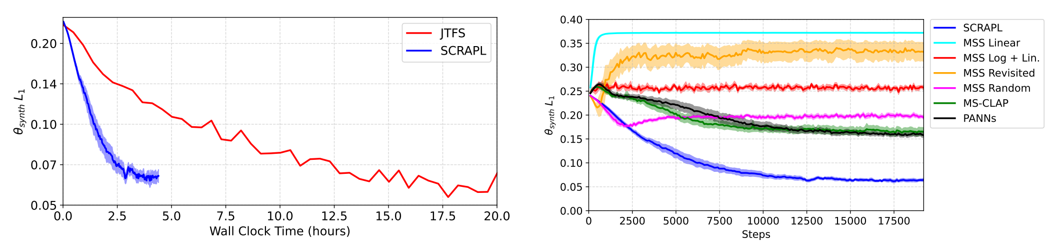

Figure 2: Left: JTFS vs. SCRAPL wall-clock training times on a single NVIDIA RTX A5000 GPU. Shaded areas are 95% confidence intervals for 20 training runs using different random seeds. Due to computational limitations, the JTFS method is only evaluated once. Right: Validation convergence graphs for the unsupervised granular synth sound matching task. Shaded areas are 95% confidence intervals for 20 training runs using different random seeds.

Table 2 summarizes the results of an ablation of SCRAPL and its $\mathscr{P}$-Adam, $\mathscr{P}$-SAGA, and θ-IS optimization techniques for the granular synth sound matching task. There is a clear monotonic improvement in accuracy and convergence time for each technique, as well as a reduction in variance provided by $\mathscr{P}$-SAGA, and θ-IS. It is also worth noting that SCRAPL without any extra stochastic optimization techniques still outperforms all other non-JTFS methods in terms of accuracy, making

Table 2: Ablation table for SCRAPL with test results and validation θ_{synth} L_1 total variation and convergence steps for the unsupervised granular synth sound matching task. Convergence is defined as θ_{synth} $L_1 < 100$ ‰. Uncertainties are 95% confidence intervals for 20 training runs using different random seeds. Due to computational limitations, the JTFS method is only evaluated once.

Method	$\mathscr{P}$-Adam	$\mathscr{P}$-SAGA	θ-IS	Test: θ_{synth} L_1 ‰ ↓	Validation: Total Var. ↓	Validation: Conv. Steps ↓
SCRAPL	✗	✗	✗	99.7 ± 8.2	5.30 ± 0.25	10 906 ± 1170
	✓	✗	✗	87.4 ± 15	6.98 ± 0.25	8006 ± 697
	✓	✓	✗	73.8 ± 13	3.46 ± 0.15	7296 ± 683
	✓	✓	✓	**65.7** ± 4.2	**3.27** ± 0.12	**6014** ± 642
JTFS				42.4	5.66	1442
P-Loss				20.5 ± 0.20	1.83 ± 0.025	672 ± 23

Table 3: Evaluation results for SCRAPL with and without the θ-importance sampling initialization heuristic on unsupervised sound matching of four different AM / FM chirplet synths. Uncertainties are 95% confidence intervals for 20 training runs using different random seeds.

Sampling Method	Synth Configuration: θ_{AM} (Hz)	Synth Configuration: θ_{FM} (oct/s)	θ_{AM} L_1 ‰ ↓	θ_{FM} L_1 ‰ ↓
Uniform	1.0 – 2.0	0.5 – 1.0	124 ± 10	155 ± 18
θ-IS	1.0 – 2.0	0.5 – 1.0	**77.7** ± 6.7	**78.4** ± 11
Uniform	1.0 – 2.0	2.0 – 4.0	111 ± 20	68.6 ± 11
θ-IS	1.0 – 2.0	2.0 – 4.0	**55.5** ± 4.1	**44.4** ± 2.8
Uniform	2.8 – 8.4	2.0 – 4.0	122 ± 22	238 ± 21
Adaptive	2.8 – 8.4	2.0 – 4.0	**54.9** ± 3.5	**48.5** ± 4.7
Uniform	2.8 – 8.4	4.0 – 12.0	108 ± 12	95.6 ± 20
θ-IS	2.8 – 8.4	4.0 – 12.0	**81.5** ± 12	**82.1** ± 11

stochastic sampling of scattering transforms a viable alternative if the additional memory and computational requirements of $\mathscr{P}$-Adam, $\mathscr{P}$-SAGA, and θ-IS are undesirable. Finally, from Table 1, we see that θ-IS results in a better overall accuracy of θ_{synth} than uniform sampling (despite θ_{density} now being slightly worse), which is consistent with our hypothesis from Section 3.4 that θ-IS results in more balanced convergence of all synth parameters. Validation accuracy curves for all ablations are provided in Appendix C, Figure 3.

5.2 Chirplet Synth Sound Matching

Table 3 summarizes the chirplet synth evaluation results, with Appendix D, Figure 4 showing validation accuracy curves for uniform and θ-importance sampling on the four synth configurations. θ-IS improves the prediction of θ_{AM} by 25–50% and of θ_{FM} by 15–80%, while reducing time to convergence by 30–50%: see Appendix D, Table 6. Of course, these improvements are for synth configurations that have been designed to showcase the benefit of nonuniform sampling of paths; however, this overall trend remains true, albeit not as pronounced, for the granular synth (Table 2) and real-world sound matching task (Table 4). Finally, we plot the path θ-IS probabilities in Appendix D, Figure 5 and observe that indeed, a unique distribution is learned for each synth, and the greater than uniform probabilities appear to roughly correspond to each configuration's limited AM/FM range.

