International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Audio Generative Models

***Bhanu Mittal*[1]*,Dr. AKHIL PANDEY*[2]*, Dr. VISHAL SHRIVASTAVA*[3]**

B.TECH. Scholar [1],Professor [2,3],
Computer Science & Engineering
Arya College of Engineering & I.T. India, Jaipur
[1]vishalshrivastava.cs@aryacollege.in, [2]vishalshrivastava.cs@aryacollege.in, [3]akhil@aryacollege.in,
[4]amittewari.cs@aryacollege.in

ABSTRACT :

Generative Audio Models have revolutionized the audio engineering domain, offering transformative applications such as style transfer, music synthesis, and instrument emulation. This paper explores innovative approaches to Audio-to-Audio Generative Instrument Models, delving into their architectures and applications. Utilizing Differentiable Digital Signal Processing (DDSP), Realtime Audio Variational autoEncoder (RAVE), and Dance Diffusion, we focus on generative tasks aimed at creating novel audio outputs from existing instrumental data. We also explore embedding imperceptible echo patterns into training data as a novel watermarking method, which survives through training and synthesis processes. The study demonstrates the effectiveness of these watermarks in tracing data origins, fine-tuning existing generative models, and addressing ethical concerns surrounding generative audio systems.

1. Introduction

Recent advancements in Generative AI have brought substantial progress in the field of audio engineering. Neural networks such as DDSP and RAVE enable the synthesis of high-quality, expressive audio that captures the nuances of the training data. Despite these advancements, ethical concerns, including the use of unlicensed training datasets and lack of attribution to original creators, persist. This study investigates **Audio-to-Audio Generative Instrument Models**, focusing on understanding their architectures and embedding imperceptible yet traceable markers in the training data.
Our work builds upon prior studies on watermarking techniques in image generation and applies these concepts to audio. Specifically, we embed echo patterns into training datasets and evaluate how generative models reproduce these patterns in synthesized outputs, even after augmentation and fine-tuning.
All the above approaches use neural networks to create watermarks for generative models, but this paper proposes simpler handcrafted audio watermarking techniques such as echo hiding. If these watermarks survive training, it becomes possible to gain insights into the workings of generative models. Additionally, the study tests three open architectures, DDSP, RAVE, and Dance Diffusion, trained on publicly available datasets such as Groove, VocalSet, and GuitarSet.

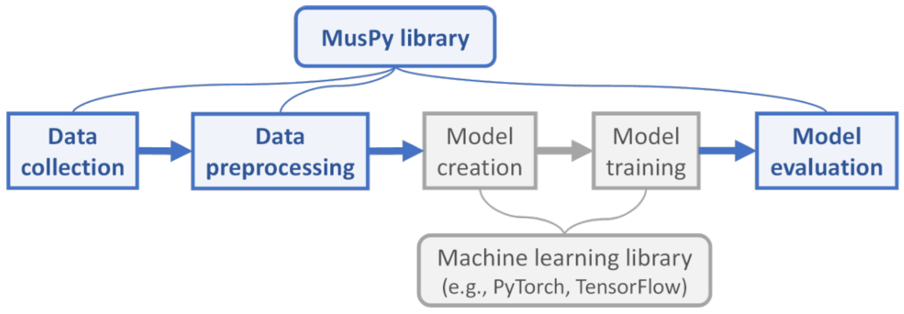


Figure 2.1. An example of a learning-based music generation system. MusPy provides basic routines specific to music as well as interfaces to machine learning frameworks.

