DEEP RECOMBINANT TRANSFORMER: ENHANCING LOOP COMPATIBILITY IN DIGITAL MUSIC PRODUCTION

Muhammad Taimoor Haseeb* MBZUAI

Ahmad Hammoudeh* MBZUAI

Gus Xia MBZUAI

ABSTRACT

The widespread availability of music loops has revolutionized music production. However, combining loops requires a nuanced understanding of musical compatibility that can be difficult to learn and time-consuming. This study concentrates on the 'vertical problem' of music loop compatibility, which pertains to layering different loops to create a harmonious blend. The main limitation to applying deep learning in this domain is the absence of a large, high-quality, labeled dataset containing both positive and negative pairs. To address this, we synthesize high-quality audio from multi-track MIDI datasets containing independent instrument stems, and then extract loops to serve as positive pairs. This provides models with instrument-level information when learning compatibility. Moreover, we improve the generation of negative examples by matching the key and tempo of candidate loops, and then employing AutoMashUpper [1] to identify incompatible loops. Creating a large dataset allows us to introduce and examine the application of Transformer architectures for addressing vertical loop compatibility. Experimental results show that our method outperforms the previous state-of-the-art, achieving an 18.6% higher accuracy across multiple genres. Subjective assessments rate our model higher in seamlessly and creatively combining loops, underscoring our method's effectiveness. We name our approach the Deep Recombinant Transformer and provide audio samples [1] .

1. INTRODUCTION

The widespread availability of music loops used in Digital Audio Workstations (DAWs) has revolutionized music production. For example, *Umbrella* by Rihanna, composed using the "Vintage Funk Kit 03" GarageBand loop, transformed a royalty-free sample into a Grammy-winning global hit [2]. However, combining loops requires a nuanced understanding of musical compatibility and mostly relies on manual selection. Furthermore, the vast number of available loops presents a daunting challenge in deciding which loops pair well, leading to a combinatorial problem. Finding compatible loops was recognized as one of the grand challenges in MIR research [3].

The loop compatibility problem can be broken down into two sub-problems: the *vertical* problem and the *horizontal* problem. The vertical problem pertains to the layering of different loops — and understanding how rhythm, melody, and harmony interact within a single moment of music — to create a harmonious blend. Conversely, the horizontal problem addresses the sequencing of loops over time, ensuring that transitions between different loops are smooth and maintain the overall coherence of the musical piece. This research focuses on the vertical problem.

A major limitation to applying deep learning to this domain has been the absence of high-quality, labeled, datasets. Previous works propose source separating existing music, extracting loops from each stem, and creating positive pairs [4, 5]. Source separation models produce these four stems: *vocal, bass, drum,* and *other*. The outputs of source separation models are not perfect and often suffer from noise and distortion. Moreover, the "other" category can include a wide range of sounds — for example, entire string sections — and can be too *noisy* for the model to learn what makes two loops compatible. To generate negative samples, loop reversal, beat shifting, or key and tempo modifications are made to a loop in a positive pair. Altering loop characteristics to generate negative samples risks misleading models to distinguish these superficial differences rather than learning true musical incompatibility.

Our proposed solution to the above mentioned problems is to generate positive examples using MIDI datasets containing independent stems for each instrument, synthesizing them into audio, and extracting loops. This provides models with more granular information about *each* instrument when learning loop compatibility. Similarly, we find that while AutoMashUpper (AMU) demonstrates modest success in identifying compatible loops, its strength lies in accurately identifying incompatible loops after we match the tempo, key, and phase of query and target loops — thereby providing more realistic negative samples for model training [1]. Obtaining a large, high-quality, labeled dataset allows us to introduce and examine the application of Transformer-based architectures for addressing the vertical loop compatibility problem.

Our method outperforms the previous state-of-the-art for loop compatibility by 18.6% higher accuracy, proving its versatility and robustness across 13 genres through rigorous evaluations. Our contributions are as follows:

1. **A novel method to generate a large, high-quality, labeled dataset** for models to learn musical compatibility by providing positive and negative loop pairs that

* The first two authors contributed equally.

[1] Samples available at: https://conference-demo-2024.github.io/demo/

share identical keys, tempos, and phases.