5.3 Roland TR-808 Sound Matching

Table 4 and Appendix E, Tables 7, and 8 summarize the unsupervised Roland TR-808 synth sound matching audio distance, transient, and decay perceptual similarity results. Overall, we observe that JTFS dominates almost all metrics in both micro and meso environments, showcasing its suitability

Table 4: Audio distance evaluation results for the unsupervised Roland TR-808 DDSP synth sound matching task. Uncertainties are 95% confidence intervals for 40 training runs using different random seeds and dataset splits. Due to computational limitations, the JTFS method is only trained and evaluated for 4 random seeds.

Method	**MSS Log. + Linear ↓**		**JTFS ↓**	
	Micro	Meso	Micro	Meso
JTFS	617 $\pm$46	622 $\pm$45	**490** $\pm$28	**523** $\pm$17
SCRAPL (no θ-IS)	862 $\pm$36	944 $\pm$48	1140 $\pm$48	1250 $\pm$51
SCRAPL	857 $\pm$42	879 $\pm$42	1050 $\pm$50	1110 $\pm$52
MSS Linear	611 $\pm$15	724 $\pm$37	779 $\pm$31	1470 $\pm$83
MSS Log + Linear	**596** $\pm$19	**615** $\pm$18	1260 $\pm$58	1390 $\pm$49
MSS Revisited	637 $\pm$16	797 $\pm$20	870 $\pm$23	1250 $\pm$27
MSS Random	682 $\pm$25	700 $\pm$26	1410 $\pm$87	1500 $\pm$59

for transient percussive sounds and temporal invariance. After JTFS, MSS tends to be best when samples are perfectly aligned (micro), but performs worse in the unaligned (meso) setting and is unable to match the transient, which is the most salient part of a drum hit. In contrast, SCRAPL shows good sound matching performance in both micro and meso environments and is able to preserve the transient even when audio is misaligned. However, SCRAPL fails to recover the less audible decay portion of the signal. We hypothesize this is due to informative, low-frequency paths for the decay being sparse and underrepresented in the categorical distribution over paths, even after accounting for θ-IS. We provide listening samples at the accompanying website[5] and encourage readers to evaluate the results directly.

6 Conclusion

Differentiable similarity measures have the potential to enhance the perceptual quality of generative models and deep inverse problem solvers. In spite of their mathematical guarantees and neurophysiological plausibility, scattering transforms (ST) have not been able to realize this potential, for lack of tractable optimization algorithms. To fill this gap, SCRAPL takes advantage of the tree-like structure of ST to save computation at each backward pass. Our numerical simulations show the value of SCRAPL for unsupervised sound matching, particularly when the synthesizer of interest is nondeterministic. Although our ST architecture of choice is joint time–frequency scattering (JTFS), we stress that SCRAPL is agnostic to the specifics of multivariable filterbank design: beyond wavelet scattering, it extends to learnable scattering-like architectures (Lattner et al., 2019; Cotter & Kingsbury, 2019; Gauthier et al., 2022). As a longer-term perspective, the success of our synthesis-informed importance sampling heuristic highlights the opportunity to meta-learn the relative importance of each ST path for the task at hand over the course of neural network training (Yamaguchi et al., 2023).

[5] Anonymous companion website: https://icewithfrosting.github.io/scrapl/

REPRODUCIBILITY STATEMENT

Appendix F contains all hyperparameters and training details for each of the three experiments in this paper. We also provide listening samples, anonymized source code, configuration files, and instructions to reproduce our experiments at the anonymous companion website: https://icewithfrosting.github.io/scrapl/

REFERENCES

Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, Florida, 2011.

Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

Joakim Andén, Vincent Lostanlen, and Stéphane Mallat. Joint time–frequency scattering. *IEEE Transactions on Signal Processing*, 67(14):3704–3718, 2019.

Tomás Angles and Stéphane Mallat. Generative networks as inverse problems with scattering transforms, 2018.

Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Kin Wai Cheuk, Hans Anderson, Kat Agres, and Dorien Herremans. nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolutional neural networks. *IEEE Access*, 8: 161981–162003, 2020. doi: 10.1109/ACCESS.2020.3019084.

Fergal Cotter and Nick Kingsbury. A learnable ScatterNet: Locally invariant convolutional layers. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. 2014.

J-M Delouis, Erwan Allys, Edouard Gauvrit, and François Boulanger. Non-gaussian modelling and statistical denoising of planck dust polarisation full-sky maps using scattering transforms. *Astronomy & Astrophysics*, 668:A122, 2022.

Aymeric Dieuleveut, Gersende Fort, Eric Moulines, and Hoi-To Wai. Stochastic approximation beyond gradient for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 71:3117–3148, 2023.

Benjamin Elizalde, Soham Deshmukh, and Huaming Wang. Natural language supervision for general-purpose audio representations, 2023. arXiv preprint: https://arxiv.org/abs/2309.05767.

Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable digital signal processing. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Shanel Gauthier, Benjamin Thérien, Laurent Alsene-Racicot, Muawiz Chaudhary, Irina Rish, Eugene Belilovsky, Michael Eickenberg, and Guy Wolf. Parametric scattering networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Han Han, Vincent Lostanlen, and Mathieu Lagrange. Learning to solve inverse problems for perceptual sound matching. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2605–2615, 2024.

Ben Hayes, Jordie Shier, György Fazekas, Andrew McPherson, and Charalampos Saitis. A review of differentiable digital signal processing for music and speech synthesis. *Frontiers in Signal Processing*, 3:1284100, 2024.

Vassilis N. Ioannidis, Siheng Chen, and Georgios B. Giannakis. Pruned graph scattering transforms. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Proceedings of the International Speech Communication Association Conference (Interspeech)*, pp. 2350–2354, 2019. doi: 10.21437/Interspeech.2019-2219.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 28:2880–2894, November 2020. ISSN 2329-9290. doi: 10.1109/TASLP.2020.3030497. URL https://doi.org/10.1109/TASLP.2020.3030497.

Nina Kowalski, Didier A Depireux, and Shihab A Shamma. Analysis of dynamic spectra in ferret primary auditory cortex. i. characteristics of single-unit responses to moving ripple spectra. *Journal of Neurophysiology*, 76(5):3503–3523, 1996.

Stefan Lattner, Monika Dörfler, and Andreas Arzt. Learning complex basis functions for invariant representations of audio. In *Proceedings of the International Society Society for Music Information Retrieval Conference (ISMIR)*, 2019.

Vincent Lostanlen and Stéphane Mallat. Wavelet scattering on the pitch spiral. In *Proceedings of the Digital Audio Effects Conference (DAFx)*, 2016.

Vincent Lostanlen, Joakim Andén, and Mathieu Lagrange. Fourier at the heart of computer music: From harmonic sounds to texture. *Comptes Rendus. Physique*, 20(5):461–473, 2019.

Vincent Lostanlen, Christian El-Hajj, Mathias Rossignol, Grégoire Lafay, Joakim Andén, and Mathieu Lagrange. Time–frequency scattering accurately models auditory similarities between instrumental playing techniques. *EURASIP Journal on Audio, Speech, and Music Processing*, 2021(1):3, 2021.

Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65 (10):1331–1398, 2012.

Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

John Muradeli, Cyrus Vahidi, Changhong Wang, Han Han, Vincent Lostanlen, Mathieu Lagrange, and George Fazekas. Differentiable time-frequency scattering on gpu. In *Proceedings of the Digital Audio Effects Conference (DAFx)*, 2022.

Jean-François Muzy, Emmanuel Bacry, Stéphane Mallat, and Joan Bruna. Intermittent process analysis with scattering moments. *Annals of Statistics*, 43(1):323, 2015.

Sam V Norman-Haignere and Josh H McDermott. Neural responses to natural and model-matched stimuli reveal distinct computations in primary and nonprimary auditory cortex. *PLoS biology*, 16 (12):e2005127, 2018.

Edouard Oyallon, Stéphane Mallat, and Laurent Sifre. Generic deep networks with wavelet scattering. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop Track*, 2014.

Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, and Eugene Belilovsky. Scattering networks for hybrid representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2208–2221, 2018.

Kailash Patil, Daniel Pressnitzer, Shihab Shamma, and Mounya Elhilali. Music in our ears: The biological bases of musical timbre perception. *PLoS computational biology*, 8(11):e1002759, 2012.

Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1–2):83–112, March 2017. ISSN 0025-5610. doi: 10.1007/s10107-016-1030-6. URL https://doi.org/10.1007/s10107-016-1030-6.

Simon Schwär and Meinard Müller. Multi-scale spectral loss revisited. *IEEE Signal Processing Letters*, 30:1712–1716, 2023. doi: 10.1109/LSP.2023.3333205.

Jordie Shier, Charalampos Saitis, Andrew Robertson, and Andrew McPherson. Real-time timbre remapping with differentiable dsp. In *Proceedings of the International Conference on New Interfaces for Musical Expression NIME*, 2024. URL https://arxiv.org/abs/2407.04547.

Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1233–1240, 2013.

Christian J. Steinmetz and Joshua D. Reiss. auraloss: Audio focused loss functions in PyTorch. In *Digital Music Research Network One-day Workshop (DMRN+15)*, 2020.

Modan Tailleur, Junwon Lee, Mathieu Lagrange, Keunwoo Choi, Laurie M. Heller, Keisuke Imoto, and Yuki Okamoto. Correlation of fréchet audio distance with human perception of environmental audio is embedding dependant. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2024.

Haokun Tian, Stefan Lattner, and Charalampos Saitis. Assessing the alignment of audio representations with timbre similarity ratings. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2025.

Cyrus Vahidi, Han Han, Changhong Wang, Mathieu Lagrange, György Fazekas, and Vincent Lostanlen. Mesostructures: Beyond spectrogram loss in differentiable time-frequency analysis. *Journal of the Audio Engineering Society*, 71(9):577–585, 2023.