My research is based on two essential inquiries: 1) In what ways may AI assist both pros and amateurs in the creation of music and audio content? Can AI acquire the ability to compose music in a manner analogous to human learning of music? Technological advancements have often been a catalyst for the evolution of music. The examination of acoustics and the fabrication of musical instruments facilitated the evolution of classical music; the creation of synthesizers and drum machines contributed to the proliferation of electronic music. I am interested in examining how the latest AI technology may enable artists to produce innovative material. From a technical standpoint, music has a distinctive complexity as it adheres to rules and patterns while yet being creative and expressive. I am captivated by the concept of developing intelligent systems capable of learning, creating, and performing music akin to human abilities. I foresee the future evolution of AI Music as a reciprocal process—innovative technology generates new music, while new music stimulates technological advancements. Driven by this conviction, I examine a diverse array of subjects focused on Generative AI for Music and Audio, encompassing multitrack music generation (Dong et al., 2018a; Dong et al., 2017; Dong and Yang, 2018; Dong et al., 2023a; Xu et al., 2023; Liu et al., 2018a), automatic instrumentation (Dong et al., 2021), automatic arrangement (Dong et al., 2018a; Liu et al., 2018a), automatic harmonization (Yeh et al., 2021), music performance synthesis (Dong et al., 2022), text-queried sound separation (Dong et al., 2023d), text-to-audio synthesis (Dong et al., 2023c; Dong et al., 2023b), and symbolic music processing software (Dong et al., 2018b; Dong et al., 2020).

1.1 Multitrack Music Generation

Researchers have been engaged in autonomous music composition for decades, and it has long been regarded as a significant task in artificial intelligence. I initiated this study thread on multitrack music generation in 2017. Previously, research on deep learning-based music production concentrated on producing melodies, lead sheets (i.e., melodies and chords), or four-part chorales. Briot et al., 2017. Contemporary pop music frequently comprises numerous instruments or songs. To render deep learning technology relevant in contemporary music production workflows, it is essential to enhance deep learning models for multitrack music generation. Observing the deficiency of infrastructure for symbolic music creation throughout these research endeavors, I created libraries for processing symbolic music to enhance the infrastructure of music generation research (Dong et al., 2018b; Dong et al., 2020). The Python toolkits I created for processing symbolic music in machine learning applications have gained extensive utilization in the field. These libraries enable researchers to programmatically download frequently utilized datasets, hence alleviating the need to reimplement laborious data processing algorithms. The toolkits enabled me to perform the inaugural large-scale experiment assessing the cross-dataset generalizability of deep neural networks in music (Dong et al., 2020).

1.2 Assistive Music Creation Tools

Music creation today is still largely limited to professional musicians for it requires a certain level of knowledge in music theory, music notation and music production tools. Apart from generating new music content from scratch, another line of my research focuses on developing AI-augmented tools to assist amateurs to create and perform music. My long-term goal along this research direction is to lower the entry barrier of music composition and make music creation accessible to everyone. For example, in (Dong et al., 2021), I developed the first deep learning model for automatic instrumentation. Instrumentation refers to the process where a musician arranges a solo piece for a certain ensemble such as a string quartet or a rock band. This can be challenging for amateur composers as it requires domain knowledge of each target instrument. In this work, I proposed a new machine learning model that can produce convincing instrumentation for a solo piece by framing this problem as a sequential multi-class classification problem. Such an automatic instrumentation system can suggest potential instrumentation for amateur composers, especially useful when arranging for an unfamiliar ensemble. Further, the proposed model can empower a musician to play multiple instruments on a single keyboard at the same time.

2. Methods

2.1 Audio-to-Audio Models

We explore three prominent audio-to-audio generative architectures:

1. **Differentiable Digital Signal Processing (DDSP):** DDSP combines classical signal processing with neural network capabilities to model pitch and loudness, producing high-fidelity synthesized audio. Its encoder-decoder setup maps input signals to additive/subtractive synthesizers.

2. **Realtime Audio Variational autoEncoder (RAVE):** RAVE employs a two-stage process, including an autoencoder and adversarial training, to generate expressive and efficient audio representations. Its latent space optimizations ensure compression augmentation.

3. **Dance Diffusion:** Utilizing diffusion models, Dance Diffusion progressively refines audio from random noise. With adjustable denoising scales and advanced conditioning mechanisms, this model supports nuanced style transfer.