2. **Transformer-based architectures** to enhance accuracy for instrument-level music loop compatibility.

3. **Extensive objective and subjective assessments** demonstrating our method's effectiveness.

2. RELATED WORK

Two approaches exist in the literature: rule-based and learning-based. Rule-based methods establish a set of rules to generate a compatibility score. In contrast, learning-based methods require positive and negative examples to train models for compatibility prediction. We review both.

2.1 Rule-Based Methods

Davies et al. set the groundwork for loop compatibility [6]. Their model, AutoMashUpper, computes mashability estimation by evaluating a weighted average of harmonic and rhythmic compatibility, and spectral balance across key-adjusted sections within a loop database. Best matching loops are aligned through time stretching and pitch shifting to match the query loop. Davies et al. introduce further improvements in a subsequent study, enhancing their system's capabilities [1]. Key improvements include the development of a faster algorithm for assessing harmonic similarity, integration of rhythm and loudness for mashability evaluation, and a subjective evaluation to assess the overall mashability of music pieces. Later works use this as a baseline to compare loop compatibility performance.

Lee et al. introduce a framework incorporating both vertical and horizontal dimensions of musical segments to create harmonious mashups [7]. Features include tempo, beat-synchronous chromagram, chord signatures, Mel Frequency Cepstral Coefficients, and volume levels. The system uses a Group of Background Units (GBU) from a specific track, typically comprising multiple background units that adhere to prevalent structures found in popular music genres, forming the foundational layer of a mashup. It evaluates potential lead units to layer atop the established GBUs, which pivots on three factors: Harmonic Matching which determines the harmonic compatibility between lead and background units, Harmonic Change Balance monitors the rate of harmonic transitions between to reduce monotony, while Volume Weighting calibrates the audibility of lead units. The framework computes a vertical mashability score for each candidate pair and selects those with the highest compatibility. Tsuzuki et al. overlay vocal tracks from other artists who have performed the same piece, aligning them with the instrumental track [8].

Bernardes et al. assess the harmonic compatibility of musical tracks through small- and large-scale structures [9]. Small-scale compatibility is determined by blending dissonance and perceptual relatedness, derived from the Tonal Interval Space [10], resilient to instrumental timbral variations. Large-scale compatibility is based on key estimations, aiding in overarching harmonic planning. Software showcases these metrics through interactive visualization to aid in finding harmonically compatible tracks. Maças et al. present MixMash by building on this method [9, 11]. MixMash enhances user interaction through a force-directed graph that visualizes multidimensional musical attributes like hierarchical harmonic compatibility, onset density, spectral region, and timbral similarity. The visualization represents tracks as nodes with varying distances and connections indicating their compatibility.

2.2 Learning-Based Methods

A major limitation in using Deep Neural Networks to evaluate the compatibility of musical loops has been the lack of adequately labeled datasets. Chen et al. are the first to use neural network models [4]. First, they propose an innovative pipeline to generate a labeled dataset using the Free Music Archive. To create positive samples they employ an unsupervised MSS algorithm that isolates looped content [12]. Negative samples are created by editing a loop in a positive loop pair by doing one of three things: reversing, randomly shifting beats, or rearranging beats of one of the loops. They propose using two architectures, a Convolution Network (CNN) and a Siamese Network (SNN), to learn compatibility between two loops. While both models outperform traditional rule-based systems, the CNN model demonstrates superior performance. Subsequent studies have identified limitations in the proposed data acquisition process. Specifically, it employs multiple heuristics for source separation and does not ensure the outputs consist of distinct instruments, e.g. a positive training example could comprise two similar drum loops [5]. In addition, they restrict their work to hip-hop without exploring how well this approach generalizes to other genres. Finally, they use a two-second input which may not capture the complexity and variability of longer musical pieces.

