# ONE-SHOT PARAMETRIC AUDIO PRODUCTION STYLE TRANSFER WITH APPLICATION TO FREQUENCY EQUALIZATION

*Stylianos I. Mimilakis*[♯,♮] *Nicholas J. Bryan*[♮] *Paris Smaragdis*[♮,♭]

[♯] Fraunhofer-IDMT
[♮] Adobe Research
[♭] University of Illinois at Urbana-Champaign, Department of Computer Science

## ABSTRACT

Audio production is a difficult process for many people, and properly manipulating sound to achieve a certain effect is non-trivial. In this paper, we present a method that facilitates this process by inferring appropriate audio effect parameters in order to make an input recording sound similar to an unrelated reference recording. We frame our work as a form of *parametric* style transfer that, by design, leverages existing audio production semantics and manipulation algorithms, avoiding several issues that have plagued audio style transfer algorithms in the past. To demonstrate our approach, we consider the task of controlling a parametric, four-band infinite impulse response equalizer and show that we are able to predict the parameters necessary to transform the equalization style of one recording to another. The framework we present, however, is applicable to a wider range of parametric audio effects.

***Index Terms*—** Parametric style transfer, one-shot learning, deep learning, parametric equalization

## 1. INTRODUCTION

Obtaining high-quality recordings (e.g. speech, music) that can be directly used for commercial use (e.g. podcasting) is challenging. Everyday users, for example, typically record in poor acoustic conditions and use low-quality recording equipment, resulting in substandard recording quality. To perform enhancement on these recordings, audio engineers typically employ digital audio effects (DAFX) or a series of signal manipulations that sequentially improve the sound quality along semantically meaningful axis (e.g. noise level, reverberation level, equalization, and loudness). This process is commonly referred to as audio *production* [1]. Although there are a plethora of tools for this purpose these tools can be difficult and time-consuming to use for both novice and expert users.

To automate audio production, deep learning approaches have shown promise [2, 3, 4, 5, 6, 7, 8]. These approaches can be organized into two categories. The first category aims at replacing existing DAFX by directly mapping untreated recordings to an enhanced output [4, 9, 5] and the second aims to predict parameters to control existing tools that themselves manipulate the recording [3, 6, 7]. The latter approach is appealing because existing tools typically use semantically meaningful control, induce minimal audible artifacts, and allow end-users to refine the predicted parameters.

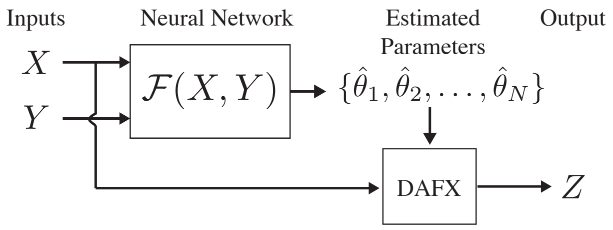

**Fig. 1**. One-shot *parametric* production style transfer.

Past methods for parameter prediction typically focus on transforming recordings into a *single* production style. As a result, they must be retrained to change the predicted production style and require large, expensive human-labeled datasets per style. Recent work [8] avoids this issue by proposing a method that receives both an untreated recording and a paired produced reference example recording and then uses a deep Siamese network and decision tree estimator to predict parameters that control a dynamic range compressor. While promising, this approach has several limitations including 1) assuming the reference signal is the exact produced version of the untreated recording (paired reference) and 2) requiring a two-step training process involving both deep learning and decision trees, complicating both the training and inference design steps.

To address these limitations, we present a new method for multiple-style *one-shot* parameter prediction by example as illustrated in 1. Compared to prior work, 1) we eliminate the need for paired inputs and develop a general method that learns directly from unpaired data in a self-supervised manner to control discrete, quantized parameters 2) we frame our approach as a *parametric* style transfer in the context of one-shot learning [10], 3) we propose an updated network architecture compared to past work that enables the analysis and processing of signal of varying lengths, and 4) we experimentally demonstrate our proposed approach on the task of controlling a parametric 4-band infinite impulse response (IIR) equalizer.