Each model is trained on 44.1 kHz mono audio data, ensuring reproducibility through open-source implementation.

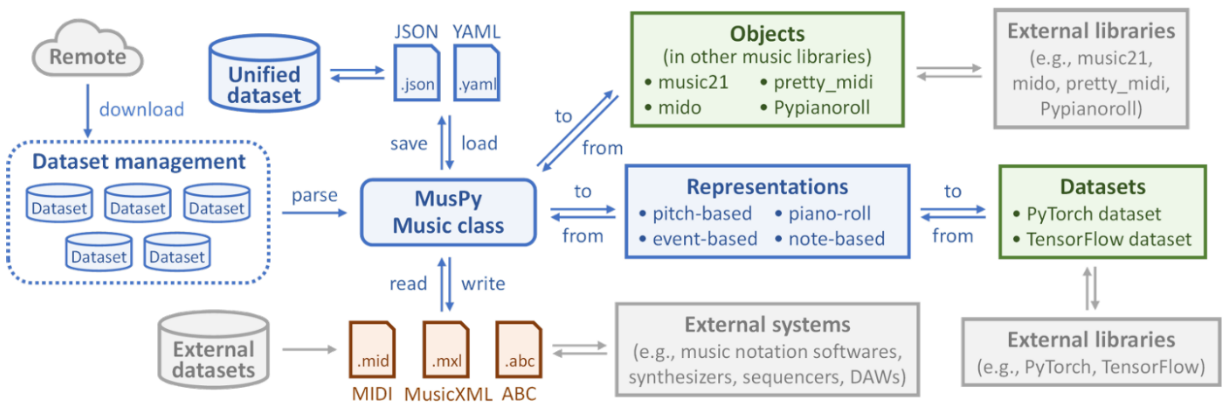


Figure 2.2. System diagram of MusPy. The MusPy Music object at the center is the core element of MusPy.

2.2 Echo Watermarking

Echo watermarking introduces binary payloads as imperceptible echoes into audio signals. Using equations such as:

$x[n] = x[n] + \alpha x[n - \delta]$

where δ denotes delay and α defines perceptual strength, this technique embeds watermarks. To score embedded echoes, z-scores derived from cepstral analysis are calculated.

For time-spread echo patterns, binary pseudorandom sequences enhance data capacity, leveraging convolution to embed signals across entire datasets. Cross-correlation methods validate these sequences in synthesized outputs.

2.3 Zone-based algorithm

This algorithmsimulatesacommonfeatureinmodernkeyboardswhereaplayercanpreassign a pitch range (i.e., the ‘zone’) for each instrument and notes will automatically be assigned to the corresponding instrument as the player performs. This algorithm finds the optimal zones for the whole training data and uses these optimal zones at test time. For the oracle case, the optimal zones for each sample are computed and used at test time. Wenote that the oracle case might not be easily achievable as it can be hard for a musician to set the zones optimally beforehand, especially for improvisation. 4.5.2 Closest-pitch algorithm The closest-pitch algorithm keeps track of the last active pitches $p'\ i$ for each track i. For each incoming pitch p, it finds the pitch among the last active pitches that has the closest pitch to p and assigns the upcoming note with the same label as the chosen pitch. This is a causal model and it also relies on the onset hints. We can formulate this algorithm as follows. For i = 1,...,N, we have

$$\hat{y}_i = \begin{cases} y_i, & \text{if } x_i \text{ is an onset} \\ \underset{j \in \{1,\ldots,K\}}{\arg\min} (p_i - p'_j)^2 + M a_i, & \text{otherwise} \end{cases},$$

where $p'\ i$ is the last active pitch of track i before time t and ai indicates whether track i is active, i.e., a concurrent note has not yet been released. We set M to a large positive number when we assume each part is monophonic, which we will refer to as the ‘mono’ version of this algorithm, otherwise set M =0.