Huang et al. introduce an alternative method for assembling a training dataset by developing their own supervised music source separation model, which splits tracks into four distinct stems: *vocal, bass, drum,* and *other*. While it is an innovative approach, it leaves serious gaps in dataset quality. The outputs of source separation models are not perfect and often suffer from noise and distortion. In addition, the "other" category can include a wide range of sounds — e.g. guitars, pianos, trumpets, saxophones, violins, cellos, ambient sounds, synthesizers, reverb, choirs, flutes, clarinets, and even entire string sections. Since professional-grade musical loops contain distinct sounds — a guitar riff, a saxophone lick, a violin jig, etc. — using "other" may be too *noisy* for the model to learn what makes two instrument loops compatible. Similarly, as observed by Chen et al., bass and drum loops seamlessly adapt across most genres and styles when matched for tempo and key, and are somewhat trivial to learn for the model. On the other hand, they generate negative samples by varying the basic characteristics of a loop — key, tempo, or phase shifts. Even though it guarantees incompatibility, simply altering the basic loop characteristics to generate negative examples risks misleading the model to learn to distinguish these superficial differences rather than understanding true musical incompatibility. Instead, we propose training models to determine compatibility between loops sharing identical tempo, key, and phase to mirror choices made in actual music production — a significantly more challenging task. While there are some similarities, Huang

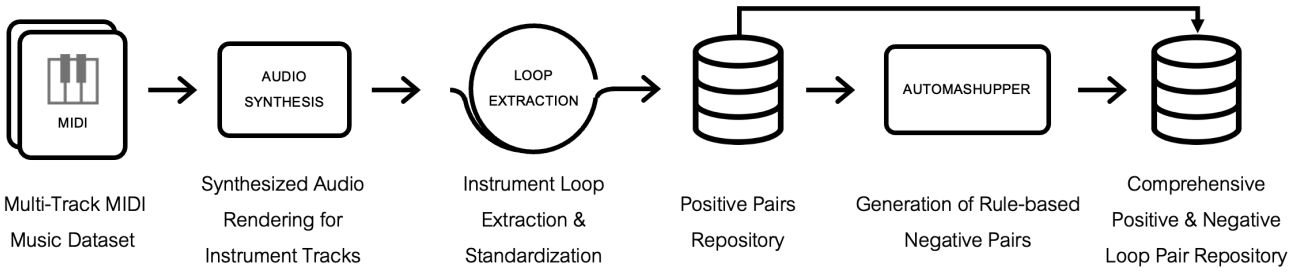

Figure 1. Our dataset generation pipeline takes multi-track MIDI music as input and generates a labeled loop dataset.

et al. do not extract loops and instead use complete stems to train a model on mash-ups involving combinations of vocal and backing track stems; whereas our research delves into the compatibility of instrumental music loops. Therefore, we use Chen et al. as a baseline for our work.

Broadly, despite their utility, these methods do not capture the complex interplay of musical elements, underscoring the necessity for more advanced methods. With the availability of a larger dataset, we introduce and examine the application of Transformer-based architectures for addressing the vertical loop compatibility problem.

3. DATA GENERATION PIPELINE

We introduce a novel self-supervised method to create a large, high-quality, labeled dataset to provide instrument-level granularity. Our method also ensures identical basic attributes — such as tempo, key, and phase — amongst incompatible pairs to compel models to learn compatibility and not focus on such superficial differences.

3.1 Generating Positive Examples

Synthesized datasets have shown promise in enhancing model performance across various music information retrieval tasks, including transcription, understanding compositional semantics, sound synthesis, and instrument recognition [13]. In the absence of an instrument-level labeled dataset, we modify the data collection pipeline proposed by Chen et al. to instead create loops from synthesized data [4]. Similar to Flakh, we generate our dataset by taking songs from the Lakh dataset, rendering MIDI files using sample-based synthesizer and then extracting loops [13, 14]. For this task, we used FluidSynth [2] .

