AUDIO PROCESSOR PARAMETERS: ESTIMATING DISTRIBUTIONS INSTEAD OF DETERMINISTIC VALUES

Côme Peladeau[*] [♯], *Dominique Fourer*[♭] *and Geoffroy Peeters*[♯]

[♯] LTCI, Télécom-Paris, Institut Polytechnique de Paris, Palaiseau, France
[♭] IBISC, Univ. Évry Paris Saclay, Évry-Courcouronnes, France
come.peladeau@telecom-paris.fr

ABSTRACT

Audio effects and sound synthesizers are widely used processors in popular music. Their parameters control the quality of the output sound. Multiple combinations of parameters can lead to the same sound. While recent approaches have been proposed to estimate these parameters given only the output sound, those are deterministic, i.e. they only estimate a single solution among the many possible parameter configurations. In this work, we propose to model the parameters as probability distributions instead of deterministic values. To learn the distributions, we optimize two objectives: (1) we minimize the reconstruction error between the ground truth output sound and the one generated using the estimated parameters, as is it usually done, but also (2) we maximize the parameter diversity, using entropy. We evaluate our approach through two numerical audio experiments to show its effectiveness. These results show how our approach effectively outputs multiple combinations of parameters to match one sound.

1. INTRODUCTION

Audio processors play a central part in music production [1]. They fall into two categories: synthesizers, which generate sounds, and audio effects, which process an existing sound. For both, parameters allow to control the resulting output sound. Learning to use these processors thus comes down to learn how to parametrize them adequately.

Popular music is mainly transmitted through recordings, which involve audio processors. Their analysis is therefore of great interest not only in itself, but also towards a broader understanding of musical practices [2].

Audio effects and sound synthesizers are in most cases hard to use and usually require a long learning process. Therefore a set of research aims at helping users to perform audio processors related tasks, whether it is matching a sound with a synthesizer [3, 4, 5, 6], automatic mixing [7, 8], mastering [9], reverse-engineering a mix [10, 11] or production style transfer [12, 13].

As noted in recent research [14], the various parameters do not have the same influence on the output. Moreover, it is common that multiple different parameters combinations result in approximately the same sound—increasing the amplitude of low frequencies is similar to attenuating high frequencies, up to a level factor. However, to the best of our knowledge, this has never been studied.

[*] This research was supported by the French ANR project AQUA-RIUS (ANR-22-CE23-022).

Copyright: © 2025 Côme Peladeau et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, adaptation, and reproduction in any medium, provided the original author and source are credited.

In this paper, we propose a probabilistic framework for the inference of the parameters of audio processors. Rather than estimating deterministic values, we propose to estimate distributions of parameters. These distributions are optimized following two objectives: (1) sampled parameters should reconstruct accurately the target sound and (2) they should be diverse. As a proof of concept, we apply our framework to the blind estimation of audio effects [15, 16] and synthesizer sound matching [3]. Our method allows to sample multiple parameters, all resulting in similar output sounds. We share the code used to perform the experiments[1].

1.1. Paper organization

We list works on estimation of processor parameters using neural networks in Section 2. We then present our approach in Section 3. This approach is evaluated in two experiments: estimation of audio effects parameters (Sec. 4) and estimation of synthesizer parameters (Sec. 5). For the first experiment, we report both some quantitative results (Sec. 4.7.1) and an analysis of the output distribution of our model (Sec. 4.7.2). For the second experiment, we also report quantitative results (Sec. 5.6.1) and a qualitative analysis of the optimized model (Sec. 5.6.2).

2. RELATED WORKS

In the following we denote by $\mathbf{x}$ an unprocessed audio signal, by $\mathbf{y}$ the output signal obtained with audio processors parameters $\mathbf{v}$, and by $\hat{\mathbf{v}}$ the estimated parameters (see also Fig. 1). Estimating $\hat{\mathbf{v}}$ with neural network can be done in two ways.

2.1. Parameter-based approach

The classical approach is based on supervised learning where we synthesize examples $\mathbf{y}$ using its known parameters $\mathbf{v}$ and we train a neural network to reconstruct them by minimizing **parameter loss** $\mathcal{L}(\mathbf{v}, \hat{\mathbf{v}})$ (for instance the mean square error). Esling *et al.* [17] proposed an approach first relying on a audio variational autoencoder (VAE) with normalizing flows (NFs) and then designing a mapping between the VAE latent space and the synthesizer parameters using a dataset of synthesized sounds. Following Esling's work, Le Vaillant [6, 18, 19] took this approach further by improving the reconstruction quality and ensuring that the interpolation between two presets is perceptually smooth.

The same can be done with audio effects with a model trained to estimate the parameters [15], but then the training also requires a set of unprocessed audio signals which are processed during training with known parameters.

[1] github.com/peladeaucome/DAFx_Params_Distrib

2.2. Differentiable digital signal processing

Differentiable digital signal processing (DDSP) is a term coined by Engel *et al.* [3] designating the use of explicit signal processing blocks in deep learning frameworks in order to incorporate domain knowledge directly into the models. It was originally used in a monophonic synthesizer using a harmonic plus noise model [3]. Various common sound synthesis methods have then been implemented, including additive/substractive synthesis [5, 20], frequency modulation (FM) synthesis [4], waveshaping [21] and wavetable [22]. DDSP enables to train models such that the output of the DDSP module $\hat{\mathbf{y}}$ matches the target audio $\mathbf{y}$ (rather than matching the ground truth parameters). Thus, it allows to use datasets without ground truth parameters.

The most popular audio effects have also been implemented as differentiable units. They are usually called differentiable digital audio effects (DDAFx). They include equalization [23, 24, 25], dynamic range compression [12, 26, 27], reverberation [28, 29, 30], phaser [31], non-linear distortion [32, 33].

These DDAFx can be used to estimate audio effects in two ways: non-blind or blind.

Non-blind estimation is the case when we have access to both the input $\mathbf{x}$ and output signals of the effects $\mathbf{y} = f_{\text{Fx}}(\mathbf{x}, \mathbf{v})$, but not to the effects chain f_{Fx} nor its parameters $\mathbf{v}$. Approaches based on DDAFx work in the following way [10, 11]: we design a differentiable effects chain f_{DDSP} and we optimize its parameters $\mathbf{z}$ to minimize an **audio loss** ℓ between $\mathbf{y}$ and $\hat{\mathbf{y}} = f_{\text{DDSP}}(\mathbf{x}, \mathbf{z})$: $\ell(\mathbf{y}, \hat{\mathbf{y}})$. Note that we distinguish $\mathbf{z}$ from $\hat{\mathbf{v}}$ since they don't necessarily have the same number of dimensions (f_{Fx} and f_{DDSP} may have a different implementation), as noted in [16]. Since f_{DDSP} is differentiable, $\mathbf{z}$ can be optimized using a gradient descent algorithm.