* This work was performed during an internship at Adobe Research.

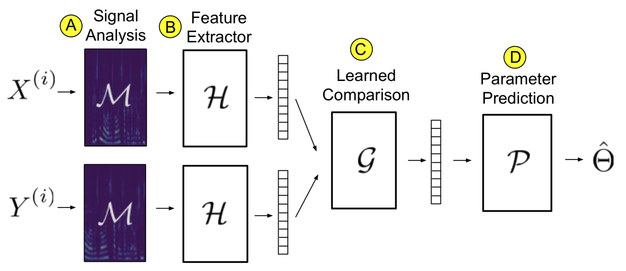

**Fig. 2**. Overview of our proposed model $\mathcal{F}$. (A) Front-end Mel-spectrogram feature extraction module, (B) latent feature extractor module (with shared weights), (C) learned comparison module, and (D) parameter prediction module.

## 2. PROPOSED MODEL

Our proposed model $\mathcal{F}$ inputs an untreated recording $X^{(i)}$ and an unpaired produced recording $Y^{(i)}$ and predicts the DAFX parameters $\Theta^{(i)}$. The proposed model consists of four modules. The latter three are learnable.

### 2.1. Signal analysis

The signal analysis module, $\mathcal{M}$, computes a decibel-scaled Mel-spectrogram of the input signal(s). The computation is based on the short-time Fourier transform with 20ms window size, 3ms hop size, and a 128 Mel-band filter-bank.

### 2.2. Latent feature extractor

Our latent feature extraction module has two parallel, identical branches $\mathcal{H}$ that follow a Siamese architecture [11]. Each branch independently extracts a latent representation (embedding) using five blocks of non-linear operations. This module is loosely based on [8]. Each block includes a two-dimensional (2D) convolutional operator using multiple convolutional kernels, followed by 2D batch-normalization [12]. For batch-normalization, we use a single scalar for the average and variance, respectively, for each convolution channel.

After batch-normalization, we use the leaky rectified linear unit (LReLU) activation function [13] with a negative slope of $1e^{-2}$, followed by a max-pooling operator. Compared to the model presented in [8], we added a max-pooling operator that is applied across time. The shape of the max-pooling operator is $(2, 1)$. The size of the convolutional kernels in each set is $(3, 3)$. The main difference between the blocks is the number of the employed convolutional kernels. Specifically, for the first set, 10 kernels are used, whereas for the second and third sets, 15 kernels are employed. For the last two sets, the number of kernels is 20. Following the convolutional layers, we compute a global average across spectrogram time frames. The global averaging reduces learnable time-based dependencies and enables the usage of variable-length recordings as inputs to our model. The resulting features are denoted as $\hat{\xi}_X^{(i)} \in \mathbb{R}^F$ and $\hat{\xi}_Y^{(i)} \in \mathbb{R}^F$ for the untreated and the reference signals, respectively.

### 2.3. Learned comparison

Following the latent feature extractor module, we feed the feature embeddings $\hat{\xi}_X$ and $\hat{\xi}_Y$ into a learned comparison module $\mathcal{G}$. This is in contrast to prior work, which uses an element-wise subtraction comparison [8]. The learned comparison employs two steps. The first step is a location-preserving concatenation of $\hat{\xi}_X$ and $\hat{\xi}_Y$ that creates a matrix $\mathbf{Z} \in \mathbb{R}^{2\times F}$ of two "channels" and each channel is of size $F$. The second step is to perform a one-dimensional convolution of $\mathbf{Z}$ with a filter, that is subject to training, and projects down the number of channels to one. In addition to this, the one-dimensional filter has length $B$, essentially taking into account $B$ elements for the comparison. We set $B = 129$, based on the frequency dimension of the feature extraction model. To avoid changes in the dimensionality of the resulting feature vector after convolution, we apply zero-padding of the order $\frac{B-1}{2}$. The motivation for using a learnable comparison module comes from 1) recent advances in one-shot (few-shot) learning in alternative domains [14] and 2) the following observation of the behavior of $\mathcal{H}$ in the employed Siamese setting.