Shin'ya Yamaguchi, Daiki Chijiwa, Sekitoshi Kanai, Atsutoshi Kumagai, and Hisashi Kashima. Regularizing neural networks with meta-learning generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:27315–27331, 2023.

A Proof of Proposition 3.1

Let us write $\tilde{\boldsymbol{x}} = F_{\boldsymbol{x}}(\boldsymbol{w})$. By linearity of the gradient, we may decompose $\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\Phi}(\tilde{\boldsymbol{x}})$ over paths:

$$\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\Phi}(\tilde{\boldsymbol{x}}) = \frac{1}{P}\sum_{p=0}^{P-1}\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}}). \tag{13}$$

Let us denote the Jacobian of $F_{\boldsymbol{x}}$ at $\boldsymbol{w}$ by $\mathbf{J}_{F_{\boldsymbol{x}}}(\boldsymbol{w})$. For each $p \in \mathscr{P}$, we apply the chain rule:

$$\boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\phi_p} \circ F_{\boldsymbol{x}})(\boldsymbol{w}) = \boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}})^\top \mathbf{J}_{F_{\boldsymbol{x}}}(\boldsymbol{w}). \tag{14}$$

Plugging the identity above into Equation 13 yields:

$$\begin{aligned}\boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\Phi} \circ F_{\boldsymbol{x}})(\boldsymbol{w}) &= \frac{1}{P}\sum_{p=0}^{P-1}\left(\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}})^\top \mathbf{J}_{F_{\boldsymbol{x}}}(\boldsymbol{w})\right) \\ &= \left(\frac{1}{P}\sum_{p=0}^{P-1}\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}})\right)^\top \mathbf{J}_{F_{\boldsymbol{x}}}(\boldsymbol{w}),\end{aligned} \tag{15}$$

where the latter equation holds by associativity of matrix multiplication.

We now compute the expected value of $\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_z}(\tilde{\boldsymbol{x}})$ for $z \sim \mathscr{U}_P$, i.e., a uniform distribution over $\mathscr{P}$:

$$\mathbb{E}_{z\sim\mathscr{U}_P}\left[\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_z}(\tilde{\boldsymbol{x}})\right] = \frac{1}{P}\sum_{p=0}^{P-1}\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_p}(\tilde{\boldsymbol{x}}). \tag{16}$$

We recognize the row vector on the right-hand side of Equation 15. Thus:

$$\begin{aligned}\boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\Phi} \circ F_{\boldsymbol{x}})(\boldsymbol{w}) &= \mathbb{E}_{z\sim\mathscr{U}_P}\left[\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_z}(\tilde{\boldsymbol{x}})\right]^\top \mathbf{J}_{F_{\boldsymbol{x}}}(\boldsymbol{w}) \\ &= \mathbb{E}_{z\sim\mathscr{U}_P}\left[\boldsymbol{\nabla}\mathscr{L}_{\boldsymbol{x}}^{\phi_z}(\tilde{\boldsymbol{x}})^\top \mathbf{J}_{F_{\boldsymbol{x}}}(\boldsymbol{w})\right],\end{aligned} \tag{17}$$

where the latter equation holds by linearity of the expected value. Finally, we use the reverse form of the chain rule in Equation 14 to identify the expected SCRAPL gradient:

$$\boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\Phi} \circ F_{\boldsymbol{x}})(\boldsymbol{w}) = \mathbb{E}_{z\sim\mathscr{U}_P}\left[\boldsymbol{\nabla}(\mathscr{L}_{\boldsymbol{x}}^{\phi_z} \circ F_{\boldsymbol{x}})(\boldsymbol{w})\right] \tag{18}$$

concluding the proof.

B SCRAPL ALGORITHM

Algorithm 1 "**Sc**attering transform with **Ra**ndom **P**aths for machine **L**earning" (SCRAPL). The pseudo-code below describes SCRAPL with a batch size equal to one, without loss of generality.

Require: $\Phi = (\phi_p)_0^{P-1}$: Scattering transform (ST) with P paths
Require: π: Categorical distribution over the path set $\mathscr{P} = \{0, \ldots, P-1\}$
Require: F: Autoencoder with trainable parameters $\boldsymbol{w}$
Require: $\boldsymbol{w}$: Neural network weights at initialization
Require: $\beta_1, \beta_2, \varepsilon$: Adam hyperparameters
Require: $(\alpha_k)_0^{K-1}$: Learning rate schedule