Blind estimation is when we have access only to the output $\mathbf{y} = f_{\text{Fx}}(\mathbf{x}, \mathbf{v})$, even though we may use the input $\mathbf{x}$ when designing the approach, for instance during training. In this case, direct optimization of the audio effects parameters is thus impossible. Blind approaches need to optimize a model which "learns" the distribution of unprocessed signals $\mathbf{x}$. It can then estimate which effects have been applied to obtain the processed signal $\mathbf{y}$, which ultimately relies on an estimation of what the unprocessed version is. They are usually based on neural networks optimized in a supervised fashion to minimize the distance between the estimated and the ground truth parameters $\mathcal{L}(\mathbf{v}, \hat{\mathbf{v}})$ [15]. In our previous work [16], we proposed an approach where the ground truth parameters are no longer needed. It relies on DDAFx and optimizes an audio loss $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$. This approach is depicted in Fig. 1.

3. INFERENCE OF PROCESSOR PARAMETERS

In Fig. 1, we illustrate our framework previously proposed in [16] to optimize a deterministic model for estimating processor parameters with DDSP. We consider an unprocessed audio signal $\mathbf{x} \in \mathbb{R}^N$, with N being the signal length in samples. It is processed by several audio effects or sound synthesizers parametrized by $\mathbf{v} \in \mathbb{R}^{d_v}$. This produces a processed audio sample $\mathbf{y} \in \mathbb{R}^N$. Previous approaches [3, 4, 16] focus on optimizing a deterministic model which, given the processed audio signal $\mathbf{y}$, estimates parameters $\mathbf{z} = f_\theta(\mathbf{y}) \in \mathbb{R}^d$ of a differentiable processor (DDSP) that allow to map the unprocessed signal $\mathbf{x}$ to $\mathbf{y}$ such as $\hat{\mathbf{y}} = f_{\text{DDSP}}(\mathbf{x}, \mathbf{z})$.

If the processor is a pure synthesizer, then there is no unprocessed audio signal, i.e. $\mathbf{x} = \mathbf{0}$, and the processor creates a signal from it.

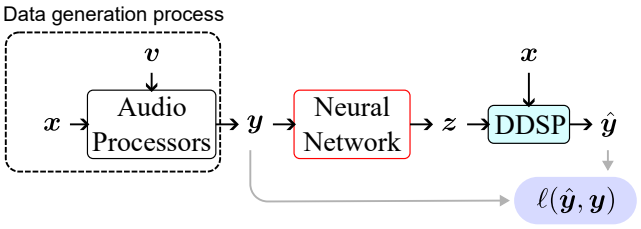


Figure 1: Optimization of a deterministic model to estimate audio processor parameters with DDSP.

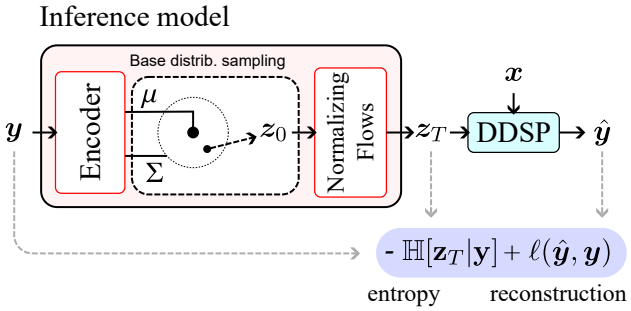


Figure 2: The proposed inference model and its loss function.

In this paper, instead of estimating $\mathbf{z}$, we estimate a distribution $p_\theta(\mathbf{z}|\mathbf{y})$ parametrized by θ. We aim to optimize this model to output diverse $\mathbf{z}$ parameters that allow to match $\mathbf{y}$. The new proposed approach is shown in Fig. 2.

3.1. Formalization

We assume that the true posterior probability of a processor's parameter vector $p(\mathbf{z}|\mathbf{x}, \mathbf{y})$ can be related to a metric ℓ which measures how two sounds $\mathbf{y}$ and $\hat{\mathbf{y}}$ are similar. $\hat{\mathbf{y}}$ is obtained by processing $\mathbf{x}$ with a differentiable processor f_{DDSP}. We define our posterior as the following Boltzmann-Gibbs distribution, as it is usual:

$$p(\mathbf{z}|\mathbf{x}, \mathbf{y}) = a \cdot \exp\left(-\ell\left(f_{\text{DDSP}}(\mathbf{x}; \mathbf{z}), \mathbf{y}\right)\right), \tag{1}$$

with $a \in \mathbb{R}$ being a real-valued normalization constant defined as $1/a = \int_{\mathcal{Z}} \exp\left(-\ell\left(f_{\text{DDSP}}(\mathbf{x}; \boldsymbol{z}), \mathbf{y}\right)\right) \mathrm{d}\boldsymbol{z}$. We have:

$$\ln p(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \ln a - \ell\left(f_{\text{DDSP}}(\mathbf{x}; \mathbf{z}), \mathbf{y}\right), \tag{2}$$

$$= \ln a - \ell(\hat{\mathbf{y}}, \mathbf{y}). \tag{3}$$

We want to find a parametric distribution $p_\theta(\mathbf{z}|\mathbf{y})$ that is similar to the posterior $p(\mathbf{z}|\mathbf{x}, \mathbf{y})$. To this end, we minimize the Kullback-Leibler divergence between them:

$$D = D_{\text{KL}}\left(p_\theta(\boldsymbol{z}|\mathbf{y}) \| p(\boldsymbol{z}|\mathbf{x}, \mathbf{y})\right), \tag{4}$$

$$= \mathbb{E}_{\mathbf{z}\sim p_\theta(\boldsymbol{z}|\mathbf{y})}\left[\ln p_\theta(\mathbf{z}|\mathbf{y}) - \ln p(\mathbf{z}|\mathbf{x}, \mathbf{y})\right], \tag{5}$$

$$= -\mathbb{H}_{p_\theta}[\mathbf{z}|\mathbf{y}] + \mathbb{E}_{\mathbf{z}\sim p_\theta(\boldsymbol{z}|\mathbf{y})}[-\ln a + \ell(\hat{\mathbf{y}}, \mathbf{y})], \tag{6}$$

with $\mathbb{H}_{p_\theta}[\mathbf{z}|\mathbf{y}] = \mathbb{E}_{\mathbf{z}\sim p_\theta(\boldsymbol{z}|\mathbf{y})}[-\ln p_\theta(\mathbf{z}|\mathbf{y})]$ being the conditional differential entropy of the parameters $\mathbf{z}$ following the parametric distribution p_θ. The constants a can be ignored as they are independent from θ.