Since we have a single target variable and a single optimization objective, it follows that in a directed hierarchical model like ours, we have also a single latent variable $\xi$ that conditions the input of $\mathcal{P}$. In an ideal scenario, let $\xi$ be a latent variable that delivers accurate parameter prediction by means of $\mathcal{P}(\xi)$. Furthermore, assume that $\mathcal{H}$ is an unbiased estimator of the ideal $\xi$. Since $\xi$ is dependent on both signals $X^{(i)}$ and $Y^{(i)}$, we expect the following:

$$\mathbb{E}[\mathcal{H}(X^{(i)}) - \xi^{(i)}] + \mathbb{E}[\mathcal{H}(Y^{(i)}) - \xi^{(i)}] = 0\,, \text{where} \quad (1)$$

the expectations *depend* on both $X^{(i)}$ and $Y^{(i)}$, and leads to:

$$\mathbb{E}[\mathcal{H}(X^{(i)}) - \mathcal{H}(Y^{(i)})] = 0\,. \quad (2)$$

In the unbiased case, (2) suggests that $\mathcal{H}$ will enforce, on average, identical latent information between the untreated and produced signal representations. For the biased case, some minor differences could be expected. In our pilot experiments, practically employing a biased estimator, little-to-no differences between $\mathcal{H}(X^{(i)})$ and $\mathcal{H}(Y^{(i)})$ were found during optimization. This explains a limitation of the element-wise subtraction-based comparison model presented in [8], that requires additional training stages and possibly poor prediction performance using unpaired data. Consequently, using the subtraction-based comparison [8] the module $\mathcal{P}$ is conditioned on degenerate latent information.

### 2.4. Parameter prediction

The output of the comparison module is given to our parameter prediction module $\mathcal{P}$. Here, we predict the parameters needed to transform the style of one recording into another. $\mathcal{P}$ is implemented using two feed-forward neural networks (FNN) and a one-hot decoding function. The number of input nodes in the FNNs are 2560 and 440, respectively. The number of output nodes of the second FNN is dependent on the number of parameters times the number of discrete values. After the first FNN, the LReLU activation function is used. The resulting information is passed to the final FNN. A reshaping operation is applied to the vector containing the log-probabilities computed by the last FNN layer, which rearranges the previously mentioned vector into a matrix with dimensionality consisting of the number of parameters times the number of quantized parameter values. The soft-max activation function is computed with respect to the number of discrete steps for each parameter and further decoded to recover the corresponding numerical DAFX values. The one-hot encoding/decoding steps are used only for representing the numerical values of the DAFX operators as categorical variables, which we found more convenient for modeling generic DAFX.

## 3. DATA GENERATION

Based on our proposed modifications to the Siamese model in Sec.2.3–2.4 that allow learning from unpaired data, we employ a self-supervised learning approach to optimize the parameters of our model. In Algorithm 1, we outline the complete data generation process, i.e., generating a dataset $\mathcal{D}_O$ for self-supervised learning using an input dataset $\mathcal{D}_I$ (e.g. speech recordings). By acoustic scene augmentation, we generally refer to any technique that can help simulate a diverse set of acoustic scene changes (i.e. background noise, reverberation, frequency equalization, etc.) that are relevant to the corresponding DAFX being modeled. By using this approach, we simultaneously avoid having to use any form of (expensive) human-labeled training data and are able to train our model using an effectively infinite number of unpaired style transfer examples. This is a significant advantage and opens up a wide range of production style transfer applications.

**Algorithm 1** Data generation for self-supervised learning

```
Require: Input recording dataset $\mathcal{D}_I$, iteration count $M$
 1: Initialize empty data-set: $\mathcal{D}_O \leftarrow \{\}$
 2: for all $M$ do
 3:     Randomly sample $X$ and $Y$ from $\mathcal{D}_I$.
 4:     Apply identical acoustic scene augmentation to $X, Y$.
 5:     Divide $X, Y$ into $N$ overlapping segments $X^{(i)}, Y^{(i)}$.
 6:     Shuffle the segments.
 7:     for all $N$ do
 8:         Randomly sample one-hot categorical parameters.
 9:         Compute $\Theta^{(i)}$ by one-hot decoding.
10:         Update each $Y^{(i)}$ using $\Theta^{(i)}$ and the DAFX.
11:         Append ($X^{(i)}$, $Y^{(i)}$, $\Theta^{(i)}$) to $\mathcal{D}_0$.
12:     end for
13: end for
14: return Dataset: $\mathcal{D}_O$
```

