ICLR 2026

SCRAPL: Scattering with Random Paths for Machine Learning

Christopher Mitcheltree[1] Vincent Lostanlen[2] Emmanouil Benetos[1] Mathieu Lagrange[2]

[1]Centre for Digital Music, Queen Mary University of London, UK

[2]Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, France

```
pip install scrapl-loss
christhetree.github.io/scrapl
```

One Sentence Summary

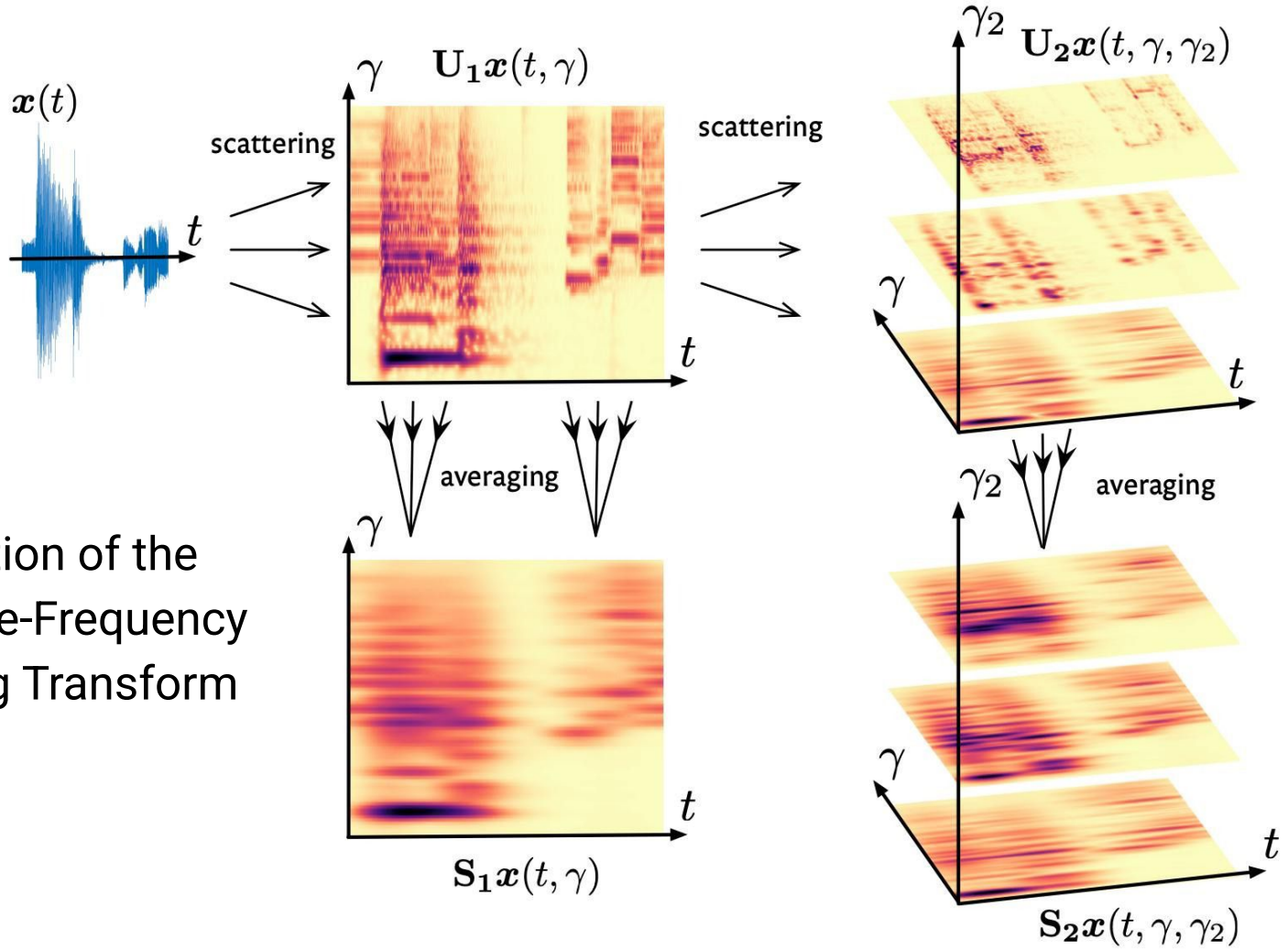
SCRAPL is an algorithm for efficient evaluation of multivariable wavelet scattering transforms, so they can serve as differentiable loss functions for neural network training.

```
pip install scrapl-loss
```

`github.com/christhetree/scrapl`

Background

Visualization of the Joint Time-Frequency Scattering Transform (JTFS)