As proposed by Higgins *et al.* [34], we define a new loss function for our optimization problem by introducing a weighting

coefficient β to the entropy term:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \theta) = -\beta \mathbb{H}_{p_\theta}[\mathbf{z}|\mathbf{y}] + \mathbb{E}_{\mathbf{z}\sim p_\theta(\boldsymbol{z}|\mathbf{y})}[\ell(\hat{\mathbf{y}}, \mathbf{y})]. \quad (7)$$

This allows us to balance the trade-off between estimation accuracy and the regularization of the distribution's width: the greater the entropy, the more "diverse" the output parameter sets are.

3.2. Distribution modeling

We model p_θ using a d-dimensional Gaussian distribution modified with normalizing flows (NFs) [35] (see Fig. 2), d being the number of DDSP parameters. The encoder outputs both the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and the coefficients $\sigma \in \mathbb{R}^d$ of the diagonal covariance matrix $\boldsymbol{\Sigma} = \sigma\mathbf{I}$, as in [36]. Samples from this Gaussian distribution are then fed into a normalizing flow, which allows to transform the simple Gaussian distribution into arbitrarily complex ones. Our NF uses T layers f_t, such as $f_{\text{NF}} = f_T \circ f_{T-1} \circ \dots f_1$. T is a hyperparameter defining the expressive power of our distribution model. We therefore introduce:

$$\mathbf{z}_0 \sim p_\theta(\boldsymbol{z}_0|\mathbf{y}), \quad (8)$$

$$\boldsymbol{z} \triangleq \boldsymbol{z}_T, \quad (9)$$

$$\forall t \in [1; T], \boldsymbol{z}_t = f_t(\boldsymbol{z}_{t-1}). \quad (10)$$

A sample $\mathbf{z}_0$ is drawn from a base distribution $p_\theta(\boldsymbol{z}_0|\boldsymbol{y})$ and then passed through the NF to get $\mathbf{z}_T$. We can compute the probability of $\mathbf{z}_T$ with the change of variable theorem [35]:

$$\ln p_\theta(\boldsymbol{z}_T|\mathbf{y}) = -\ln|\det \boldsymbol{J}_{f_{\text{NF}}}(\boldsymbol{z}_0|\mathbf{y})| + \ln p_\theta(\boldsymbol{z}_0|\mathbf{y}), \quad (11)$$

with $\ln|\det \boldsymbol{J}_{f_{\text{NF}}}(\boldsymbol{z}_0|\mathbf{y})| = \sum_{t=1}^{T} \ln\left(|\det \boldsymbol{J}_{f_t}(\boldsymbol{z}_{t-1}|\mathbf{y})|\right)$ being the absolute value of the determinant of the Jacobian matrix of f_{NF} evaluated at $\boldsymbol{z}_0$. The conditional entropy of $\mathbf{z}$ is then:

$$\mathbb{H}_{p_\theta}[\mathbf{z}|\mathbf{y}] = \mathbb{H}_{p_\theta}[\mathbf{z}_0|\mathbf{y}] + \mathbb{E}_{\mathbf{z}_0\sim p_\theta(\boldsymbol{z}_0|\mathbf{y})}\left[\ln|\det \boldsymbol{J}_{f_{\text{NF}}}(\mathbf{z}_0)|\right]. \quad (12)$$

Using this, Eq. (7) can then be rewritten as:

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{y}, \theta) = &-\beta \mathbb{H}_{p_\theta}[\mathbf{z}_0|\mathbf{y}] \\ &-\beta \mathbb{E}_{\mathbf{z}_0\sim p_\theta(\boldsymbol{z}_0|\mathbf{y})}\left[\ln|\det \boldsymbol{J}_{f_{\text{NF}}}(\mathbf{z}_0|\mathbf{y})|\right] \\ &+\mathbb{E}_{\mathbf{z}_T\sim p_\theta(\boldsymbol{z}_T|\mathbf{y})}[\ell(\hat{\mathbf{y}}_i, \mathbf{y})].\end{aligned} \quad (13)$$

The total entropy is the sum of two parts: the entropy of the Gaussian and the entropy added by the normalizing flow. We estimate the expectation terms with the Monte-Carlo method using only one sample $\mathbf{z}_0$ per input $\mathbf{y}$ [36]. We compute the entropy of the Gaussian base distribution using the closed-form [37]:

$$\mathbb{H}_{p_\theta}[\mathbf{z}_0|\boldsymbol{y}] = \frac{d}{2} + \frac{d}{2}\ln(2\pi) + \frac{1}{2}\ln|\det \boldsymbol{\Sigma}|, \quad (14)$$

which is more precise than the Monte Carlo estimation [36].

3.3. In practice

We compare the use of deterministic and the proposed inference models with equivalent backbones in two experiments: (1) estimating the parameters of an audio effect (here an equalizer), (2) estimating the parameters of a FM synthesizer.

In both experiments, we compare the use of three models:

1. a deterministic baseline (cf. Fig. 1),
2. an inference model with $T = 1$ NF layers,
3. an inference model with $T = 2$ NF layers.

We ensure that the output $\mathbf{z}$ of all models is bounded by using an output sigmoid layer. For the inference models, this last layer is included in the normalizing flow: the entropy term in the loss (Eq. (13)) aims to maximize **the entropy of the sigmoid's output**. We use deep sigmoidal flow (DSF) layers [38] with hidden size of 8 as our NF. We only denote the number of DSF layers with T: it does not count the output sigmoid. We use deep sigmoidal flow layers as they are very expressive [39] and easy to implement.

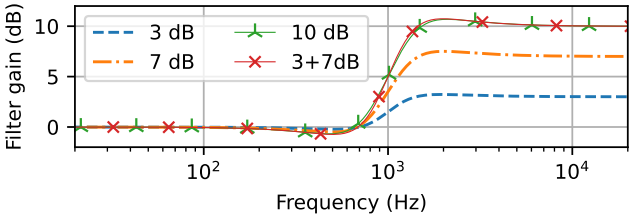


Figure 3: Frequency response of high shelf filters with respective gains of 3 dB (blue), 7 dB (orange) and 10dB (green). The red line shows the response of the cascade of the 3 dB and 7 dB filters. All filters have the same center frequency and quality factor.