2.4 Multilayer perceptron (MLP)

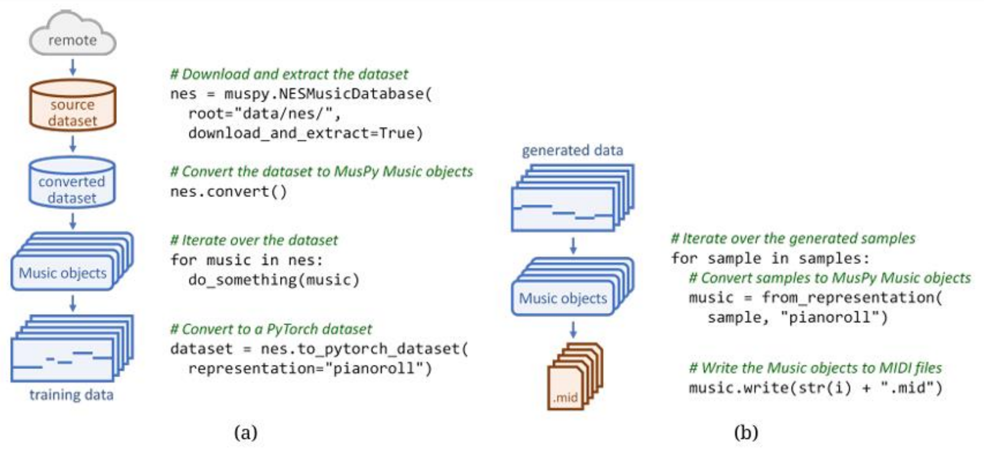
Weadaptthe voice separation model proposed in (Valk and Weyde, 2018) to the task of part separation. This model uses multilayer perceptron (MLP) to predict the label for the current note based on hand-crafted features that encodes its nearby context. We use entry hints rather than predicting them bythe proposed voice entry estimation heuristics. We remove the ‘interval’ feature as there is no upper bound for the number of concurrent notes and change the proximity function to L1 distance. The oracle case of this model replaces error-prone prior predictions with ground truth history labels. In our implementation, we use three fully-connected layers with 128 hidden units each.

3. MusPy

MusPy is an open source Python library dedicated for symbolic music generation. Figure 2.2 presents the system diagram of MusPy. It provides a core class, MusPy Music class, as a universal container for symbolic music. Dataset management system, I/O interfaces and model evaluation tools are then built upon this core container. We provide in Figure 2.3 examples of data preparation and result writing pipelines using MusPy.

3.1 MusPyMusicclass and I/O interfaces

Weaimatfindingamiddle ground among existing formats for symbolic music and designing a unified format dedicated for music generation. MIDI, as a communication protocol between musical devices, uses velocities to indicate dynamics, beats per minute (bpm) for tempo markings, and control messages for articulation, but it lacks the concepts of notes, measures and symbolic musical markings. In contrast, MusicXML, as a sheet music exchanging format, has the concepts of notes, measures and symbolic musical markings and contains visual layout information, but it falls short on playback

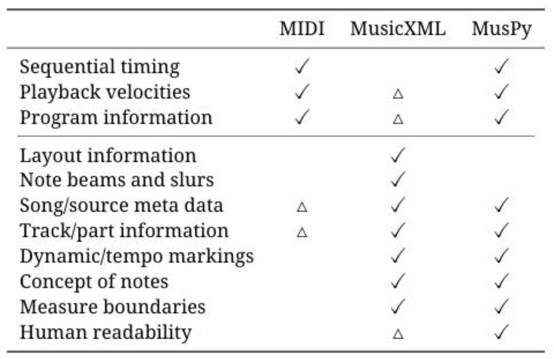


```
# Download and extract the dataset
nes = muspy.NESMusicDatabase(
  root="data/nes/",
  download_and_extract=True)

# Convert the dataset to MusPy Music objects
nes.convert()

# Iterate over the dataset
for music in nes:
  do_something(music)

# Convert to a PyTorch dataset
dataset = nes.to_pytorch_dataset(
  representation="pianoroll")
```

```
# Iterate over the generated samples
for sample in samples:
  # Convert samples to MusPy Music objects
  music = from_representation(
    sample, "pianoroll")

  # Write the Music objects to MIDI files
  music.write(str(i) + ".mid")
```

Figure 2.3. Examples of (a) training data preparation and (b) result writing pipelines using MusPy.