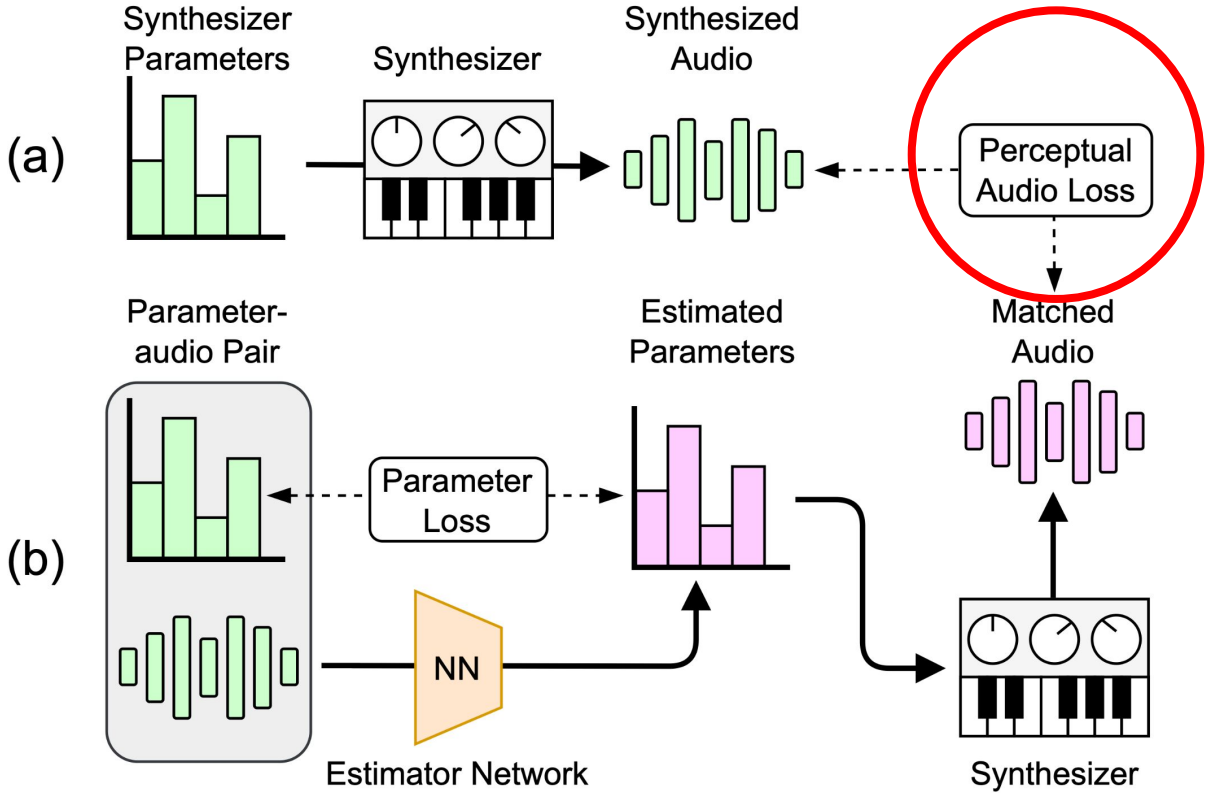


Why wavelet scattering transforms?

1. Physical interpretability
2. Biological plausibility
3. Mathematical understanding
4. Learning with limited data
5. Artistic creation