4. EXPERIMENT 1: ESTIMATING EQUALIZER PARAMETERS

To gain deeper insight into the results of our experiment, we conduct it in a controlled setting that is, where the applied audio effects and their parameters are known. Since real audio data typically does not include this information, we use synthetic data for this purpose. We also use a specific setting of audio effects which allows to have several parameter settings leading to the same audio output. The synthetic data $\mathbf{y}$ are obtained by processing $\mathbf{x}$ with a single high shelf filter while our model estimate this process using 2 high shelf filters. Several solution therefore exist.

4.1. High shelf filters

High shelf filters allow to control the gain above a center frequency. They have 3 parameters: (1) the center frequency f ; (2) the gain of the filter above the center frequency g ; (3) the quality factor Q which sets the transition bandwidth. There are multiple combinations of parameters that allow to match closely the curve of one filter using two filters [40], as illustrated in Fig. 3.

Our model outputs the parameters of 2 high shelf filters in cascade and tries to reconstruct a curve made with only 1 high shelf filter. We therefore expect our model to estimate the correct $f_1 = f_2 = f$ and $Q_1 = Q_2 = Q$, such that the sum of their gains matches the one of the ground truth $g_1 + g_2 = g$, as shown in Fig. 3.

4.2. Dataset

We randomly selected unprocessed audio samples $\mathbf{x}$ from the Million Song Dataset (MSD) [41]. It contains stereo files sampled at 22.05 kHz, which we converted to mono. We use a subset of 40,000 for training, 5,000 for validation and 5,000 for testing. At each epoch, we randomly choose a 5 seconds long chunk of each track. We constructed the processed signals $\mathbf{y}$ by applying one high

shelf filter [42] to $\mathbf{x}$ with random parameters $\mathbf{v}$ (cf. Fig. 1) where $\mathbf{v}$ is sampled from the uniform distribution $\mathbf{v} \sim \mathcal{U}(\bullet; [0,1]^3)$. We then map the parameters using either affine or exponential functions [43]:

$$f = e^{(\ln(f_{\max})-\ln(f_{\min}))\mathrm{v}_1+\ln(f_{\min})}, \tag{15a}$$

$$g = (g_{\max} - g_{\min})\mathrm{v}_2 + g_{\min}, \tag{15b}$$

$$Q = (Q_{\max} - Q_{\min})\mathrm{v}_3 + Q_{\min}, \tag{15c}$$

with $f \in [500, 2500]$ Hz, $g \in [-20, 20]$ dB and $Q \in [0.1, 3]$.

4.3. Model architecture

All our models use the same backbone and have roughly the same number of trainable parameters ($\approx$ 295 k). Our models take as inputs the Log-scaled Mel spectrograms (LMS) of $\mathbf{y}$, which are computed using the `nnAudio` Python library [44]. We use LMS with 64 Mel bands and a hop length of 23 ms. We consider this LMS as a multi-channel signal, where each Mel band is a channel. This LMS is fed into multiple 1-D ConvNeXt blocks [45]: 3 blocks with 64 channels and 3 with 128 channels. The output of this convolutional network is then averaged along the time dimension and fed to a small multi-layer perceptron (MLP). The MLP uses batch normalization [46]. All activation functions (in the ConvNeXt blocks and in the MLP) are Swish activations [47], except the output which is a Sigmoid. We denote the output $\mathbf{z} \in [0,1]^6$.

4.4. Differentiable audio effects

The network estimates the parameters of 2 high shelf filters. Each band has 3 parameters, $\mathbf{z}$ is therefore 6-dimensional. We map $\mathbf{z}$ to the frequency range using exponential functions (as Eq. (15a)) and to the gain and Q's range using affine functions (like Eqs. (15b), (15c)). Since there are two differentiable EQ bands, we restrain their gain range to $g_1, g_2 \in [-10, 10]$ dB. We use the parameter ranges for frequencies and quality factors that were used to create the signals $\mathbf{y}$: $f_1, f_2 \in [500, 2500]$ Hz, $Q_1, Q_2 \in [0.1, 3]$. We implement filtering operations in the frequency domain as it is significantly faster than filtering in the time domain for training using graphics processing units (GPUs).

4.5. Loss function

As in our previous work [16], we use a Log-scale Mel spectrogram loss:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\ln(|\mathrm{Mel}\{\hat{\boldsymbol{y}}\}| + 1) - \ln(|\mathrm{Mel}\{\boldsymbol{y}\}| + 1)\|_2^2 . \tag{16}$$

We compute the Mel spectrograms using the nnAudio library [44] that is based on a short-time Fourier transform (STFT) with a window size of 4096 and a hop length of 1024 samples (46 ms). We then compute the Mel spectrograms with 128 Mel bands.

For the inference models, we set empirically the weight of the entropy term in the loss of (Eq. (7)) to $\beta = 0.1$.

4.6. Training details

Each model is trained during 200 epoch. One epoch corresponds to 40,000 training samples and 5,000 validation samples. We use the AdamW optimizer [48] with a learning rate of 5×10^{-5}, a weight decay of 10^{-3} and a batch size of 16. We save the weights that led to the best validation score and use them for testing.

Table 1: Quantitative results of the equalizer experiment.

Model	T	Entropy bits/dim ↑	Audio distortion Mel ↓	MR-STFT ↓	SI-SDR ↑
Deter.	-	-	0.235 ± 0.429	0.412 ± 0.760	14.8 ± 5.7
Infer.	1	-1.51 ± 0.73	0.286 ± 0.458	0.477 ± 0.800	13.5 ± 6.1
Infer.	2	-1.41 ± 0.70	0.289 ± 0.471	0.481 ± 0.824	13.6 ± 6.0
Input	-	-	2.154 ± 2.910	4.302 ± 6.685	3.0 ± 9.0

Table 2: Results when we take the best of N_{best} samples. The inference model uses $T = 1$ flow layers.

Model	N_{best}	Mel ↓	MR-STFT ↓	SI-SDR ↑
Deter.	-	0.235 ± 0.429	0.412 ± 0.760	14.8 ± 5.7
Infer.	1	0.286 ± 0.458	0.477 ± 0.800	13.5 ± 6.1
	2	0.241 ± 0.428	0.401 ± 0.731	15.3 ± 6.2
	3	0.226 ± 0.435	0.373 ± 0.731	16.2 ± 6.3
	4	0.214 ± 0.415	0.354 ± 0.714	16.8 ± 6.4
	5	0.206 ± 0.413	0.340 ± 0.700	17.3 ± 6.5
	10	0.186 ± 0.396	0.307 ± 0.668	18.7 ± 6.8
Uniform	10	0.667 ± 1.138	1.141 ± 2.271	$11.8 + \pm 7.5$

4.7. Results

In this section, we analyze quantitatively and qualitatively our results.