## 4. EXPERIMENTAL PROCEDURE

To demonstrate the power of our proposed method, we apply our method to the task of parametric frequency equalization (EQ). More specifically, we use our method to match the frequency EQ between two speech recordings using a multi-band parametric EQ. While parametric EQs are a commonly used for audio production, they are inherently difficult to optimize to fit a desired frequency response (non-convex), let alone perform matching between two speech signals [15, 16, 17, 18, 19]. For this task, we employ a parametric 4-band EQ [20] consisting of a low-shelving filter (Low), a low-mid range band pass filter (Mid1), a mid-high range bandpass filter (Mid2), and a high shelving filter (High). Each band has the following control parameters: center frequency (Hz), gain (dB), and quality factor (Q-factor) which controls the bandwidth of the filter.

All of the EQ parameters are discretized in each band to 21 values. The values for the Q-factor are logarithmically spaced in the interval [0.01 – 10], whereas the values for the gain parameter are linearly spaced in the interval of [-10, +10] dB. The center frequencies for all the bands are logarithmically spaced. For the Low band, the values lie in the interval of [85 – 255] Hz, [260 – 2000] Hz for the Mid1 band, [2005 – 4000] Hz for the Mid2 band, and [4005 – 16000] Hz for the High. We chose these values following common frequency settings available in commercial production tools for EQ, e.g. [21]. The implementation of the EQ is based on the open source project "yodel" [22].

We employ the "clean" and "clean-raw" subsets from the Device and Produced Speech (DAPS) dataset [23], sampled at 44.1kHz for our input dataset $\mathcal{D}_I$. The collection of the recordings is split into training, validation, and test sets. The splitting is performed on the file level to ensure each subset includes the same number of female and male speakers and, between speakers, the same number of scripts are used. The splitting ratios used are $50\%$ ($\sim$ 2.3 hours) for training, $10\%$ ($\sim$ 0.42 hours) for validation and $40\%$ ($\sim$ 1.75 hours) for testing. Each split is given to Algorithm 1 with $M = \{100, 80, 20\}$ for each split respectively. The random acoustic scene augmentation is based on random equalization via a separate finite impulse response Linkwitz-Riley style filterbank [24]. The length of each sub-band filter is 16,384 samples (370 ms), and the corresponding center frequencies for the filters are $200, 500, 2,000$, and $8,000$ Hz. The gain values for each band are randomly sampled from a uniform distribution in the range of [-10 – +10] dB. Segments with loudness below -55 dB LUFS are discarded from the data-sets.

The total number of parameters in our model is $1.2$ million. We randomly initialize the parameters of our model by sampling from a normal distribution and scaled according to [25]. Stochastic gradient descent is performed to minimize the negative log-likelihood using the Adam optimizer [26], with a learning rate set to $5\text{e}{-4}$ and a batch size equal to 8. After each update, we enforce the first convolutional layer to be loudness invariant by subtracting its current average value [27].

Every 2e3 weight updates, the model is evaluated using the validation set. The absolute errors for each band in the respective units of each parameter is computed. After 20 consecutive sets of 2e3 updates with non-decreasing error, we terminate the training of the model. The maximum number of weight updates is set to 2e5. At the end of optimization, we test our proposed method using the test sub-set and report the mean absolute error average (MAE) and standard deviation (STD) per frequency band.

## 5. RESULTS & DISCUSSION

We compare the estimation results using our proposed method and two baselines. The first baseline (BL-linear) is an adapted network from [8], which we modify for our parametric EQ estimation task and train on unpaired data. The second baseline is an updated version of the first baseline, but uses a decibel-valued Mel-spectrogram input (BL-log) as opposed to a linear-valued Mel-spectra. The mean absolute error (MAE) and standard deviation (STD) results for the EQ center frequencies, gains, and Q-factor are shown in Table 1, Table 2, and Table 3, respectively, for 3 second input samples.