`kymat.io/ismir23-tutorial/`

Sound Matching

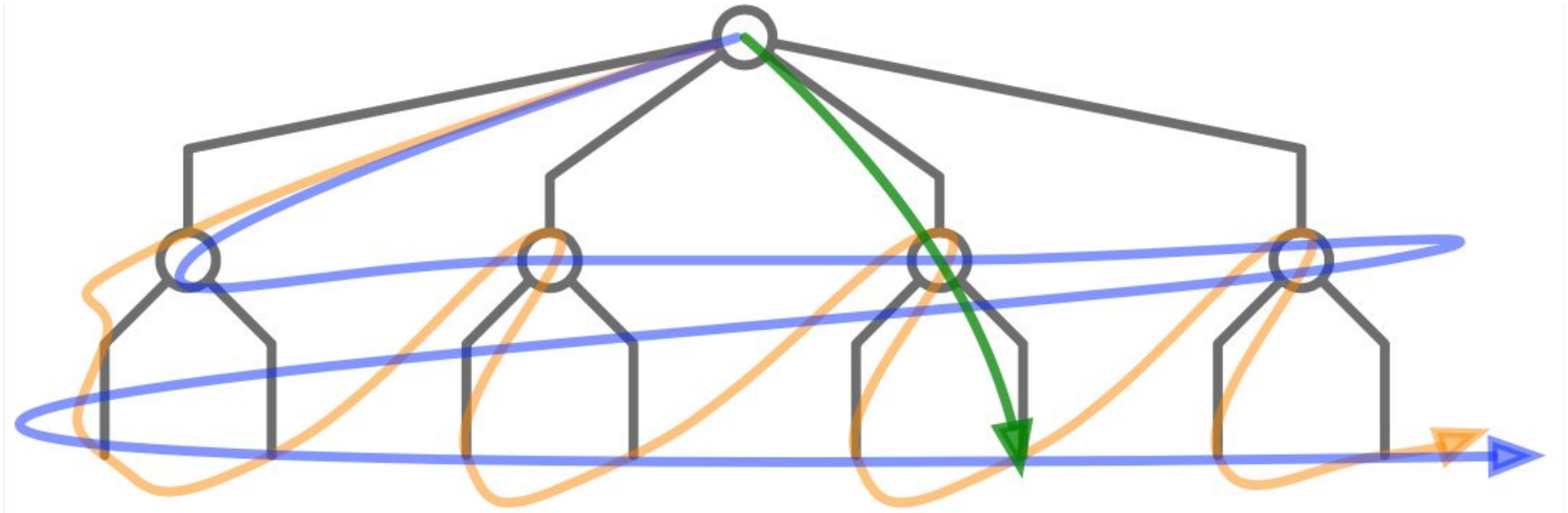


SCRAPL:
Scattering with Random Paths for Machine Learning

Scattering with Random Paths

- Wavelet scattering transforms are very computationally expensive and require a lot of memory due to their many paths
- Not all paths (wavelets) are informative for the task at hand
- **Why not sample paths uniformly to approximate a scattering transform?**
- ~10-1000x speedup and memory reduction
- Should mathematically converge to the full-tree scattering transform (i.e., this approach is unbiased, see Appendix A in the paper for a proof)

SCRAPL vs. Full-tree Scattering Transforms



Breadth-first search (blue), depth-first search (orange), and random access (green).

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: $(\alpha_k)_1^K$: Learning rate schedule

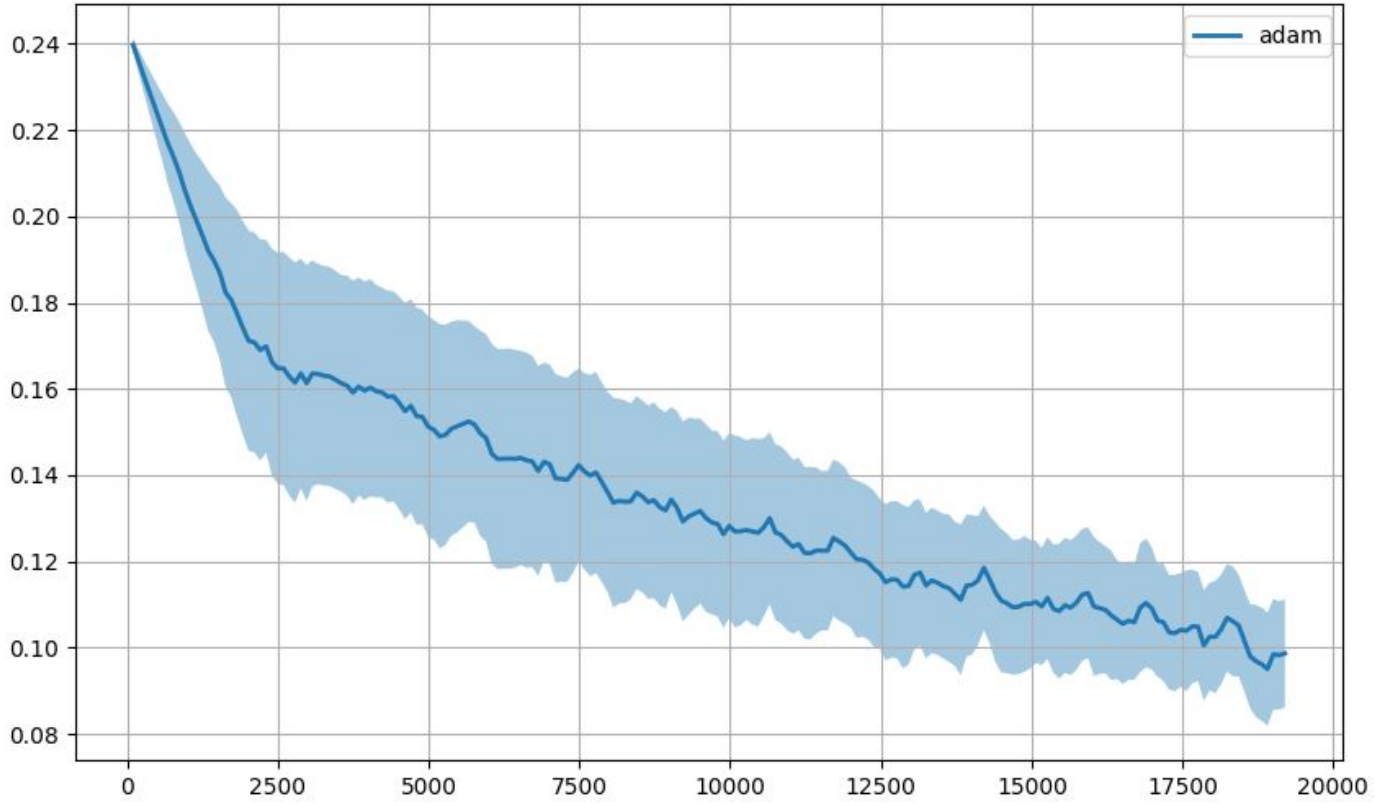
Algorithm 1 "**Sc**attering transform with **Ra**ndom **P**aths for machine **L**earning" (SCRAPL). The pseudo-code below describes SCRAPL with a batch size equal to one, without loss of generality.