4.7.1. Quantitative results

The test is performed over the 5,000 tracks of the test dataset. As the signals $\mathbf{x}$ are processed with random parameters $\mathbf{v}$ to create $\mathbf{y}$, we run the test 5 times and indicate the average and standard deviation value over those.

In Table 1, we present for each model the results both in terms of audio quality and conditional differential entropy $\mathbb{H}_{p_\theta}[\mathbf{z}|\mathbf{y}]$. We measure the entropy in bits/dim:

$$\mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{y})}[-\log_2 p_\theta(\mathbf{z}|\mathbf{y})]/d. \tag{17}$$

It is a differential entropy computed using a density function rather than a probability mass function, so it is not necessarily positive. Moreover, since the output parameters are bounded $\mathbf{z} \in [0;1]^6$, the entropy has an upper bound: $\mathbb{H}[\mathbf{z}|\mathbf{y}] \leq 0$ bits/dim [49, p. 269].

To quantify the audio quality we use three metrics: (1) the Log-scaled Mel Spectrogram loss used for training (Mel) ; (2) the multi-resolution STFT (MR-STFT) loss revisited by Schwär and Müller [50] ; (3) the scale invariant signal-to-distortion ratio (SI-SDR) [51], which is measured in dB. As a baseline, we also report in row "Input", the value of the losses that would be obtained by directly comparing the unprocessed signal $\mathbf{x}$ to the ground truth processed signal $\mathbf{y}$.

The best model in terms of audio quality is the deterministic baseline. This was expected since our two inference models have additional objectives (entropy). We also see that our two inference models (Infer. 1 and Infer. 2) have roughly have the same performance: 1 DSF layer seems sufficient for such a simple effects chain.

In Table 1, we represented the "Expectation" of the results:

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p(\boldsymbol{x},\boldsymbol{y}),\mathbf{z}\sim p_\theta(\boldsymbol{z}|\mathbf{y})}[\ell(\mathbf{y}, f_{\text{DDSP}}(\mathbf{x},\mathbf{z}))]. \tag{18}$$

Indeed, once our model estimates a distribution, we sample $\mathbf{z}$ from this distribution, for each compute a loss and finally get the "Expectation" (mean value) of this loss over the $\mathbf{z}$.

We can also mimick a real user scenario, in which the user runs once the estimation (with a first sampling of $\mathbf{z}$) and runs it again (with another choice of $\mathbf{z}$) until they are satisfied with the results. In Table 2, we indicate the results we get by sampling a number N_{best} of parameter vectors $\mathbf{z}$, and then only retaining the best results over the N_{best}. We see that taking the best sample of $N_{\text{best}} = 2$ already outperforms the deterministic baseline, and the results get better and better as we increase the number of tries N_{best}.

As a baseline, we report in row "Uniform" the results obtained by sampling $\mathbf{z}$ from a uniform distribution $\mathcal{U}(\bullet; [0,1]^6)$ (instead of the distribution estimated by our model). We see that sampling from an uniform distribution is not efficient. Even with as low as 6 parameters in this case, the best out of 10 parameters sets sampled this way still performs significantly worse than using the deterministic baseline.

4.7.2. Qualitative analysis

As a reminder, our goal is to design a model that could obtain the same output sound using multiple combination of parameters (exploiting the links and redundancies in the processor parameters). We want to see if our model behaves as we expected in section 4.1[2].

To illustrate this, we perform the following experiment: for each examples of the test set, we sample 20 parameter combinations using our inference model with $T = 1$ DSF layer. We then compute the Pearson correlation matrix of the parameters based on these 20 samples.

In Fig. 4, we show the mean of the correlations matrices computed over the whole test dataset, with f_1, g_1 and Q_1 being the parameters of the first equalizer band and f_2, g_2, Q_2 the parameters of the second. As expected, most parameters sampled from our approximate distribution are uncorrelated. However, the gains of the two bands (g_1 and g_2) show the strongest correlation in absolute value (-0.51): if one increases, the other decreases such that $g_1 + g_2 = g$.

In Fig 5, we compare the ground-truth value (green +), deterministically estimated value (red $\times$) and estimated distribution (background 2D-histogram) for a given audio. Since the distribution lies in a $d = 6$ dimensional space, we only represent the marginal distributions $p_\theta(f_1, f_2|\boldsymbol{y})$, $p_\theta(g_1, g_2|\boldsymbol{y})$ and $p_\theta(Q_1, Q_2|\boldsymbol{y})$. For this example, both the deterministic and inference model estimate quite precisely the filter frequency (f_1 and f_2), but the gain estimations (g_1 and g_2) are not as good and the quality factors (Q_1 and Q_2) are completely off. This can be explained by the choice of the loss function. However, as expected from Fig. 4, the frequencies and Q's sampled from the approximate distribution do not seem correlated; while the two gains g_1 and g_2 seem anti-correlated. Some of the parameter sets sampled in this figure have $g_2 \approx 0$, which means means that in practice, the model sometimes uses only one filter.

[2]We expect our model to estimate the correct $f_1 = f_2 = f$ and $Q_1 = Q_2 = Q$, and that the sum of their gains matches the one of the ground truth $g_1 + g_2 = g$, as in Fig. 3.

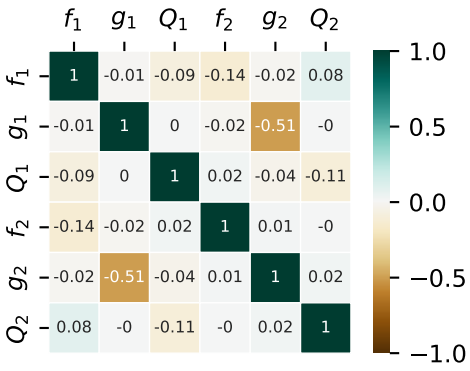
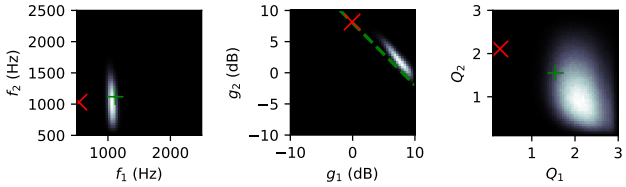


Figure 4: Mean correlation matrix of the equalizer parameters sampled with our inference model.