The Lakh MIDI dataset, with over 175,000 unique MIDI files, provides detailed musical score data for various instruments that can be synthesized due to distinct track segmentation. We chose files with significant parts for piano, bass, guitar, and drums. A total of 20,371 files are identified, of which 15,000 were taken at random and rendered [13]. Each MIDI file is split into individual instrument tracks, matched with appropriate patches based on program numbers, and rendered into audio. As observed by Chen et al., when adjusted for tempo and key, drum and bass loops tend to be universally compatible. Therefore, all drum and bass MIDI tracks were removed from synthesis and subsequent creation of positive and negative pairs [4]. The collected set of 15,000 songs spans 13 genres and 47 instruments. We then use the same method as Chen et al. to extract loops from each rendered audio [4]. Of the 15,000 songs, 12,193 songs have at least one valid loop pair. Specifically, of these 12,193 songs, we obtained 126,746 loops and 90,376 valid positive pairs of loops. This provides our training models with more granular information about *each* instrument loop while learning what constitutes compatibility. Files were separated into training (72,301 loop pairs), validation (9,037 loop pairs), and testing (9,038 loop pairs) — leaving us with a total of 251 hours of positive examples, with roughly equal representation of instruments and genres in each set. To ensure consistency, we standardize the duration of each loop to 10 seconds by either repeating or trimming the loops.

3.2 Generating Negative Examples

Generating pairs of negative loops is a difficult task. One naive approach could be to randomly select combinations from our loop set. However, this does not guarantee incompatibility. Unlike what's been proposed in similar works, we argue that simply altering the basic loop characteristics to guarantee the generation of negative examples risks misleading the model to learn to distinguish these superficial differences rather than understanding true musical incompatibility. Instead, to reflect real-life music production choices, we train models to determine incompatibility between loops sharing identical tempo, key, and phase. We find that while AutoMashUpper demonstrates modest success in identifying compatible loops, its strength lies in identifying incompatible loops within the same tempo, key, and phase, thus furnishing reliable negative labels for compatibility modeling. Inversely applying the original method focuses on *least* compatible pairs. Harmonic incompatibility finds significant chord progression clashes, rhythmic incompatibility leads to off-sync combinations, and spectral imbalance points to lopsided energy distributions, cultivating disturbances and noise.

We adopt AMU to a subset of loops by drawing, without replacement, 1,500 positive pairs (3,000 loops) from varied genres and instruments. For each loop in this collection, we calculate its incompatibility against every other loop by adjusting the target loop's keys and tempos to match the query loop and then calculating weighted sums of harmonic, rhythmic, and spectral compatibility between the source and target. For this task, we use a Python im-

[2] Available at: http://www.fluidsynth.org/

plementation [3] of the Krumhansl-Schmuckler key-finding algorithm [15], Rubber Band [4] for sound stretching and pitch-shifting, and weights proposed by Bernardo — 0.4 for both harmonic and rhythmic, and 0.2 for spectral compatibility — to derive an overall compatibility score between loop pairs [16]. After obtaining all scores, the 35 least compatible loops are paired with each loop in the set. We exclude any duplicate pairs, culminating in 95,281 unique negative pairs. More than 1,000 negative pairs are tested at random by the research team to confirm incompatibility. The final pairs are then partitioned into training (76,225 loop pairs), validation (9,528 loop pairs), and testing segments (9,528 loop pairs), leaving us with a total of 264 hours of negative examples. The negative set can be significantly expanded by drawing more pairs at the start.

Data Type	# Loops	# Loop Pairs	# Hours
Training	101,397	148,526	412
Validation	12,674	18,565	51.5
Test	12,675	18,566	51.5
In Total	126,746	185,657	515

Table 1. Overview of data, including positive and negative examples, across training, validation, and test subsets.

4. MODEL ARCHITECTURE

Recent advancements in self-attention networks, particularly the Transformer architecture, provide a new perspective for solving the vertical loop compatibility challenge [17]. In this study, a large labeled dataset allows us to introduce and examine the application of Transformer-based architectures. Specifically, we use the same model architecture as MusicTaggingTransformer (MTT), proposed by Won et al., for its robustness on other MIR tasks [18]. We refer to this adapted Transformer architecture as the Deep Recombinant Transformer (DRT). Initial pre-processing employs MelSpectrogram transformation and AmplitudeToDB conversion of the input, which comprises the summed audio signals of two candidate loops. This is followed by Res2DMaxPoolModule for downsampling, with subsequent convolutional layers and max-pooling operations for detailed feature extraction. The core Transformer architecture is equipped with 256-dimensional attention vectors across four layers and eight heads, PreNorm, Residual structures, and GELU-activated Feed Forward networks for processing. A unique class ([CLS]) token, alongside positional embedding, is added to the feature set for sequence analysis. The output from the Transformer is directed through a sigmoid function, mapping the high-dimensional feature vectors to a binary outcome space, and delineating the likelihood of each audio sample belonging to a specific category. Then, we compute the binary cross entropy loss (BCELoss) to update the parameters of the whole model. Model's output is between 0 and 1, with values closer to 1 indicating a higher probability that the pair of loops are compatible, and closer to 0 when they are not. Therefore, we can use its output