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: $(\alpha_k)_1^K$: 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

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: $(\alpha_k)_1^K$: 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 $\{1, \ldots, K\}$ **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 π
    $\mathscr{L}(\boldsymbol{w}) \leftarrow P \left\| \phi_p(\boldsymbol{x}_n) - (\phi_p \circ F_{\boldsymbol{w}})(\boldsymbol{x}_n) \right\|_2^2$ {Stochastic approx.}
    $\boldsymbol{g} \leftarrow \boldsymbol{\nabla} \mathscr{L}(\boldsymbol{w})$

SCRAPL (vanilla)



for k in $\{1, \ldots, K\}$ **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 π
$\mathscr{L}(\boldsymbol{w}) \leftarrow P \left\| \phi_p(\boldsymbol{x}_n) - (\phi_p \circ F_{\boldsymbol{w}})(\boldsymbol{x}_n) \right\|_2^2$ {Stochastic approx.}
$\boldsymbol{g} \leftarrow \boldsymbol{\nabla} \mathscr{L}(\boldsymbol{w})$

$\boldsymbol{m}_p \leftarrow \beta_1^{(k-\tau_p)/P} \boldsymbol{m}_p + (1 - \beta_1^{(k-\tau_p)/P}) \boldsymbol{g}$
$\boldsymbol{v}_p \leftarrow \beta_2^{(k-\tau_p)/P} \boldsymbol{v}_p + (1 - \beta_2^{(k-\tau_p)/P}) (\boldsymbol{g} \odot \boldsymbol{g})$
$\hat{\boldsymbol{m}} \leftarrow \boldsymbol{m}_p / (1 - \beta_1^{k/P})$ {$\mathscr{P}$-Adam}
$\hat{\boldsymbol{v}} \leftarrow \boldsymbol{v}_p / (1 - \beta_2^{k/P})$
$\boldsymbol{g}_{\text{current}} \leftarrow \hat{\boldsymbol{m}} / \sqrt{\varepsilon + \hat{\boldsymbol{v}}}$
$\tau_p \leftarrow k$

for k in $\{1,\dots,K\}$ **do**

$n \leftarrow$ draw an integer uniformly at random in $\{0,\dots,N-1\}$
$p \leftarrow$ draw an integer at random in $\{0,\dots,P-1\}$ according to π
$\mathscr{L}(\boldsymbol{w}) \leftarrow P\left\|\boldsymbol{\phi}_p(\boldsymbol{x}_n) - (\boldsymbol{\phi}_p \circ F_{\boldsymbol{w}})(\boldsymbol{x}_n)\right\|_2^2$ {Stochastic approx.}
$\boldsymbol{g} \leftarrow \boldsymbol{\nabla}\mathscr{L}(\boldsymbol{w})$

$\boldsymbol{m}_p \leftarrow \beta_1^{(k-\tau_p)/P}\boldsymbol{m}_p + (1-\beta_1^{(k-\tau_p)/P})\boldsymbol{g}$
$\boldsymbol{v}_p \leftarrow \beta_2^{(k-\tau_p)/P}\boldsymbol{v}_p + (1-\beta_2^{(k-\tau_p)/P})(\boldsymbol{g}\odot\boldsymbol{g})$
$\hat{\boldsymbol{m}} \leftarrow \boldsymbol{m}_p/(1-\beta_1^{k/P})$ {$\mathscr{P}$-Adam}
$\hat{\boldsymbol{v}} \leftarrow \boldsymbol{v}_p/(1-\beta_2^{k/P})$
$\boldsymbol{g}_{\text{current}} \leftarrow \hat{\boldsymbol{m}}/\sqrt{\boldsymbol{\varepsilon}+\hat{\boldsymbol{v}}}$
$\tau_p \leftarrow k$

$\boldsymbol{g}_{\text{avg}} \leftarrow \dfrac{1}{\max(1,\operatorname{card}\Gamma)}\sum_{\gamma\in\Gamma}\hat{\boldsymbol{g}}_\gamma$
$\boldsymbol{g}_{\text{SAGA}} \leftarrow \boldsymbol{g}_{\text{current}} - \hat{\boldsymbol{g}}_p + \boldsymbol{g}_{\text{avg}}$ {$\mathscr{P}$-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