$\Gamma \leftarrow \emptyset$
for p in $\{0, \ldots, P-1\}$ **do**
  $\tau_p \leftarrow 0$
  $\boldsymbol{m}_p \leftarrow \boldsymbol{0}$
  $\boldsymbol{v}_p \leftarrow \boldsymbol{0}$
  $\hat{\boldsymbol{g}}_p \leftarrow \boldsymbol{0}$
end for
for k in $\{0, \ldots, K-1\}$ **do**
  $n \leftarrow$ draw an integer uniformly at random in $\{0, \ldots, N-1\}$
  $p \leftarrow$ draw an integer at random in $\{0, \ldots, P-1\}$ according to $\boldsymbol{\pi}$ {Stochastic approx.}
  $\mathscr{L}(\boldsymbol{w}) \leftarrow P \|\phi_p(\boldsymbol{x}_n) - (\phi_p \circ F_{\boldsymbol{w}})(\boldsymbol{x}_n)\|_2^2$
  $\boldsymbol{g} \leftarrow \boldsymbol{\nabla} \mathscr{L}(\boldsymbol{w})$
  $\boldsymbol{m}_p \leftarrow \beta_1^{(k-\tau_p)/P} \boldsymbol{m}_p + (1 - \beta_1^{(k-\tau_p)/P}) \boldsymbol{g}$
  $\boldsymbol{v}_p \leftarrow \beta_2^{(k-\tau_p)/P} \boldsymbol{v}_p + (1 - \beta_2^{(k-\tau_p)/P}) (\boldsymbol{g} \odot \boldsymbol{g})$
  $\hat{\boldsymbol{m}} \leftarrow \boldsymbol{m}_p / (1 - \beta_1^{k/P})$ {$\mathscr{P}$-Adam}
  $\hat{\boldsymbol{v}} \leftarrow \boldsymbol{v}_p / (1 - \beta_2^{k/P})$
  $\boldsymbol{g}_{\text{current}} \leftarrow \hat{\boldsymbol{m}} / \sqrt{\varepsilon + \hat{\boldsymbol{v}}}$
  $\tau_p \leftarrow k$
  $\boldsymbol{g}_{\text{avg}} \leftarrow \frac{1}{\max(1, \operatorname{card} \Gamma)} \sum_{\gamma \in \Gamma} \hat{\boldsymbol{g}}_\gamma$
  $\boldsymbol{g}_{\text{SAGA}} \leftarrow \boldsymbol{g}_{\text{current}} - \hat{\boldsymbol{g}}_p + \boldsymbol{g}_{\text{avg}}$ {$\mathscr{P}$-SAGA}
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha_k \boldsymbol{g}_{\text{SAGA}}$
  $\hat{\boldsymbol{g}}_p \leftarrow \boldsymbol{g}_{\text{current}}$
  $\Gamma \leftarrow \Gamma \cup \{p\}$
end for
return $\boldsymbol{w}$

C Additional Granular Synth Evaluation Results

Table 5: Loss function benchmark results for one optimization step (forward + backward, 32768 samples of audio, batch size 4, 1 thread, single precision, 1 NVIDIA RTX A5000 GPU, CUDA 12.4, PyTorch 2.8.0). SCRAPL paths are benchmarked individually and then aggregated across all paths using the median for time and interquartile range (IQR) and maximum for memory usage.

Method	**Median Time (ms)** ↓	**IQR (ms)** ↓	**Max. Memory (MB)** ↓
JTFS	1730	23.9	12 967
SCRAPL	89.8	3.62	2503
MSS Linear	26.3	1.12	694
MSS Log + Linear	19.1	0.696	702
MSS Revisited	**17.0**	**0.210**	**663**
MSS Random	24.7	5.81	706
MS-CLAP	75.6	1.69	2032
PANNs Wavegram-Logmel	29.3	5.92	1360
P-Loss ($\theta_{\text{synth}} \in \mathbb{R}^2$)	0.516	0.108	625

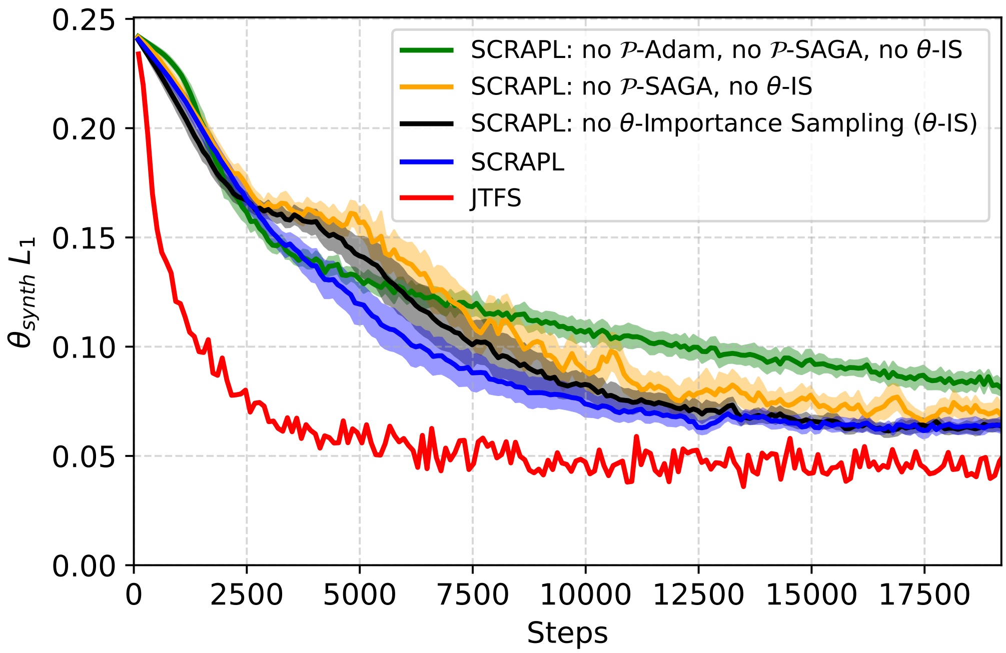

Figure 3: Validation convergence graphs of SCRAPL ablations and the JTFS for the unsupervised granular synth sound matching task. Shaded areas are 95% confidence intervals for 20 training runs using different random seeds. Due to computational limitations, the JTFS method is only evaluated once.

D ADDITIONAL CHIRPLET SYNTH EVALUATION RESULTS

Table 6: Convergence rate (CR) and steps for SCRAPL with and without the θ-importance sampling initialization heuristic on unsupervised sound matching of four different AM / FM chirplet synths. Convergence is defined as $L_1 < 100$ ‰ for θ_{AM} or θ_{FM}. Uncertainties are 95% confidence intervals for 20 training runs using different random seeds.

Sampling Method	**Synth Configuration**		θ_{AM}		θ_{FM}	
	θ_{AM} (Hz)	θ_{FM} (oct/s)	CR ↑	Conv. Steps ↓	CR ↑	Conv. Steps ↓
Uniform	1.0 – 2.0	0.5 – 1.0	60%	3944 ± 342	45%	4064 ± 372
θ-IS	1.0 – 2.0	0.5 – 1.0	**100%**	**2002 ± 324**	**100%**	**3134 ± 492**
Uniform	1.0 – 2.0	2.0 – 4.0	**100%**	2203 ± 135	**100%**	1536 ± 194
θ-IS	1.0 – 2.0	2.0 – 4.0	**100%**	**1099 ± 173**	**100%**	**768 ± 118**
Uniform	2.8 – 8.4	2.0 – 4.0	95%	3254 ± 250	0%	N/A
θ-IS	2.8 – 8.4	2.0 – 4.0	**100%**	**1925 ± 165**	**100%**	**2966 ± 210**
Uniform	2.8 – 8.4	4.0 – 12.0	**100%**	3096 ± 334	**95%**	3208 ± 235
θ-IS	2.8 – 8.4	4.0 – 12.0	95%	**2253 ± 218**	**95%**	**2178 ± 173**

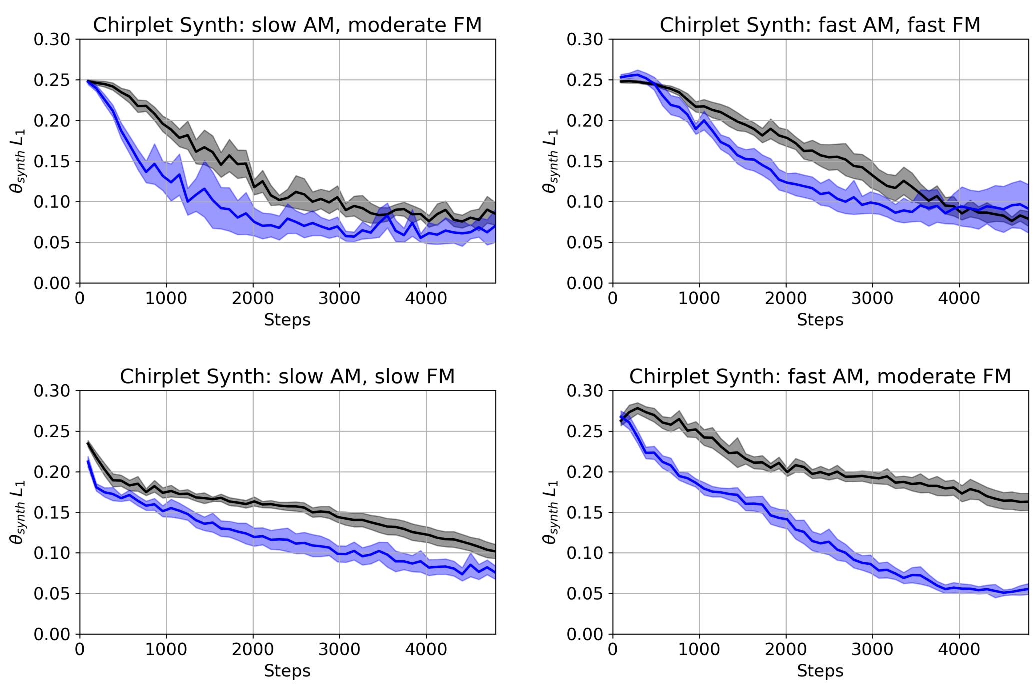

Figure 4: SCRAPL θ_{synth} L_1 validation values during training for four different AM / FM chirplet synths. Blue is using the θ-importance sampling initialization heuristic, and black is using uniform sampling. Shaded areas are 95% confidence intervals for 20 training runs using different random seeds.

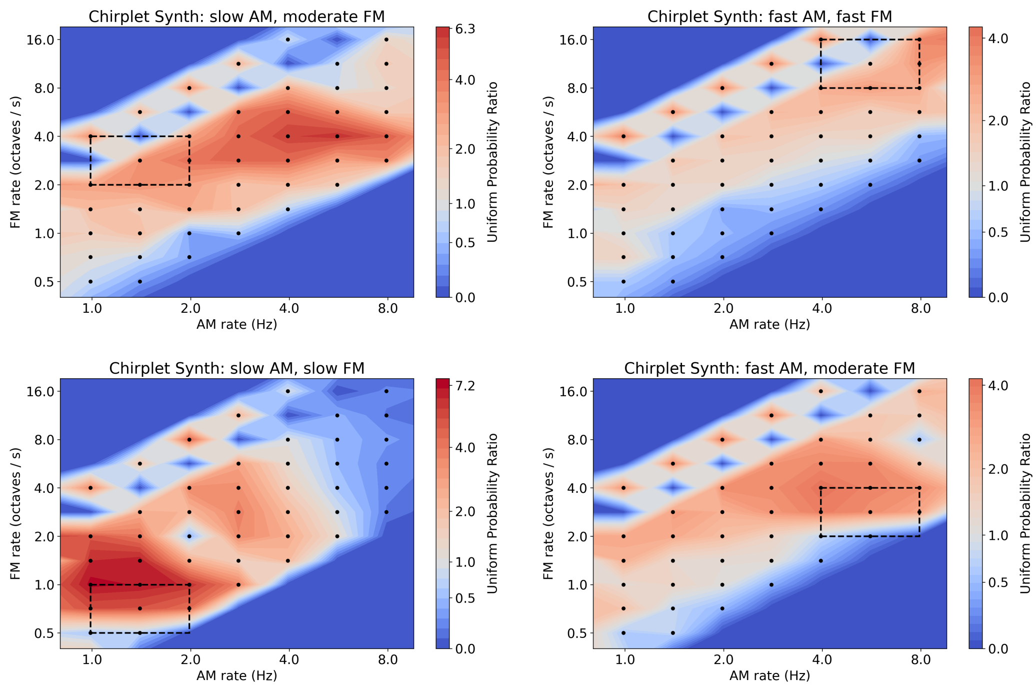

Figure 5: SCRAPL ($J = 12, Q_1 = 8, Q_2 = 2, J_{\mathrm{fr}} = 3, Q_{\mathrm{fr}} = 2, N_{\mathrm{paths}} = 315$) path θ-importance sampling probabilities for four different AM / FM chirplet synths calculated from 1 batch of 32 log-uniformly randomly sampled θ_{synth} values. Black dots are individual path (wavelet) AM / FM center frequency locations and each dashed rectangle is the θ_{synth} range for each synth configuration. A uniform probability ratio of 1.0 means a path is sampled with probability $\frac{1}{P}$.

E Additional Roland TR-808 Evaluation Results

Table 7: Drum transient evaluation results for the unsupervised Roland TR-808 DDSP synth sound matching task. Uncertainties are 95% confidence intervals for 40 training runs using different random seeds and dataset splits. Due to computational limitations, the JTFS method is only trained and evaluated for 4 random seeds.

Method	Loudness $L_1 \downarrow$		Spectral Centroid $L_1 \downarrow$		Spectral Flatness $L_1 \downarrow$	
	Micro	Meso	Micro	Meso	Micro	Meso
JTFS	**137** $\pm$ 10	**158** $\pm$ 20	**859** $\pm$ 68	**819** $\pm$ 96	1200 $\pm$ 87	**1090** $\pm$ 110
SCRAPL (no LS)	389 $\pm$ 41	460 $\pm$ 60	1020 $\pm$ 100	992 $\pm$ 110	1800 $\pm$ 170	1990 $\pm$ 180
SCRAPL	374 $\pm$ 39	377 $\pm$ 32	1000 $\pm$ 120	1080 $\pm$ 120	1750 $\pm$ 270	1820 $\pm$ 220
MSS Lin.	381 $\pm$ 12	2510 $\pm$ 480	902 $\pm$ 27	2350 $\pm$ 310	962 $\pm$ 43	3620 $\pm$ 680
MSS L+L	492 $\pm$ 44	1080 $\pm$ 91	928 $\pm$ 65	1380 $\pm$ 76	**916** $\pm$ 50	1320 $\pm$ 150
MSS Rev.	330 $\pm$ 21	808 $\pm$ 40	1070 $\pm$ 49	1540 $\pm$ 53	1390 $\pm$ 62	2640 $\pm$ 96
MSS Rand.	584 $\pm$ 75	1030 $\pm$ 89	1200 $\pm$ 100	1350 $\pm$ 99	1690 $\pm$ 120	1950 $\pm$ 140

Table 8: Drum decay evaluation results for the unsupervised Roland TR-808 DDSP synth sound matching task. Uncertainties are 95% confidence intervals for 40 training runs using different random seeds and dataset splits. Due to computational limitations, the JTFS method is only trained and evaluated for 4 random seeds.

