International Journal for Housing Science and Its Applications
Publish September 16, 2025. Volume 47, