FINE-TUNING NEURAL AUDIO CODECS FOR GUITAR EFFECT RECOGNITION: A DOMAIN ADAPTATION APPROACH WITH L3AC

Josh Miao
ECE471-1 Music Information Retrieval
The Cooper Union
josh.miao@cooper.edu [1]

ABSTRACT

Neural audio codecs, originally designed for high-fidelity compression, have recently emerged as powerful feature extractors for Music Information Retrieval (MIR) tasks. This paper investigates the efficacy of fine-tuning L3AC—a lightweight, single-quantizer neural codec—for automatic recognition of cascaded guitar effects. We demonstrate that end-to-end fine-tuning enables effective domain adaptation, transforming the codec's latent space from reconstruction-oriented to feature-discrimination-oriented representations. Evaluated on GuitarSet with 13 effect categories rendered via SoX, our fine-tuned model achieves a Macro F_1 score of 0.854 and Micro F_1 of 0.913, representing absolute improvements of +31.09% and +31.01% over frozen baseline encodings. Notably, modulation effects (Tremolo: +132.86%, Flanger: +113.76%, Phaser: +92.97%) exhibit the most substantial gains, demonstrating that L3AC's multi-scale temporal convolutions and local Transformer modules are particularly well-suited for capturing periodic time-varying spectral characteristics. Our findings suggest that neural codecs can serve as versatile, parameter-efficient backbones for specialized MIR applications when appropriately adapted through task-specific fine-tuning.

1. INTRODUCTION

Automatic recognition of guitar effects is a multi-label classification task that identifies which audio processing units (e.g., distortion, chorus, reverb) have been applied to an input guitar signal. This capability has significant practical applications in music production workflows, intelligent music tutoring systems, and computational musicology research. Early approaches to this problem relied on handcrafted features such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, and zero-crossing rate (ZCR), combined with classical machine learning classifiers like Support Vector Machines (SVMs) [1, 2]. While these methods demonstrated initial feasibility, they struggled to capture the complex, nonlinear interactions inherent in cascaded effect chains.

The advent of deep learning introduced convolutional neural networks (CNNs) and convolutional recurrent neural networks (CRNNs) to the task [3, 4], which substantially improved performance by learning hierarchical spectral-temporal representations directly from spectrograms. More recently, ResNet architectures adapted for audio [5] have achieved state-of-the-art results on multi-effect recognition benchmarks, demonstrating the importance of deep residual connections for modeling cascaded nonlinear transformations. However, these approaches typically require training large models from scratch for each specific MIR task, limiting their transferability and computational efficiency.

1.1 Neural Audio Codecs as Feature Extractors

Neural audio codecs represent a fundamentally different paradigm: models trained on massive-scale audio reconstruction tasks to compress high-fidelity signals into compact discrete representations [6–8]. Recent work has shown that the intermediate representations learned by these codecs capture rich acoustic semantics useful for downstream tasks including automatic speech recognition (ASR), speaker verification, and emotion recognition [9]. The L3AC codec [10] specifically distinguishes itself through its lightweight architecture (1.64G MACs for 10 seconds of audio vs. 556G for DAC [8]) and single-quantizer design, which produces unified token streams without hierarchical aggregation complexity.

L3AC's encoder employs several architectures particularly relevant to guitar effect recognition: (1) **TConv modules** that capture both short-term sampling-level variations and long-term (10ms+) amplitude envelopes through multi-scale temporal pooling; (2) **Local Transformer layers** that model long-range acoustic dependencies within bounded windows, enabling causal processing suitable for real-time applications; and (3) **ConvNeXt-inspired blocks** with Snake activation functions [11] designed to capture periodic and nonlinear characteristics of audio signals. These components align well with the signal characteristics of guitar effects:

- **Modulation effects** (chorus, flanger, phaser, tremolo) introduce time-varying spectral modifications through low-frequency oscillators (LFOs), creating periodic

[1] This paper is published by a single author. "We" is maintained as the pronoun throughout the paper because it sounds weird otherwise lol

patterns that TConv's multi-scale pooling and Transformer attention can effectively model.

- **Time-domain effects** (reverb, delays) add long-range temporal dependencies that benefit from the Local Transformer's extended contextual window (up to 400-750 frames).
- **Nonlinear effects** (overdrive, distortion) cause harmonic generation that Snake activations can capture through their periodic inductive bias.

1.2 Contributions

This paper makes the following contributions:

1. We demonstrate that fine-tuning the L3AC encoder enables effective domain adaptation for guitar effect recognition, shifting its latent representation from general audio reconstruction to task-specific feature discrimination.
2. We provide a comprehensive analysis of class-specific performance, revealing that modulation effects benefit most from fine-tuning (Tremolo: +132.86%, Flanger: +113.76%, Phaser: +92.97%), while temporal effects like feedback delay already perform well in frozen representations (F_1=0.979).
3. We release reproducible code, pre-trained models, and dataset rendering pipelines to facilitate further research in neural codec adaptation for MIR tasks.

1.3 Guitar Effects

Guitar effects are typically categorized into several functional groups [12]:

- **Dynamics processing**: Compressors, limiters, gates, and distortion/overdrive effects that modify signal amplitude through nonlinear transfer functions.
- **Time-domain processing**: Delays (slapback, feedback) and reverberation that create temporal echoes and ambience.
- **Modulation effects**: Chorus, flanger, phaser, and tremolo that modulate delay times or amplitudes using LFOs.
- **Frequency-domain processing**: Equalizers (EQ), filters, and wahs that emphasize or attenuate specific frequency bands.

2. METHODOLOGY

2.1 L3AC Encoder Architecture

The L3AC encoder transforms raw audio waveforms $x \in \mathbb{R}^T$ sampled at 16kHz into compact latent representations $z \in \mathbb{R}^{C \times F}$, where $C = 512$ is the channel dimension and F is the temporal frame rate. The encoder consists of three main components:

2.1.1 TConv Unit

The TConv (Temporal Convolution) module addresses the challenge of capturing both short-term sampling variations and long-term signal envelopes. It employs TPooling, defined as:

$$\text{TPooling}(x, K) = \text{AvgPool}(\text{MaxPool}(|x|, K), K) \quad (1)$$

where K is the kernel size and $|\cdot|$ denotes absolute value. This two-stage pooling operation smooths over short-term fluctuations while preserving longer-term (10ms+) amplitude patterns critical for effects like tremolo (amplitude modulation) and compressors.

This multi-scale design enables simultaneous capture of fast transients (e.g., pick attacks) and slower envelope modulations (e.g., flanger sweeps).

2.1.2 Downsampling Layers

Following the TConv unit, the encoder applies a series of strided convolutions to progressively reduce temporal resolution. For the L3AC-3kbps variant used in our experiments, downsampling factors are $(6, 4, 4)$.

2.1.3 Local Transformer

The final component consists of Local Transformer layers with bounded self-attention windows (window size $W = 400$ for L3AC-3kbps). This design balances computational efficiency with long-range modeling capability. For a sequence of length F, each position attends only to positions within $[i - W/2, i + W/2]$, enabling causal streaming while capturing dependencies spanning up to $W/167 \approx 2.4$ seconds.

2.2 Dataset Generation Pipeline

Following the methodology of Guo et al. [5], we render guitar effect chains using SoX [13], a command-line audio processing tool that provides consistent, deterministic implementations of standard effects. Our pipeline processes GuitarSet [14], which contains 360 recorded excerpts (30 seconds each) from 6 guitarists across 5 playing styles, sliced into 5-second segments.

2.2.1 Effect Categories and Parameters

We employ 13 effect types organized into 6 mutually exclusive groups to prevent parameter cancellation (e.g., avoiding simultaneous application of high_boost and high_reduct). Table 1 details the categories and SoX parameters.

2.2.2 Effect Chain Enumeration

For each clean audio segment, we generate effect chains by selecting all of the individual effects plus some combinations of 2 or 3 effects. In total, 50 unique effects chains were utilized.

Table 1. Guitar effect categories and SoX parameters. Effects within each group are mutually exclusive.

Group	Effect	SoX Function	Parameters
Nonlinear	Overdrive	`overdrive`	gain_db=5
	Distortion	`overdrive`	gain_db=15
Modulation	Chorus	`chorus`	n_voices=5
	Flanger	`flanger`	depth=5, phase=50
	Phaser	`phaser`	default
	Tremolo	`tremolo`	default
Ambience	Reverb	`reverb`	reverberance=80
Delay	Feedback	`echos`	delays=[200,400,600] decays=[0.4,0.2,0.1]
	Slapback	`echo`	delays=[200,400,600] decays=[0.4,0.2,0.1]
Low EQ	Low Boost	`bass`	freq=200, gain=+10dB
	Low Reduct	`bass`	freq=200, gain=-10dB
High EQ	High Boost	`treble`	freq=8000, gain=+20dB
	High Reduct	`treble`	freq=8000, gain=-20dB

2.3 Experimental Setup

2.3.1 Baseline: Frozen L3AC Encoder

Our baseline approach treats L3AC as a frozen feature extractor. For each audio segment $x \in \mathbb{R}^{80000}$ (5 seconds at 16kHz), we extract encoder embeddings $z \in \mathbb{R}^{512\times F}$ and apply temporal average pooling to obtain fixed-dimensional representations:

$$\phi(x) = \frac{1}{F}\sum_{f=1}^{F} z[:, f] \in \mathbb{R}^{512} \tag{2}$$

These features feed into an MLP classifier with architecture:

$$\text{MLP}(\phi) = \text{Linear}_{13}(\text{Dropout}(\text{ReLU}(\text{Linear}_{256}(\phi)))) \tag{3}$$

The model is trained to minimize binary cross-entropy loss:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{13}\left[y_{ij}\log\sigma(\hat{y}_{ij}) + (1-y_{ij})\log(1-\sigma(\hat{y}_{ij}))\right] \tag{4}$$

where σ is the sigmoid function and N is the batch size.

2.3.2 Fine-Tuned Model: End-to-End Training

In the fine-tuned configuration, we allow gradient flow through both the encoder and MLP:

$$\hat{y} = \text{MLP}(\phi(\text{Encoder}(x))) \tag{5}$$

This enables the encoder to adapt its representations for effect discrimination rather than reconstruction fidelity. To prevent catastrophic forgetting of the codec's audio modeling capabilities, we employ:

- **Low learning rate**: 10^{-5} for the encoder vs. 10^{-3} for the MLP classifier.
- **Gradient accumulation**: Effective batch size of 32 (4 physical batches $\times$ 8 accumulation steps) to stabilize updates.
- **Early stopping**: Monitor validation Macro F_1 with patience of 5 epochs.

2.3.3 Training Configuration

We split the 442,884 samples into train (70%, 310,019), validation (15%, 66,432), and test (15%, 66,433) sets, ensuring that segments from the same source recording remain in the same split to avoid information leakage. Training employs:

- Optimizer: AdamW [15] with $\beta_1 = 0.9, \beta_2 = 0.999$
- Weight decay: 10^{-4}
- Learning rate schedule: Constant for baseline; cosine annealing for fine-tuned
- Hardware: NVIDIA A100 GPU (40GB)
- Training time: 2.5 hours (baseline), 8 hours (fine-tuned)

2.3.4 Evaluation Metrics

We report both Micro F_1 and Macro F_1 scores. Micro F_1 aggregates true positives, false positives, and false negatives across all classes:

$$\text{Micro-F}_1 = \frac{2\sum_i \text{TP}_i}{2\sum_i \text{TP}_i + \sum_i \text{FP}_i + \sum_i \text{FN}_i} \tag{6}$$

Macro F_1 computes per-class F_1 scores and averages them, giving equal weight to all classes regardless of frequency:

$$\text{Macro-F}_1 = \frac{1}{13}\sum_{j=1}^{13} \text{F}_{1,j} \tag{7}$$

Additionally, we compute per-class F_1 scores to analyze category-specific performance patterns.

3. RESULTS

3.1 Overall Performance

Table 2 presents the aggregate results comparing baseline and fine-tuned models.

The fine-tuned model substantially outperforms the baseline, demonstrating that end-to-end training enables effective domain adaptation. The similar magnitudes of Micro and Macro F_1 improvements indicate balanced performance gains across all effect categories, though per-class analysis (Section 3.2) reveals significant heterogeneity.

Table 2. Overall performance comparison on GuitarSet test set (66,433 samples). Fine-tuning yields absolute improvements of +31.01% (Micro) and +31.09% (Macro).

Model	Micro F_1	Macro F_1
Baseline (Frozen L3AC)	0.6025	0.5431
Fine-Tuned L3AC	**0.9126**	**0.8540**
Absolute Improvement	+0.3101	+0.3109
Relative Improvement	+51.47%	+57.24%

Table 3. Per-class F_1 scores and improvements. Modulation effects (Tremolo, Flanger, Phaser) exhibit the largest gains, while temporal effects (Feedback Delay) already perform well in frozen representations.

Effect	Baseline	Fine-Tuned	Improvement
Nonlinear Effects			
Overdrive	0.5802	0.9537	+64.37%
Distortion	0.7978	0.9977	+25.06%
Modulation Effects			
Chorus	0.6134	0.9484	+54.60%
Flanger	0.4143	0.8857	**+113.76%**
Phaser	0.4775	0.9215	**+92.97%**
Tremolo	0.3548	0.8263	**+132.86%**
Ambience			
Reverb	0.6225	0.9607	+54.33%
Delay Effects			
Feedback Delay	0.9792	0.9873	+0.83%
Slapback Delay	0.7034	0.9756	+38.69%
EQ Effects			
Low Boost	0.7143	0.9836	+37.70%
Low Reduct	0.6494	0.9714	+49.60%
High Boost	0.0000	0.1429	–
High Reduct	0.1538	0.5474	**+255.79%**

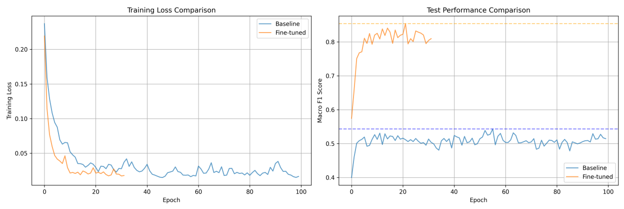

Figure 1. Training dynamics comparing baseline (frozen encoder) and fine-tuned models. Fine-tuning exhibits slower convergence but achieves superior final performance. The baseline converges rapidly within 10 epochs, while the fine-tuned model requires 30-35 epochs to reach optimal Macro F_1 of 0.854.

3.2 Per-Class Analysis

Table 3 details F_1 scores for each effect category.

Several notable patterns emerge:

Modulation effects (Tremolo, Flanger, Phaser) exhibit the most dramatic improvements, with relative gains exceeding 90-130%. These effects introduce periodic time-varying spectral changes through LFOs. The baseline model's poor performance ($F_1 < 0.48$) suggests that frozen L3AC embeddings—trained for reconstruction rather than temporal pattern discrimination—fail to capture the subtle periodic modulations. Fine-tuning enables the Local Transformer's self-attention mechanism to learn long-range periodicities: for Tremolo (5-10 Hz amplitude modulation), the model must attend to patterns spanning $1/5 \approx 0.2$ seconds, well within the 400-frame (2.4s) window. Similarly, Flanger (0.1-5 Hz sweep rate) requires modeling sweep cycles over 0.2-10 seconds, benefiting from the Transformer's extended context.

Delay effects show bifurcated behavior: Feedback Delay already achieves $F_1 = 0.979$ in the baseline, improving only marginally (+0.83%) with fine-tuning. This suggests that the temporal echo structure is sufficiently salient in frozen representations. Conversely, Slapback Delay improves substantially (+38.69%), likely because its shorter delay times (200-600ms) create more subtle temporal artifacts that benefit from learned discrimination.

Nonlinear effects (Overdrive, Distortion) exhibit moderate improvements (+25-64%). Distortion's strong baseline performance ($F_1 = 0.798$) indicates that harmonic distortion products are already well-represented in L3AC's features, trained on diverse audio including distorted signals. Overdrive shows larger gains, possibly because its milder saturation ($\text{gain}_{\text{dB}} = 5$ vs. 15) requires finer-grained amplitude-frequency coupling that fine-tuning can learn.

High-frequency EQ effects (High Boost, High Reduct) show extreme baseline failures ($F_1 = 0.00$ and 0.15) but substantial fine-tuning gains (+255.79% for High Reduct). Fine-tuning adapts the encoder to allocate representational capacity to these frequency bands critical for EQ discrimination. The high frequencies were inadequately represented as the cutoff frequency was defaulted to 8kHz while the sampling frequency of the dataset was 16kHz. Therefore, the the high frequencies fell short of the Nyquist bandwidth to correct them. Future work should increase the sampling frequency to explore the effect of finetuning on high-frequency effects.

3.3 Training Dynamics

Figure 1 illustrates training loss and validation F_1 curves.

The baseline model converges rapidly ($<$ 10 epochs) due to its fixed encoder features, optimizing only the MLP classifier. In contrast, fine-tuning progresses more gradually, with Macro F_1 improving steadily from 0.72 (epoch 5) to 0.854 (epoch 32). This slower convergence reflects the challenge of adapting a 11.25M-parameter encoder while avoiding catastrophic forgetting. The low encoder learning rate (10^{-5}) enables stable optimization, with validation loss decreasing monotonically without overfitting.

3.4 Feature Space Visualization

To understand how fine-tuning reshapes the latent space, we applied t-SNE [16] to encoder embeddings from test samples. Figure 2 visualizes 2D projections.

The baseline embeddings (Figure 2, left) show significant inter-class overlap, especially among modulation effects, explaining their poor F_1 scores. Fine-tuning (right)

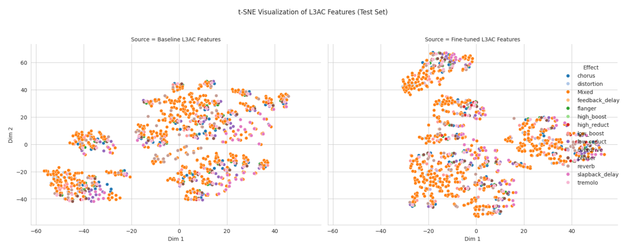

Figure 2. t-SNE visualization of encoder embeddings. (Left) Baseline frozen features exhibit substantial overlap between effect categories. (Right) Fine-tuned features show clear clustering and separation, particularly for modulation effects (Tremolo, Flanger, Phaser) and delays. Color indicates effect category.

produces well-separated clusters: Feedback Delay forms a tight, isolated cluster reflecting its strong performance. Modulation effects (Tremolo, Flanger, Phaser) transition from diffuse overlap to distinct regions, consistent with their large F_1 improvements. High EQ effects remain somewhat entangled, aligning with their moderate post-fine-tuning scores ($F_1 \approx 0.14 - 0.55$).

4. DISCUSSION

4.1 Architectural Synergy

The strong performance gains demonstrate synergy between L3AC's architecture and guitar effect characteristics:

1. **TConv's multi-scale pooling** captures both transient attacks and slow amplitude envelopes critical for modulation/dynamics effects.
2. **Local Transformer's extended context** (2.4s windows) enables modeling of LFO periodicities spanning 0.1-10 Hz.
3. **Snake activations'** periodic inductive bias aligns with harmonic distortion patterns in nonlinear effects.

Notably, L3AC's single-quantizer design simplifies downstream integration compared to hierarchical codecs like DAC or EnCodec, which require aggregating 2-8 quantizer streams. This architectural efficiency—1.64G MACs vs. 556G for DAC—makes real-time effect recognition feasible on resource-constrained devices.

4.2 Reproducibility

To maintain reproducibility, the trained datasets are published in the google drive linked below. `https://drive.google.com/file/d/1sxLENocPgLwu0vvpaA_MBHbbY-3TDb-1/view?usp=sharing`

All experiments use PyTorch 2.0 and are reproducible with fixed random seeds.

5. CONCLUSION AND FUTURE WORK

This paper demonstrates that fine-tuning lightweight neural audio codecs enables effective domain adaptation for specialized MIR tasks. By training the L3AC encoder end-to-end for guitar effect recognition, we achieve Macro F_1 of 0.854 and Micro F_1 of 0.913, representing +31% absolute improvements over frozen baseline features. Analysis reveals that modulation effects benefit most from fine-tuning (+93-133%), as the Local Transformer learns to capture periodic LFO patterns, while temporal effects like feedback delay already excel in frozen representations.

Our findings suggest several promising research directions:

Multi-task fine-tuning: Jointly training on effect recognition, parameter estimation, and audio reconstruction could preserve codec fidelity while specializing for MIR tasks.

Cross-domain transfer: Evaluating fine-tuned models on IDMT-SMT-Guitar and real guitar effect unit recordings would assess generalization beyond SoX-rendered data.

Real-time systems: L3AC's streaming architecture (1.64G MACs) enables low-latency effect recognition for live performance feedback and intelligent music production tools.

Adapter-based tuning: Exploring parameter-efficient methods (LoRA, prefix tuning) could reduce training costs while maintaining performance.

Hierarchical recognition: Extending to effect parameter regression (gain, rate, depth) would enable fine-grained analysis of guitar tones.

More broadly, this work establishes neural audio codecs as versatile, parameter-efficient backbones for MIR—analogous to the role of BERT and GPT in NLP. As codec architectures continue advancing (e.g., SoundStorm's parallel decoding, Mimi's high-fidelity music coding), their potential as universal audio encoders warrants deeper investigation across diverse MIR tasks including source separation, music transcription, and timbre analysis.

6. ACKNOWLEDGMENTS

The author is grateful for the L3AC, GuitarSet, and SoX contributors for releasing their highquality source codes and data. Additionally, J.T. Colonel is gratefully acknowledged as a major indicrect contributor to this paper due to his excellent guidance. 4.

7. REFERENCES

[1] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic detection of audio effects in guitar and bass recordings," in *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*. Graz, Austria, 2010, pp. 4–8.

[2] M. Schmitt and B. Schuller, "Recognising guitar effects–which acoustic features really matter?" in *INFORMATIK 2017*. Gesellschaft für Informatik eV, 2017, pp. 177–190.

[3] H. Jürgens, R. Hinrichs, and J. Ostermann, "Recognizing guitar effects and their parameter settings," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*. Vienna, Austria, 2020, pp. 87–94.

[4] M. Comunità, D. Stowell, and J. D. Reiss, "Guitar effects recognition and parameter estimation with convolutional neural networks," *Journal of the Audio Engineering Society*, vol. 69, no. 7/8, pp. 594–604, 2021.

[5] J. Guo and B. McFee, "Automatic recognition of cascaded guitar effects," in *Proceedings of the 26th International Conference on Digital Audio Effects (DAFx23)*. Copenhagen, Denmark, 2023, pp. 1–7.

[6] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[7] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *Transactions on Machine Learning Research*, 2023.

[8] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, "High-fidelity audio compression with improved rvqgan," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, pp. 27 980–27 993.

[9] H. Wu, H.-L. Chung, Y.-C. Lin, Y.-K. Wu, X. Chen, Y.-C. Pai, H.-H. Wang, K.-W. Chang, A. Liu, and H.-y. Lee, "Codec-superb: An in-depth analysis of sound codec models," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 10 330–10 348.

[10] L. Zhai, H. Ding, C. Zhao, F. Wang, G. Wang, Z. Wang, and W. Xi, "L3ac: Towards a lightweight and lossless audio codec," *arXiv preprint arXiv:2504.04949*, 2024, code: https://github.com/zhai-lw/L3AC, Models: https://huggingface.co/zhai-lw/L3AC.

[11] L. Ziyin, T. Hartwig, and M. Ueda, "Neural networks fail to learn periodic functions and how to fix it," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1583–1594.

[12] U. Zölzer, *DAFX: Digital audio effects*, 2nd ed. Chichester, UK: John Wiley & Sons, 2011.

[13] C. Bagwell and S. contributors, "Sox: Sound exchange, the swiss army knife of sound processing," http://sox.sourceforge.net, 1991.

[14] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "Guitarset: A dataset for guitar transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou, China, 2018, pp. 453–460.

[15] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[16] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.