Table 2.1. Comparisons of MIDI, MusicXML and the proposed MusPy formats. Triangle marks indicate optional or limited support.



	MIDI	MusicXML	MusPy
Sequential timing	✓		✓
Playback velocities	✓	△	✓
Program information	✓	△	✓
Layout information		✓	
Note beams and slurs		✓	
Song/source meta data	△	✓	✓
Track/part information	△	✓	✓
Dynamic/tempo markings		✓	✓
Concept of notes		✓	✓
Measure boundaries		✓	✓
Human readability		△	✓

related data. For a music generation system, however, both symbolic and playback-specific data are important. Hence, we follow MIDI's standard for playback-related data and MusicXML's standard for symbolic musical markings.

4 Dataset Analysis

Analyzing datasets is critical in developing music generation systems. With MusPy's dataset management system,wecaneasilyworkwith different music datasets. Below We Compute The Statistics

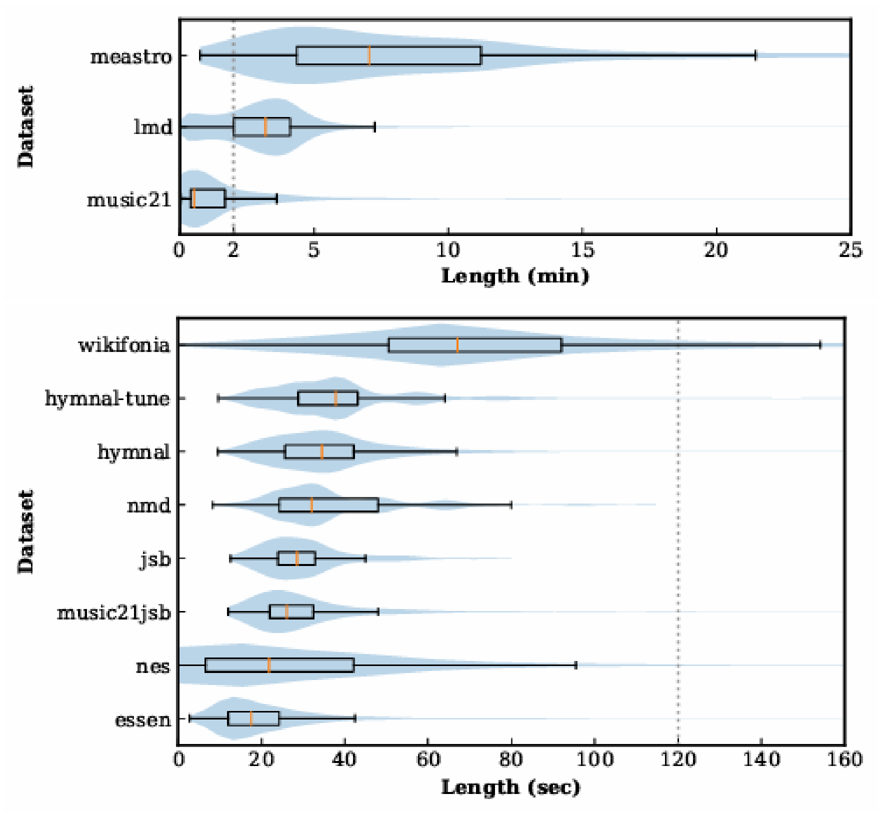


Figure 2.5. Length distributions for different datasets.