(a) Joint distribution of the frequencies. (b) Joint distribution of the gains. (c) Joint distribution of the quality factors.

Figure 5: Histograms of the parameter distribution given one input $\boldsymbol{y}$. The green symbols + or line shows the ground truth. The red $\times$ shows the estimate obtained with the deterministic baseline.

5. EXPERIMENT 2: SOUND MATCHING WITH A DIFFERENTIABLE FM SYNTHESIZER

In this experiment, we estimate the parameters used to generate sounds using a simple frequency modulation synthesizer with one modulator and one carrier.

5.1. FM synthesis

Frequency modulation synthesizers, introduced by Chowning [52], rely on modulating the phase of *carrier* operators with *modulator* operators. The output signal of a simple carrier y_c of base frequency f_c and amplitude I_c modulated with a sinusoidal modulator with frequency f_m and amplitude I_m is:

$$y_c(t) = I_c \sin(2\pi f_c t + I_m \sin(2\pi f_m t)). \tag{19}$$

A differentiable FM synthesizer was proposed by Caspe *et al.* [4] where the frequency ratios were fixed in order to facilitate training with a multi-resolution STFT (MR-STFT) loss.

We implement our own version so that our network can estimate the frequency of the modulator operator. For the sake of simplicity, we do our experiment only on a combination of one carrier and one modulator. Our synthesizer only has 2 parameters: the modulation index I_m and the modulation frequency f_m. As in the DX7, I_m is bounded in $I_m \in [0; 4\pi]$ [4]. We also bound the frequency ratio $f_m/f_c \in [0.5; 10]$. We set the carrier amplitude to $I_c = 1$ and the carrier frequency to $f_c = 110$Hz, with a sampling-rate of 44.1kHz.

5.2. Dataset

We create our dataset by sampling normalized parameters $\mathbf{v}$ in a uniform distribution and then generating the sound examples. The modulation index I_m is the mapped in the following way:

$$g_{\text{dB}} = \frac{3}{4} \cdot 99 \cdot (\text{v}_1 - 1), \qquad I_m = 4\pi \cdot 10^{\frac{g_{\text{dB}}}{20}}. \tag{20a}$$

We did this to replicate parameter/modulation index curve of the original DX7 [4, 53]. The frequency ratio is mapped with an affine function:

$$f_m/f_c = 9.5\text{v}_2 + 0.5. \tag{20b}$$

We generate sounds with a duration of 125ms.

5.3. Model architectures

We only consider stationary sounds. Therefore, our model takes as input the magnitude of the discrete Fourier transform of the signal, fed into a MLP with Swish activations [47]. All models for this experiments approximately have the same number of parameters ($\approx$ 170 k). The output normalized parameters are mapped to synthesizer parameters according to Eqs (20a) and (20b).

5.4. Loss function

Neural networks struggle to estimate correct frequencies when using typical audio losses, such as the MR-STFT loss [54]. This lead the authors of the DDX7 paper [4] to set the modulation ratios of their models to fixed values. In our experiment, we want the model to estimate correctly the frequency of the modulator operator. We therefore need a loss function that is convex with regards to the frequency ratio.

Among previously proposed solutions, the most promising is the spectral optimal transport based spectral loss function. It measures the "displacement of spectral energy" [55] across frequencies rather than the difference between the magnitudes. This loss is then proportional to the pitch error for synthesizer signals.

We use a sum of a spectral optimal transport (SOT) loss and a MR-STFT loss, following Torres *et al.* [55]. We use the SOT loss with 64 frequency bins, and the MR-STFT as revisited in [50].

5.5. Training details

Each model is trained for 700 epochs, with one epoch consisting in 100,000 training samples and 10,000 validation samples. We use the AdamW algorithm [48] with a learning rate of 10^{-4}, a weight decay of 10^{-3} and a batch size of 128. We use the weights that performed the best validation score for testing.

5.6. Results

5.6.1. Quantitative results

In Table 3, we indicate the results for the FM synthesis sound matching experiment for our test-set made of 100,000 $\mathbf{y}$ samples synthesized with random uniform parameters $\mathbf{v}$. We evaluate our system using the SOT loss used for training, the log-scaled Mel spectrogram loss and the SI-SDR [51]. We also report the conditional differential entropy of the inference models in bits/dim.

As a baseline we also report in row "Random" the results obtained when sampling random parameters $\mathbf{z} \sim \mathcal{U}(\bullet; [0, 1]^2)$.

Table 3: Sound matching results of the FM synthesis experiment. The row "Rand." indicates the results obtained when sampling $\mathbf{v}$ from a uniform distribution.

Model	T	Entropy	Audio distortion		
		bits/dim ↑	SOT ↓	Mel ↓	SI-SDR ↑
Deter.	-	-	$0.39_{\pm 1.24}$	$0.063_{\pm 0.278}$	$32.52_{\pm 19.9}$
Infer.	1	$-2.43_{\pm 2.51}$	$2.12_{\pm 2.33}$	$0.386_{\pm 0.506}$	$15.20_{\pm 20.62}$
Infer.	2	$-2.42_{\pm 2.50}$	$2.13_{\pm 2.30}$	$0.385_{\pm 0.507}$	$15.16_{\pm 20.76}$
Rand.	-	-	$51.6_{\pm 82.6}$	$4.686_{\pm 4.074}$	$2.13_{\pm 21.091}$

Both our inference models perform similarly in this experiment. Here again, 1 DSF layer is enough. Also, as for the high shelf filter experiment, results are worst when using the inference model than the deterministic baseline. However, the probabilistic models could have other uses like introducing variations in the sound : by sampling a new parameter each time a note is played so that every new note has a similar timbre than the previous but with slight variations [6].

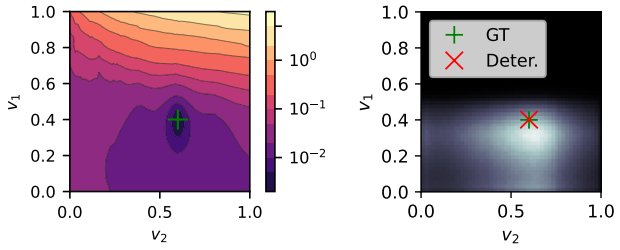
5.6.2. Qualitative analysis

We show in Figs. 6, 7 side-by-side maps of the SOT loss function as well as a histogram of parameters sampled with our inference model with $T = 1$ DSF layers. We also show the ground truth parameters (GT) and the estimation by the deterministic baseline (Deter.). On these figures, v_1 is the parameter controlling the modulation index and v_2 controls the frequency ratio (Eqs. (20a), (20b)).