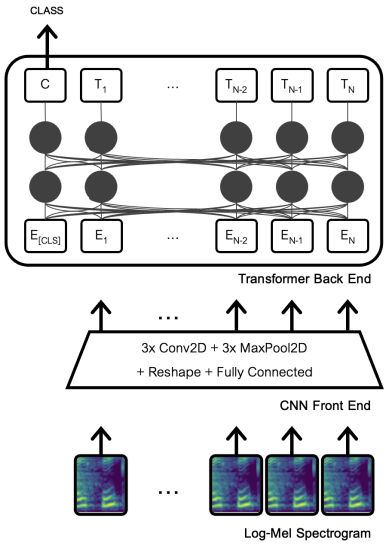

Figure 2. Architecture of Deep Recombinant Transformer.

to estimate the compatibility of any two loops. Dropout (0.1) and batch normalization strategies are implemented to mitigate over-fitting and ensure robust model generalization. This integration of convolutional and Transformer elements captures both local and global audio features for deep and context-aware analysis. To investigate the adaptability and performance of the Transformer architecture for this task, our study explores two distinct configurations: one variant employs two-encoder layers, while the other utilizes four-encoder layers. This enables us to evaluate the impact of architectural depth on model performance.

5. EXPERIMENT SETUP AND EVALUATION

We evaluate the performance of Transformer architectures in identifying compatible loops against the state-of-the-art. Following this, we focus on understanding the impact of our new dataset on model performance. Finally, we conduct a subjective assessment.

5.1 Effect of Using a Transformer

First, we compare the performance of Transformers against CNN-based architectures. We train and evaluate two configurations for the Transformer architecture — two and four encoder layers. For comparison, we explore the performance of the original CNN-based NLC [5] model proposed by Chen et al. Initially, we adhere to the original NLC specification, applying it to two-second audio segments extracted from our 10-second dataset. However, we also train a modified NLC on a 10-second long input for a fair comparison. Moreover, acknowledging that extended audio contexts may require a deeper CNN architecture, we also train Short-chunk CNN Res [19], a deeper CNN architecture, due to its strong performance on MIR-related classification tasks. We perform hyperparameter optimization for each architecture using unseen validation sets.

[3] https://pypi.org/project/pymusickit/
[4] https://breakfastquay.com/rubberband/
[5] https://github.com/mir-aidj/neural-loop-combiner

The first type of evaluation entails a classification task. It assesses a model's ability to distinguish compatible loops from incompatible ones. We report accuracy and F1 scores for each model. Table 2 summarizes these results.

Model	**Accuracy ↑**	**F1 Score ↑**
NLC (2 seconds)	62.25	68.76
NLC (10 seconds)	60.50	70.76
Short Chunk CNN Res	70.50	77.13
DRT (2 Attn Layers)	78.60	82.02
DRT (4 Attn Layers)	**80.90**	**83.66**

Table 2. Comparative performance on loop compatibility classification task for models trained using our dataset.

Model	**Avg. ↓ Rank**	**Top ↑ 10**	**Top ↑ 30**	**Top ↑ 50**
NLC (2 seconds)	43.4	0.25	0.44	0.56
NLC (10 seconds)	51.2	0.13	0.25	0.52
Short Chunk CNN Res	38.3	0.07	0.46	0.77
DRT (2 Attn Layers)	25.7	0.15	0.69	1.00
DRT (4 Attn Layers)	**16.2**	**0.44**	**0.75**	**1.00**

Table 3. Comparative performance on loop ranking task for models trained using our dataset, using average rank and accuracy in the top-k positions across 100 queries.