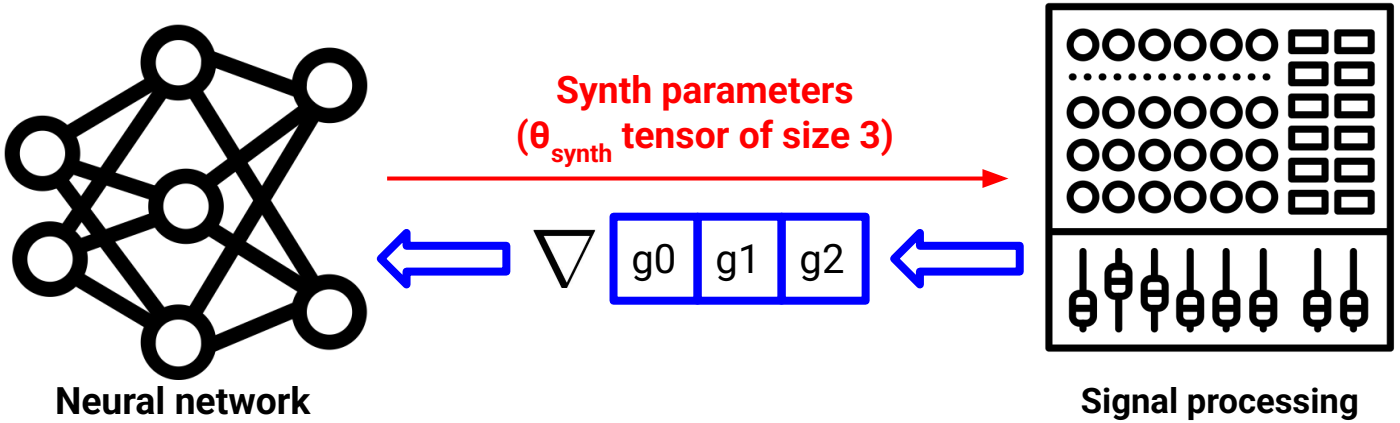
Biased SCRAPL:
θ-Importance Sampling

θ-Importance Sampling Intuition

- Not all paths (wavelets) are informative for the task at hand
- **Why sample paths uniformly then?**
- There exists an optimal sampling distribution for each optimization task
- Need a metric to calculate how “informative” a path is
  - Needs to be general and task agnostic
  - Cannot be too expensive to compute
- **Sample “informative” paths often and “uninformative” paths rarely**

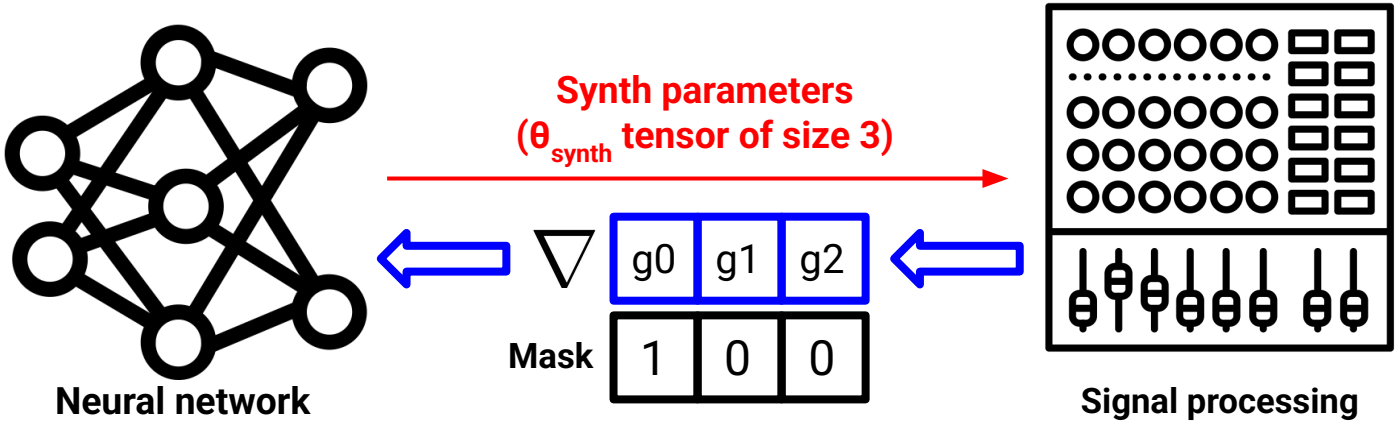
θ-Importance Sampling

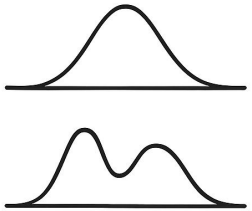
1. Before training, measure each synth parameter's sensitivity to each path on a subset of training datapoints
2. Combine these measured sensitivities into a proportional categorical probability distribution over paths $\mathscr{P}$
3. Take the average of these distributions