In Fig. 6, the model tries to reconstruct a sound synthesized using low modulation index - which yields harmonics with very low amplitudes [52]. Therefore, the frequency ratio does not matter. Also, due the mapping between the normalized parameters $\mathbf{z}$ and the modulation index (Eq. (20a)), having $\text{z}_1 < 0.4$ implies a "negligible index" $I_m \ll 1$, [53, p. 58]. Inside of this range, the exact value of the normalized parameter associated with the modulation index does not really matter, as shown on the loss map of Fig. 6. In this case, the inference model samples are all over the region where the loss is low. For this example, the conditional differential entropy is very high (-0.54 bits/dim).

In Fig. 7, the model tries to reconstruct a sound synthesized with a high modulation index I. Since the harmonics created by the synthesizer have a high amplitude [52], estimating precisely both the modulation index and frequency is important to reconstruct the sound. Thus, the approximate distribution has a very low variance: all the samples are very close (cf. Fig. 7). The conditional entropy of the parameters is low for this example (-8.28 bits/dim).

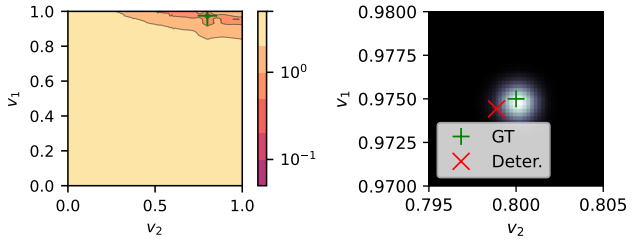
We also synthesize sounds $\boldsymbol{y} = f_{\text{DDSP}}(I_m, f_m)$ for various $I_m \in [0; 4\pi]$, $f_m \in [0.5f_c; 10f_c]$ and we show in Fig. 8 the conditional entropy of the estimated parameters from our model $\mathbb{H}_{p_\theta}[\mathbf{z}|\boldsymbol{y}]$ in function of I_m. The conditional entropy decreases as the modulation index increases. We interpret this in the following way: as the modulation index increases, the amplitude of the harmonics also increases [52] and therefore the estimation of the parameters needs to be more precise to accurately reconstruct the sound, hence the decreasing entropy.



(a) Map of the loss function.

(b) Histograms of the samples. $\mathbb{H}_{p_\theta}[\mathbf{z}|\boldsymbol{y}] = -0.54$ bits/dim.

Figure 6: Loss function map and histogram of samples for a ground truth with a low modulation index.



(a) Map of the loss function.

(b) Histograms of the samples, zoomed in on the interesting part of the figure. $\mathbb{H}_{p_\theta}[\mathbf{z}|\boldsymbol{y}] = -8.28$ bits/dim.

Figure 7: Loss function map and histogram of samples for a ground truth with a high modulation index.

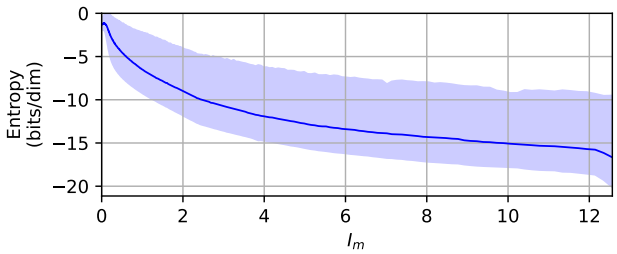


Figure 8: $\mathbb{H}_{p_\theta}[\mathbf{z}|\mathbf{y}]$ of the estimated parameters in function of the modulation index I_m. The solid line shows the average results across f_m values, the shaded area shows the 1st and 9th deciles.

6. CONCLUSION

This paper proposes a method to design and optimize a neural network that could perform the blind estimation of processor parameters and give multiple parameter sets that all reconstruct the target sound. To do this, we use normalizing flows and rely on differentiable digital signal processing and an appropriate loss function for both accurate reconstruction and diverse parameters.

We evaluate our method with two experiments, showing that our model is able to exploit the redundancies and links between the processor parameters. The distribution of estimated parameters is tailored to the desired effect: the irrelevant parameters for a given sound are chosen almost randomly while the more important ones can be very precisely set.

We will apply this method in future works to complex processors chain and use it in realistic experiments.

7. REFERENCES

[1] T. Wilmering *et al.* "A history of audio effects," *Applied Sciences*, vol. 10, no. 3, p. 791, 2020.

[2] P. Tagg, "Analysing popular music: Theory, method and practice," *Popular Music*, vol. 2, pp. 37–67, 1982.

[3] J. Engel *et al.* "DDSP: Differentiable digital signal processing," in *Proc. of ICLR*, Addis Ababa, Ethiopia, 2020.

[4] F. Caspe, A. McPherson, and M. Sandler, "DDX7: Differentiable FM synthesis of musical instrument sounds," in *Proc. of ISMIR*, Bengaluru, India, 2022, pp. 608–616.

[5] N. Masuda and D. Saito, "Improving semi-supervised differentiable synthesizer sound matching for practical applications," *IEEE/ACM TASLP*, vol. 31, pp. 863–875, 2023.

[6] G. Le Vaillant and T. Dutoit, "Latent space interpolation of synthesizer parameters using timbre-regularized auto-encoders," *IEEE/ACM TASLP*, vol. 32, pp. 3379–3392, 2024.

[7] C. J. Steinmetz *et al.* "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *Proc. of IEEE ICASSP*, Toronto, Canada, 2021, pp. 71–75.

[8] M. A. Martínez Ramírez *et al.* "Automatic music mixing with deep learning and out-of-domain data," in *Proc. of ISMIR*, Bengaluru, India, 2022, pp. 411–418.

[9] M. A. Martínez Ramírez *et al.* "Differentiable signal processing with black-box audio effects," in *Proc. of IEEE ICASSP*, Toronto, Canada, 2021, pp. 66–70.

[10] J. T. Colonel and J. D. Reiss, "Reverse engineering of a recording mix with differentiable digital signal processing," *JASA*, vol. 150, no. 1, pp. 608–619, 2021.

[11] S. Lee *et al.* "Searching for music mixing graphs: A pruning approach," in *Proc. of DAFx*, 2024, pp. 147–154.

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

[13] J. Koo *et al.* "Music mixing style transfer: A contrastive learning approach to disentangle audio effects," in *Proc. of IEEE ICASSP*, Rhodes, Greece, 2023.