Another performance evaluation reported in the research involves ranking candidate loops by compatibility with a particular query loop [4, 5]. This assessment is especially important for a model's practical use, which seeks to find loops that match a specific query from a large collection of loops. Using AMU, and the unseen test set, we create a collection of candidate loops for each query loop, ensuring that precisely one of these candidates pairs positively with the query. The model's performance is measured by where the "target loop" ranks in the list, with a higher position indicating better performance. Following the benchmark set by Chen et al., we also assess the compatibility of exactly 100 candidate loops balanced across genres and instruments. Each model is evaluated for accuracy within the top 10, 30, and 50 positions, as well as the mean rank. Table 3 shows these aggregated averages.

The results indicate that the four attention layer MusicTaggingTransformer demonstrates superior performance across loop compatibility classification and ranking tasks. We also observe that models, though not explicitly trained for it, perform well in identifying compatible drum and bass loops, confirming these are relatively trivial to learn.

5.2 Effect of Using Our Dataset

Generating negative examples by altering loop characteristics can mislead models toward learning superficial differences instead of true musical incompatibility. To objectively evaluate this, we create a control dataset using the negative sampling methodology proposed by Chen et al. In this control dataset, the positive pairs remain the same, while for negative pairs we reverse, randomly shift beats, or re-arrange beats of one of the loops. Although reversing performed best in the original study, the performance differences across the three strategies were small. To account for potential non-transferability to our dataset, we include an equal representation of all three methods in our control set. We retrain the three best-performing architectures from Table 2, and evaluate them, on this control dataset. The classification results are summarized in Table 4 while the retrieval results are summarized in Table 5.

Model	**Accuracy ↑**	**F1 Score ↑**
NLC (2 seconds)	66.4	71.4
Short Chunk CNN Res	**88.9**	**89.3**
DRT (4 Attn Layers)	88.2	88.7

Table 4. Comparative performance on loop compatibility classification task for models trained using control dataset.

Model	**Avg. ↓ Rank**	**Top ↑ 10**	**Top ↑ 30**	**Top ↑ 50**
NLC (2 seconds)	13.25	0.57	0.75	1.00
Short Chunk CNN Res	1.0	1.00	1.00	1.00
DRT (4 Attn Layers)	1.0	1.00	1.00	1.00

Table 5. Comparative performance on loop ranking task for models trained using control dataset, using average rank and accuracy in the top-k positions across 100 queries

While all models show improved performance across both tasks, we perform another set of evaluations to determine if these on-paper performance gains are transferable to real-life production scenarios. Here, we evaluate these models, trained on the control set, against the test set generated by our proposed method — where pairs sharing the same tempo, key, and phases are analyzed for compatibility. These results are presented in Tables 6 and 7.

Model	**Accuracy ↑**	**F1 Score ↑**
NLC (2 seconds)	50.95	62.49
Short Chunk CNN Res	53.30	67.52
DRT (4 Attn Layers)	**54.15**	**67.93**

Table 6. Classification performance of models trained on control set (Table 4), but evaluated on our original test (Table 1) containing loops pairs with identical tempo and keys.

Model	**Avg. ↓ Rank**	**Top ↑ 10**	**Top ↑ 30**	**Top ↑ 50**
NLC (2 seconds)	50.7	0.13	0.31	0.50
Short Chunk CNN Res	39.4	0.00	0.30	**0.85**
DRT (4 Attn Layers)	**35.8**	**0.14**	**0.40**	0.67

Table 7. Ranking performance on our original test set using average rank and accuracy in the top-k positions.

While the models trained on the controlled dataset have better performance (Tables 4 and 5) than models trained on our dataset (Tables 2 and 3), they do not generalize for negative samples that are more in line with real-world production choices (Tables 6 and 7). This is because real-life comparison involves two loops that are identical in tempo,

key, and phase, without being reversed or subjected to random beat shifts. Since these models have not encountered such incompatible samples during training, their performance tends to degrade in the production setting.