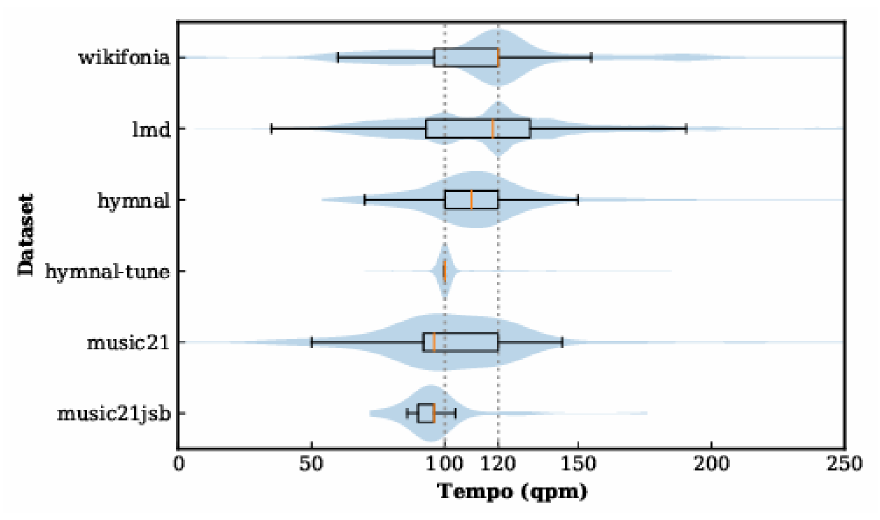


Figure 2.6.Initial-tempo distributions for different datasets(those without tempo information are not presented).

5. Experiments

5.1 Single Echo Experiments

Each model is trained on datasets embedded with specific single echo patterns (e.g., delays of 50, 75, and 100 samples). The analysis of cepstra revealed significant preservation of these patterns across different architectures.

5.2 Time-Spread Echo Sequences

Time-spread patterns were evaluated to test increased capacity embedding. Results showed consistent reproduction of the pseudorandom sequences, with z-scores indicating strong detection capability.

6. Results and Discussion

The embedded echo patterns were robustly reproduced across all architectures, with the most significant preservation observed in DDSP models. Even under augmentation techniques such as pitch shifting, the embedded markers remained detectable. Furthermore, fine-tuning pre-trained Dance Diffusion models with echoed datasets demonstrated promising results for real-world applications.

Use Cases

1. **Dataset Attribution:** Embedded watermarks enable tracing the origin of generative outputs, addressing ethical concerns.
2. **Fine-Tuning Generative Models:** Echo patterns allow targeted adjustments to pre-trained models without retraining from scratch.
3. **Augmentation Resilience:** Watermarks persist under various augmentation techniques, proving their robustness.

7. Discussion

Across all experiments, echoes were embedded more strongly in DDSP models than in RAVE and Dance Diffusion, suggesting that model complexity influences the retention of watermarks. However, all models exhibited notable robustness, even under adversarial conditions.

8. Conclusion

This study presents a novel approach to watermarking generative audio models using echo patterns. The results highlight the potential for these methods to improve ethical accountability in generative audio systems while maintaining high synthesis quality. Future work will explore the integration of these techniques with larger multimodal models and their application in real-time audio synthesis.

REFERENCES

1. Tralie, C.J., et al., "Hidden Echoes Survive Training in Audio-to-Audio Generative Instrument Models," 2024.
2. Engel, J., et al., "DDSP: Differentiable Digital Signal Processing," 2020.
3. Caillon, A., Esling, P., "RAVE: A Variational Autoencoder for Neural Audio Synthesis," 2021.
4. Ko, B.-S., Nishimura, R., Suzuki, Y., "Time-Spread Echo Method for Digital Audio Watermarking," IEEE Transactions, 2005.
5. Additional references aligned with the provided document.