Method	Loudness $L_1 \downarrow$		Spectral Centroid $L_1 \downarrow$		Spectral Flatness $L_1 \downarrow$	
	Micro	Meso	Micro	Meso	Micro	Meso
JTFS	315 $\pm$ 22	**355** $\pm$ 110	614 $\pm$ 51	617 $\pm$ 71	527 $\pm$ 31	718 $\pm$ 190
SCRAPL (no LS)	1810 $\pm$ 190	2210 $\pm$ 210	1530 $\pm$ 170	1860 $\pm$ 170	2620 $\pm$ 310	3300 $\pm$ 370
SCRAPL	1810 $\pm$ 160	1740 $\pm$ 170	1490 $\pm$ 120	1470 $\pm$ 140	2540 $\pm$ 290	2480 $\pm$ 290
MSS Lin.	357 $\pm$ 12	1120 $\pm$ 260	654 $\pm$ 18	1110 $\pm$ 160	**472** $\pm$ 17	1500 $\pm$ 350
MSS L+L	389 $\pm$ 42	466 $\pm$ 45	**563** $\pm$ 22	**597** $\pm$ 24	565 $\pm$ 29	**644** $\pm$ 51
MSS Rev.	**279** $\pm$ 12	494 $\pm$ 22	589 $\pm$ 21	801 $\pm$ 29	552 $\pm$ 22	846 $\pm$ 29
MSS Rand.	453 $\pm$ 21	485 $\pm$ 24	660 $\pm$ 27	640 $\pm$ 35	594 $\pm$ 30	658 $\pm$ 33

F Experiment Training Details and Hyperparameters

Table 9: Unsupervised granular synth sound matching task hyperparameters.

Category	Hyperparameter Name	Value
Data	N (# of examples)	5120
	train / val / test split	60% / 20% / 20%
Encoder	# of parameters	604 K
	CQT # of octaves	5
	CQT bins / octave	12
	CQT hop length	256
	CQT postprocessing	`log1p`
	CNN # of conv. blocks	5
	CNN kernel size	(3, 3)
	CNN channels / conv. block	128
	CNN embedding dim.	64
	CNN dense layer dropout prob.	0.5
Decoder (Synth)	$dim(\theta_{\text{synth}})$	2
	sampling rate	8192 Hz
	# of samples	32768
	max. # of grains	64
	grain # of samples	4096
	min. grain pitch	256 Hz
	max. grain pitch	2048 Hz
SCRAPL & JTFS	J	12
	Q_1	8
	Q_2	2
	J_{fr}	3
	Q_{fr}	2
	T	4096
	F	8
	ρ	identity function
	P (# of paths)	315
θ-Importance Sampling	N_{IS} (# of examples)	320
	λ_{max} # of deflated power iterations	20
Training	# of random seed training runs	20
	epochs	200
	batch size	32
	starting learning rate	1×10^{-5}
	learning rate scheduler	none
	Adam β_1	0.9
	Adam β_2	0.999
	weight decay	0.01

Table 10: Unsupervised AM/FM chirplet synth sound matching task hyperparameters.

Category	**Hyperparameter Name**	**Value**
Data	N (# of examples)	5120
	train / val / test split	60% / 20% / 20%
Encoder	# of parameters	604 K
	CQT # of octaves	5
	CQT bins / octave	12
	CQT hop length	256
	CQT postprocessing	`log1p`
	CNN # of conv. blocks	5
	CNN kernel size	(3, 3)
	CNN channels / conv. block	128
	CNN embedding dim.	64
	CNN dense layer dropout prob.	0.5
Decoder (Synth)	$dim(\theta_{\text{synth}})$	2
	sampling rate	8192 Hz
	# of samples	32768
	chirplet center frequency	512 Hz
	chirplet bandwidth	2 octaves
	min. time shift	-2048 samples
	max. time shift	+2048 samples
SCRAPL & JTFS	J	12
	Q_1	8
	Q_2	2
	J_{fr}	3
	Q_{fr}	2
	T	4096
	F	8
	ρ	identity function
	P (# of paths)	315
θ-Importance Sampling	N_{IS} (# of examples)	32
	$\lambda_{\max}$ # of deflated power iterations	20
Training	# of random seed training runs	20
	epochs	50
	batch size	32
	starting learning rate	1×10^{-4}
	learning rate scheduler	none
	Adam β_1	0.9
	Adam β_2	0.999
	weight decay	0.01

Table 11: Unsupervised Roland TR-808 synth sound matching task hyperparameters.

Category	**Hyperparameter Name**	**Value**
Data	N (# of examples)	681
	$N_{\text{bass drum}}$	215
	N_{snare}	240
	N_{tom}	189
	$N_{\text{hi-hat}}$	37
	N_{train}	425
	N_{val}	128
	N_{test}	128
Encoder	# of parameters	724 K
	CQT # of octaves	9
	CQT bins / octave	12
	CQT hop length	256
	CQT postprocessing	`log1p`
	CNN # of conv. blocks	5
	CNN kernel size	(3, 3)
	CNN channels / conv. block	128
	CNN embedding dim.	128
	CNN dense layer dropout prob.	0.25
Decoder (Synth)	$dim(\theta_{\text{synth}})$	14
	sampling rate	44100 Hz
	# of samples	44100
	min. time shift	-2048 samples
	max. time shift	+2048 samples
SCRAPL & JTFS	J	12
	Q_1	8
	Q_2	2
	J_{fr}	5
	Q_{fr}	2
	T	2048
	F	1
	ρ	`log1p`
	P (# of paths)	483
θ-Importance Sampling	N_{IS} (# of examples)	16
	$\lambda_{\max}$ # of deflated power iterations	20
Training	# of random seed training runs	40
	epochs	50
	batch size	8
	starting learning rate	1×10^{-5}
	learning rate scheduler	linearly decreasing until 1×10^{-4}
	Adam β_1	0.9
	Adam β_2	0.999
	weight decay	0.01