θ-Importance Sampling

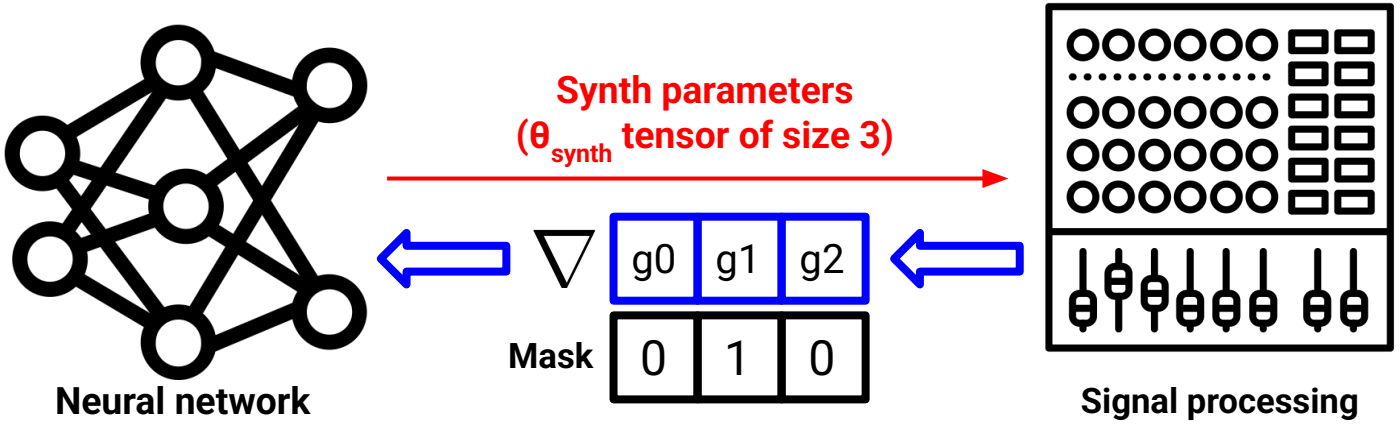
1. Before training, measure each synth parameter's sensitivity to each path on a subset of training datapoints
2. Combine these measured sensitivities into a proportional categorical probability distribution over paths $\mathscr{P}$
3. Take the average of these distributions





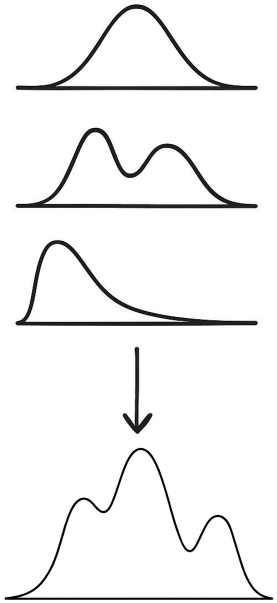
θ-Importance Sampling

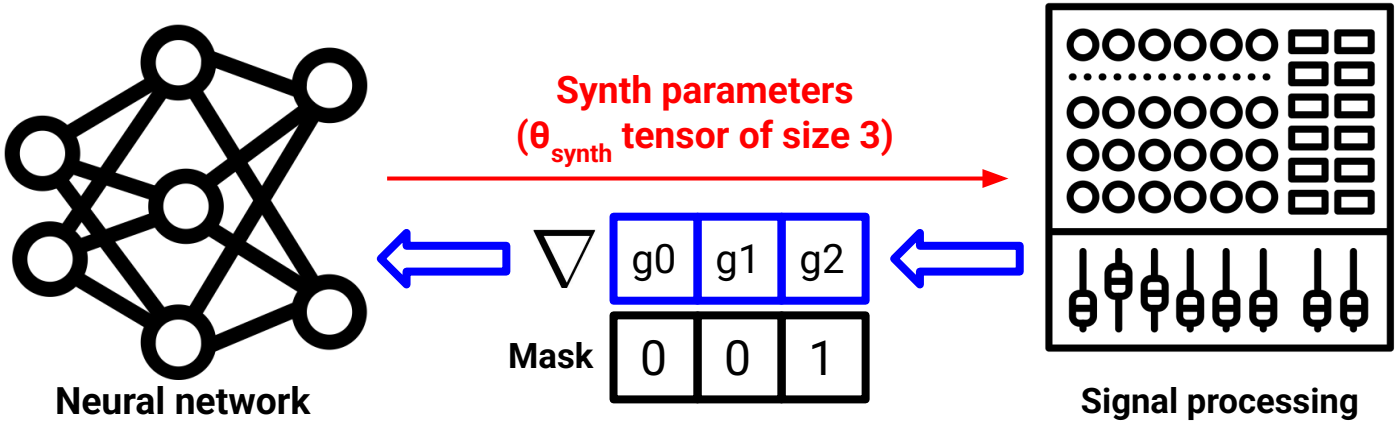
1. Before training, measure each synth parameter's sensitivity to each path on a subset of training datapoints
2. Combine these measured sensitivities into a proportional categorical probability distribution over paths $\mathscr{P}$
3. Take the average of these distributions



θ-Importance Sampling

1. Before training, measure each synth parameter's sensitivity to each path on a subset of training datapoints
2. Combine these measured sensitivities into a proportional categorical probability distribution over paths $\mathscr{P}$
3. Take the average of these distributions





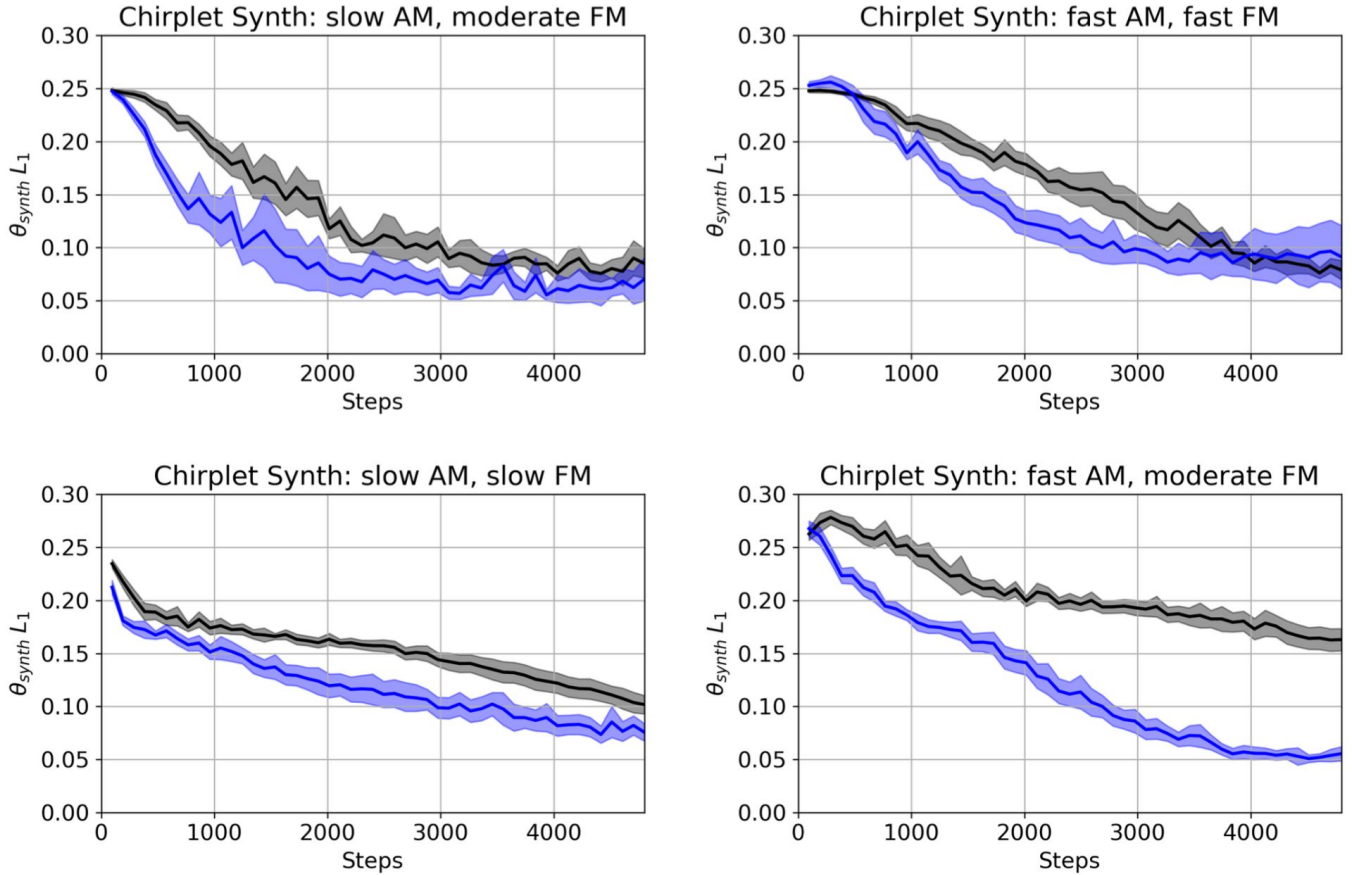
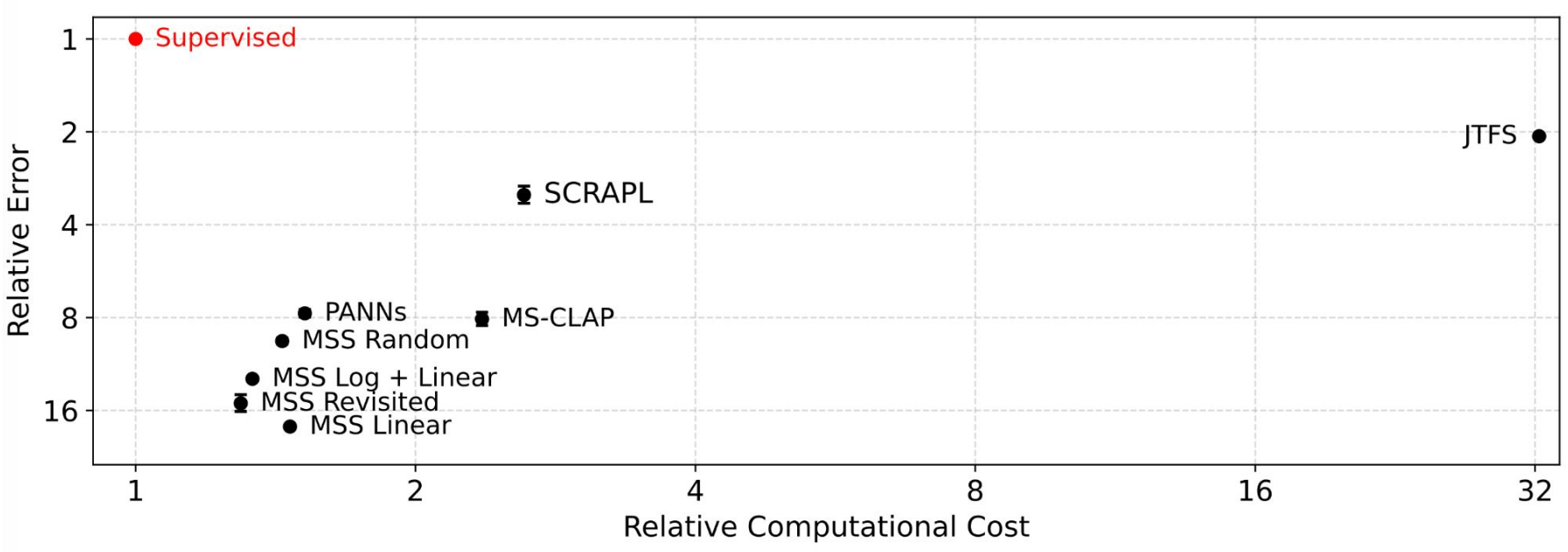


Figure 5: SCRAPL θ_{synth} L_1 validation values during training for four different AM / FM chirplet synths with two continuous θ_{synth} parameters: θ_{AM} and θ_{FM} (more details in Section 4.3). Blue is using the θ-importance sampling initialization heuristic, and black is using uniform sampling. Shaded areas are 95% CI for 20 training runs using different random seeds.

SCRAPL vs. Other Loss Functions



https://www.roland.com/global/promos/roland_tr-808/

Quantitative Sound Matching Evaluation

Table 4: Audio distance evaluation results for the unsupervised Roland TR-808 DDSP synth sound matching task with 14 continuous $\boldsymbol{\theta}_{\text{synth}}$ parameters (more details in Section 4.4). Uncertainties are 95% CI 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 ↓**		**FAD (EnCodec) ↓**	
	Micro	Meso	Micro	Meso	Micro	Meso
JTFS	617 ±46	622 ±45	**490** ±28	**523** ±17	**0.781** ± 0.069	**1.04** ±0.15
SCRAPL (no θ-IS)	862 ±36	944 ±48	1140 ±48	1250 ±51	2.75 ± 0.39	2.85 ±0.38
SCRAPL	857 ±42	879 ±42	1050 ±50	1110 ±52	2.43 ± 0.22	2.42 ±0.22
MSS Lin.	611 ±15	724 ±37	779 ±31	1470 ±83	1.22 ± 0.082	3.33 ±0.46
MSS L+L	**596** ±19	**615** ±18	1260 ±58	1390 ±49	2.14 ± 0.39	3.01 ±0.40
MSS Rev.	637 ±16	797 ±20	870 ±23	1250 ±27	2.02 ± 0.37	2.21 ±0.34
MSS Rand.	682 ±25	700 ±26	1410 ±87	1500 ±59	7.03 ± 2.2	6.65 ±1.7

Conclusion

Contributions

1. Stochastic approximation of scattering transforms
2. Path-wise adaptive moment estimation ($\mathscr{P}$-Adam)
3. Path-wise stochastic average gradient with acceleration ($\mathscr{P}$-SAGA)
4. θ-importance sampling parallelizable initialization heuristic

SCRAPL for the JTFS in PyTorch is available now:

- `pip install scrapl-loss`
- `christhetree.github.io/scrapl`