5.3 Subjective Assessment

Study Methodology: We perform a subjective analysis using Apple loops from GarageBand to evaluate the effectiveness of our proposed method and demonstrate its applicability to high-quality production loops. Based on superior objective performance, three models are selected for this user study: NLC (2 seconds), Short Chunk CNN Res, and DRT (4 Attn Layers). Two variants of each model were included, the first trained on our proposed dataset, and the second trained on the control dataset. For this subjective assessment, we employed a methodology similar to that used by Zhao et al. [20]. We paired one query loop against 99 candidates, within the query loop's genre, and formulated audio test clips by combining the query loop with the highest-ranked match. Each set contained eight query-target pairs: top matches proposed by each variant of the three models, a human musician-generated pair, and a randomly selected target loop to serve as a control group. Each audio sample was of equal length (10 seconds). A total of 6 such sets were created. The subjects were asked to rate each sample on a 5-point Likert scale according to the following criteria:

- **Seamlessness:** Naturalness of the loop combination.
- **Creativity:** Originality and inventive quality.

The study engaged a total of 37 participants. To qualify, participants were required to have a baseline engagement with music, defined as listening to at least five hours of music per week, to ensure sufficient exposure to music to provide informed feedback. Each survey participant listened to exactly three of the sets chosen at random (24 audio combinations, or 240 seconds of audio). To ensure diverse and representative survey respondents, we employed demographic filtering to include different ages, genders, and cultural backgrounds. The sequence of the presentation was randomized to eliminate any potential bias, and the origins of the pairs were not disclosed to participants.

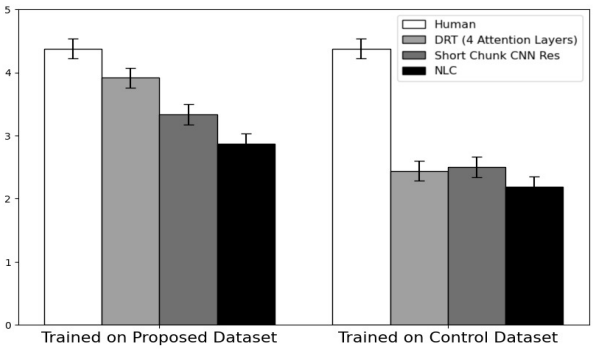

Figure 3. Subjective evaluation results for composition seamlessness computed using within-subject ANOVA.

Results and Analysis: Figures 3 and 4 display our findings from the subjective evaluation. The y-axis represents the mean scores and the error bars denote the Standard Deviation calculated through a within-subject ANOVA [21]. Our model demonstrated superior performance over the control, achieving statistical significance ($p<0.05$) for both measures. The proposed DRT architecture, trained on our dataset, surpassed other models by a significant performance difference ($p < 0.001$). The participant responses in the survey demonstrated high reliability, as evidenced by a Cronbach's α of 0.812 [22]. Overall, the scores for our approach were on par with human music compositions.

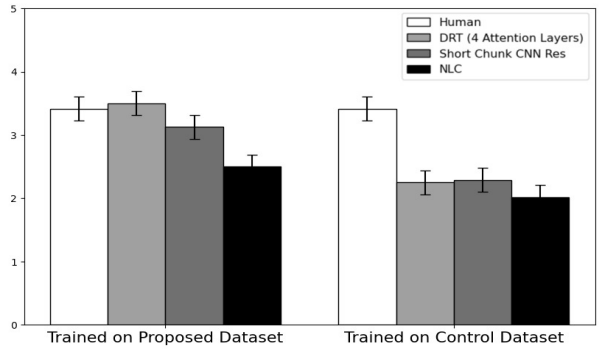

Figure 4. Subjective evaluation results for composition creativity computed using within-subject ANOVA.

6. CONCLUSION

We explored the vertical loop compatibility problem in music production. One major limitation to applying deep neural networks in this domain has been the absence of labeled datasets. We presented a novel self-supervised method for generating a large, high-quality, labeled dataset from a multi-track MIDI dataset, containing separate instrument tracks, and synthesizing them into audio to extract loops. This provides our training models with more granular information about each instrument across different genres and provides negative pairs with matching tempo, key, and phase to force models to learn true musical compatibility. A large dataset allows us to introduce and examine Transformer-based architectures. Our architecture employs a larger context window of ten seconds allowing a holistic input representation and consequently better compatibility prediction. Experimental results show that our method outperforms the previous state-of-the-art, achieving an 18.6% higher accuracy across multiple genres. Subjective assessments rate our model higher in seamlessly and creatively combining music loops.