By analyzing the results, we first notice that simply using a decibel-scaled Mel-spectrogram increases performance by a noticeable amount, particularly for the center frequency and Q-factor parameters. This effect, however, is much smaller for the gain parameter, which is a surprising result. We hypothesize, however, that because the gain parameters in all models perform relatively well, it is more difficult to improve the performance further. Second, for nearly all parameters, applying our learnable comparison module improves the results. This is particularly true for the center frequency, Q-factor parameters, and high-frequency bands of the gain and is consistent with our comparison analysis described above. Also, the improvement caused by the learnable comparison is more prevalent in the higher-frequency filter bands. From this observation and qualitative listening, we found this to be a consistent issue. We believe this is because low-frequency speech content can change dramatically depending on the linguistic information content in the recording. This is less prevalent for higher frequency content. In a sense, we found it is more difficult to disentangle the style and content of low-frequency speech EQ compared to high-frequency EQ.

Third, we show results of our model trained on 3-second data, but using 12-second input at inference time in Table 4. We do this to 1) illustrate the fact that our proposed model can handle varying length input and 2) try to improve the performance of the low frequency filter bands. When we compare these results to Tables 1-3, we see that the Q-factor results improve, but the center frequency and gain parameters are roughly comparable. We hypothesize that long-speech pauses cause issues and leave it to future work to improve the global time-averaging by training on varying length inputs and investigating an attention mechanism to better combine latent style features across time.

| | **Models** | | |
|---|---|---|---|
| **Band** | BL-linear [8] | BL-log [8] | Proposed |
| Low | 69.2 (±51.1) | 54.4 (±40.4) | **48.1** (±45.3) |
| Mid1 | 578.8 (±514.9) | 566.8 (±445.4) | **440.4** (±409.8) |
| Mid2 | 1.1k (±604.6) | 695.7 (±497.6) | **515.5** (±494.0) |
| High | 6.3k (±3.3k) | 3.5k (±2.7k) | **2.5**k (±2.9k) |

**Table 1**. Center frequency (Hz) MAE (± STD).

| | **Models** | | |
|---|---|---|---|
| **Band** | BL-linear [8] | BL-log [8] | Proposed |
| Low | 5.2 (±3.0) | 4.9 (±4.3) | **4.1** (±3.1) |
| Mid1 | 5.3 (±3.0) | 4.6 (±3.9) | **4.2** (±4.0) |
| Mid2 | 5.3 (±3.0) | 4.5 (±3.9) | **2.5** (±2.6) |
| High | 5.2 (±3.1) | 5.2 (±4.2) | **0.8** (±0.9) |

**Table 2**. Gain (dB) MAE (± STD).

| | **Models** | | |
|---|---|---|---|
| **Band** | BL-linear [8] | BL-log [8] | Proposed |
| Low | 1.7 (±2.6) | 2.3 (±2.8) | **1.2** (±1.6) |
| Mid1 | 5.7 (±2.0) | **2.1** (±2.5) | 2.1 (±2.8) |
| Mid2 | 8.3 (±2.7) | 2.7 (±2.9) | **1.8** (±2.9) |
| High | 1.7 (±2.7) | 2.1 (±2.9) | **0.5** (±1.2) |

**Table 3**. Q-factor MAE (± STD).

| | **Parameters** | | |
|---|---|---|---|
| **Band** | Freq. (Hz) | Gain (dB) | Q-factor |
| Low | 50.0 (±46.2) | 5.3 (±4.7) | 1.1 (±1.7) |
| Mid1 | 426.9 (±457.9) | 4.2 (±4.0) | 1.75 (±2.7) |
| Mid2 | 513.6 (±500.4) | 2.2 (±2.2) | 1.3 (±2.5) |
| High | 2.5k (±2.9)k | 0.8 (±0.9) | 0.4 (±1.0) |