[14] H. Han, V. Lostanlen, and M. Lagrange, "Learning to solve inverse problems for perceptual sound matching," *IEEE/ACM TASLP*, vol. 32, pp. 2605–2615, 2024.

[15] S. Lee *et al.* "Blind estimation of audio processing graph," in *Proc. of IEEE ICASSP*, Rhodes, Greece, 2023.

[16] C. Peladeau and G. Peeters, "Blind estimation of audio effects using an auto-encoder approach and differentiable digital signal processing," in *Proc. of IEEE ICASSP*, Seoul, South Korea, 2024, pp. 856–860.

[17] P. Esling *et al.* "Flow synthesizer: Universal audio synthesizer control with normalizing flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.

[18] G. Le Vaillant, T. Dutoit, and S. Dekeyser, "Improving synthesizer programming from variational autoencoders latent space," in *Proc. of DAFx*, Virtual only conference, 2021, pp. 276–283.

[19] G. Le Vaillant and T. Dutoit, "Synthesizer preset interpolation using transformer auto-encoders," in *Proc. of IEEE ICASSP*, Rhodes, Greece, 2023.

[20] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable dsp," in *Proc. of ISMIR*, Online, 2021.

[21] B. Hayes, C. Saitis, and G. Fazekas, "Neural waveshaping synthesis," in *Proc. of ISMIR*, Online, 2021, pp. 254–261.

[22] S. Shan *et al.* "Differentiable wavetable synthesis," in *Proc. of IEEE ICASSP*, 2022, pp. 4598–4602.

[23] S. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. of DAFx*, Virtual only conference, 2020, pp. 265–272.

[24] B. Kuznetsov, J. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. of DAFx*, Virtual only conference, 2020, pp. 265–272.

[25] C.-Y. Yu and G. Fazekas, "Singing voice synthesis using differentiable LPC and glottal-flow-inspired wavetables," in *Proc. of ISMIR*, Milan, Italy, 2023, pp. 667–675.

[26] J. T. Colonel and J. Reiss, "Approximating ballistics in a differentiable dynamic range compressor," in *Proc. of the AES*, New York, USA, 2022.

[27] C.-Y. Yu *et al.* "Differentiable all-pole filters for time-varying audio systems," in *Proc. of DAFx*, Surrey, United Kingdom, 2024, pp. 345–352.

[28] S. Lee, H.-S. Choi, and K. Lee, "Differentiable artificial reverberation," *IEEE/ACM TASLP*, vol. 30, pp. 2541–2556, 2022.

[29] G. D. Santo *et al.* "Differentiable feedback delay network for colorless reverberation," in *Proc. of DAFx*, Copenhagen, Denmark, 2023, pp. 244–251.

[30] A. I. Mezza *et al.* "Data-driven room acoustic modeling via differentiable feedback delay networks with learnable delay lines," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2024, no. 1, p. 51, 2024.

[31] A. Carson *et al.* "Differentiable grey-box modelling of phaser effects using frame-based spectral processing," in *Proc. of DAFx*, Copenhagen, Denmark, 2023, pp. 219–226.

[32] J. T. Colonel, M. Comunità, and J. D. Reiss, "Reverse engineering memoryless distortion effects with differentiable waveshapers," in *Proc. of the AES*, New York, USA, 2022, p. 10.

[33] Y.-T. Yeh *et al.* "DDSP guitar amp: Interpretable guitar amplifier modeling," in *Proc. of IEEE ICASSP*, 2025, pp. 1–5.

[34] I. Higgins *et al.* "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. of ICLR*, Toulon, France, 2017.

[35] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. of ICML*, PMLR, 2015, pp. 1530–1538.

[36] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. of ICLR*, Banff, Canada, 2014.

[37] K. P. Murphy, *Probabilistic Machine Learning: An Introduction* (Adaptive Computation and Machine Learning Series). Cambridge, Massachusetts: The MIT Press, 2022.

[38] C.-W. Huang *et al.* "Neural autoregressive flows," in *Proc. of ICML*, vol. 80, Stockholm, Sweden: PMLR, 2018, pp. 2078–2087.

[39] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 3964–3979, 2021.

[40] J. Abel and D. Berners, "Filter design using second-order peaking and shelving sections," in *Proc. of ICMC*, Miami, FL, USA, 2004.

[41] T. Bertin-Mahieux *et al.* "The million song dataset," in *Proc. of ISMIR*, Miami, USA, 2011, pp. 591–596.

[42] R. Bristow-Johnson, *RBJ audio-EQ-cookbook*, 2005.

[43] F. Mockenhaupt, J. S. Rieber, and S. Nercessian, "Automatic equalization for individual instrument tracks using convolutional neural networks," in *Proc. of DAFx*, Surrey, United Kingdom, 2024, pp. 57–64.

[44] K. W. Cheuk *et al.* "nnAudio: An on-the-fly GPU audio to spectrogram conversion toolbox using 1D convolutional neural networks," *IEEE Access*, vol. 8, pp. 161 981–162 003, 2020.

[45] Z. Liu *et al.* "A ConvNet for the 2020s," in *Proc. of CVPR*, New Orleans, LA, USA: IEEE, 2022, pp. 11 976–11 986.

[46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of ICML*, vol. 37, Lille, France: PMLR, 2015, pp. 448–456.

[47] P. Ramachandran, B. Zoph, and Q. V. Le, *Searching for activation functions*, 2017. arXiv: 1710.05941 [cs].

[48] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. of ICLR*, New Orleans, LA, USA, 2019. arXiv: 1711.05101.

[49] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, N.J, USA: Wiley-Interscience, 2006.

[50] S. Schwär and M. Müller, "Multi-scale spectral loss revisited," *IEEE SPL*, vol. 30, pp. 1712–1716, 2023.

[51] J. L. Roux *et al.* "SDR – half-baked or well done?" In *Proc. of IEEE ICASSP*, Brighton, United Kingdom, 2019, pp. 626–630.

[52] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Computer Music Journal*, vol. 21, no. 2, pp. 46–54, 1977. JSTOR: 23320142.

[53] J. M. Chowning and D. Bristow, *FM Theory & Applications: By Musicians for Musicians*. Tokyo, Japan: Yamaha Music Foundation, 1986.

[54] J. Turian and M. Henry, "I'm sorry for your loss: Spectrally-based audio distances are bad at pitch," in *Proc. of NeurIPS*, Virtual only conference, 2020.

[55] B. Torres, G. Peeters, and G. Richard, "Unsupervised harmonic parameter estimation using differentiable DSP and spectral optimal transport," in *Proc. of IEEE ICASSP*, Seoul, South Korea, 2024, pp. 1176–1180.