Nevertheless, implementing Transformer architectures demands significant computational resources. Also, while AMU performs well in identifying incompatible pairs, it does not guarantee incompatibility and may contain leakage. Finally, synthesized audio from MIDI may not fully capture the richness of professionally recorded music. This could limit the model's learning scope, especially regarding timbral and expressive nuances which otherwise may be important to learn. Future work may involve experimenting with more efficient architectures, collecting human-labeled datasets, and synthesizing MIDI using professional virtual instruments for better dataset quality.

7. REFERENCES

[1] M. E. Davies, P. Hamel, K. Yoshii, and M. Goto, "Automashupper: Automatic creation of multi-song music mashups," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1726–1737, August 2014.

[2] G. Sorcinelli, "From garageband loop to grammy award: A look back at rihanna's "umbrella"," *Micro-Chop*, Oct 2016.

[3] M. Goto, "Grand challenges in music information research," in *Multimodal Music Processing*, ser. Dagstuhl Follow-Ups, M. Müller, M. Goto, and M. Schedl, Eds. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012, vol. 3, pp. 217–226. [Online]. Available: https://drops-dev.dagstuhl.de/entities/document/10.4230/DFU.Vol3.11041.217

[4] B.-Y. Chen, J. B. L. Smith, and Y.-H. Yang, "Neural loop combiner: Neural network models for assessing the compatibility of loops," *arXiv preprint arXiv:2008.02011*, 2020.

[5] J. Huang, J. C. Wang, J. B. Smith, X. Song, and Y. Wang, "Modeling the compatibility of stem tracks to generate music mashups," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, May 2021, pp. 187–195.

[6] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "Automashupper: An automatic multi-song mashup system," in *International Society for Music Information Retrieval Conference*, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:28117

[7] C.-L. Lee, Y.-T. Lin, Z.-R. Yao, F.-Y. Lee, and J.-L. Wu, "Automatic mashup creation by considering both vertical and horizontal mashabilities," in *International Society for Music Information Retrieval Conference*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:17802326

[8] K. Tsuzuki, T. Nakano, M. Goto, T. Yamada, and S. Makino, "Unisoner: An interactive interface for derivative chorus creation from various singing voices on the web," in *ICMC*, 2014.

[9] G. Bernardes, M. Davies, and C. Guedes, "A hierarchical harmonic mixing method," in *Music Technology with Swing*, ser. Lecture Notes in Computer Science, M. Aramaki, M. Davies, R. Kronland-Martinet, and S. Ystad, Eds., vol. 11265. Cham: Springer, Cham, 2018, cMMR 2017. [Online]. Available: https://doi.org/10.1007/978-3-030-01692-0_11

[10] G. Bernardes, D. Cocharro, M. Caetano, C. Guedes, and M. Davies, "A multi-level tonal interval space for modelling pitch relatedness and musical consonance," *Journal of New Music Research*, vol. 45, pp. 1–14, May 2016.

[11] C. Maçãs, A. Rodrigues, G. Bernardes, and P. Machado, "Mixmash: A visualisation system for musical mashup creation," in *2018 22nd International Conference Information Visualisation (IV)*, 2018, pp. 471–477.

[12] J. B. L. Smith, Y. Kawasaki, and M. Goto, "Unmixer: An interface for extracting and remixing loops," in *ISMIR*, 2019, pp. 824–831.

[13] E. Manilow, G. Wichern, P. Seetharaman, and J. L. Roux, "Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 45–49.

[14] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, PhD Thesis, 2016.

[15] C. L. Krumhansl, "Cognitive foundations of musical pitch," 2001.

[16] G. Bernardo and G. Bernardes, "Leveraging compatibility and diversity in computer-aided music mashup creation," *Personal and Ubiquitous Computing*, vol. 27, no. 5, pp. 1793–1809, 2023.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[18] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," *arXiv preprint arXiv:2111.13457*, 2021.

[19] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," *arXiv preprint arXiv:2006.00751*, 2020.

[20] J. Zhao and G. Xia, "Accomontage: Accompaniment arrangement via phrase selection and style transfer," 2021.

[21] H. Scheffe, *The Analysis of Variance*. John Wiley Sons, 1999, vol. 72.

[22] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, Sep 1951.