**Table 4**. MAE (± STD) with 12-second inputs at inference time.

## 6. CONCLUSIONS

In this work, we present a method to transform an untreated speech recording into the production style of a reference recording. Our method receives as input an untreated recording and an unrelated reference recording, and predicts a set of parameters. The parameters are then used to control digital audio effects, which are, in turn, used to transform the production style of the untreated recording into the style of the reference recording. We frame our work as a form of one-shot *parametric* audio product style transfer that, by design, leverages existing audio production semantics and manipulation algorithms. To demonstrate our approach, we consider the task of controlling a parametric, four-band infinite impulse response equalizer and show that we are able to predict the parameters necessary to transform the equalization style of one recording to another. For future work, we plan to investigate our proposed method on additional audio production effects, including speech enhancement and production tools (i.e. denoisers, dereverberators). For additional results see: `https://js-mim.github.io/sp-demo/`.

## 7. REFERENCES

[1] R. Izhaki, *Mixing Audio: Concepts, Practices and Tools*, Electronics & Electrical. Focal Press, 2008.

[2] Y. Zhicheng, Z. Hao, W. Baoyuan, P. Sylvain, and Y. Yizhou, "Automatic photo adjustment using deep neural networks," *ACM TOG*, vol. 35, no. 2, Feb. 2016.

[3] S. I. Mimilakis, K. Drossos, T. Virtanen, and G. Schuller, "Deep neural networks for dynamic range compression in mastering applications," in *AES Convention 140*. AES, 2016.

[4] M. A. Martínez Ramírez and J. D. Reiss, "Modeling nonlinear audio effects with end-to-end deep neural networks," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 171–175.

[5] S. H. Hawley, B. Colburn, and S. I. Mimilakis, "SignalTrain: Profiling audio compressors with deep neural networks," in *AES Convention 147*. AES, 2019.

[6] J. Rämö and V. Välimäki, "Neural third-octave graphic equalizer," in *DAFX*, 2019, p. 6.

[7] V. Välimäki and J. Rämö, "Neurally controlled graphic equalizer," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2140–2149, Dec 2019.

[8] D. Sheng and G. Fazekas, "A feature learning Siamese model for intelligent control of the dynamic range compressor," in *2019 International Joint Conference on Neural Networks (IJCNN)*, July 2019, pp. 1–8.

[9] E. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, May 2019, pp. 471–475.

[10] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *in International Conference on Machine Learning (ICML)*, 2015, ICML.

[11] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "Siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *in International Conference on Machine Learning (ICML)*, 2015, vol. 37 of *ICML*, pp. 448–456.

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[14] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. HS. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208.

[15] T. W. Parks and C. S. Burrus, *Digital filter design*, Wiley-Interscience, 1987.

[16] K. Steiglitz and L. McBride, "A technique for the identification of linear systems," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 461–464, 1965.

[17] E. C. Levy, "Complex-curve fitting," *IRE transactions on automatic control*, , no. 1, pp. 37–43, 1959.

[18] B. Friedlander and B. Porat, "The modified Yule-Walker method of ARMA spectral estimation," *IEEE Transactions on Aerospace and Electronic Systems*, , no. 2, pp. 158–173, 1984.

[19] A. Rimell and M. J. Hawksford, "The application of genetic algorithms to digital audio filters," in *AES Convention 98*. AES, 1995.

[20] J. O. Smith, *Introduction to Digital Filters with Audio Applications*, W3K Publishing, http://www.w3k.org/books/, 2007.

[21] Adobe Systems, "Adobe Audition CC 2019," 2019.

[22] R. Clement, "Yodel," `https://pypi.org/project/yodel/`, 2014.

[23] G. J. Mysore, "Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech? A Dataset, insights, and challenges," *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1006–1010, 2014.

[24] S. H. Linkwitz, "Active crossover networks for noncoincident drivers," *Journal of the Audio Engineering Society*, vol. 24, no. 1, pp. 2–8, 1976.

[25] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010, pp. 249–256.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR-15*, 2015.

[27] J. Schlüter and B. Lehner, "Zero-mean convolutions for level-invariant singing voice detection," in *ISMIR*, Paris, France, 2018, pp. 321–326.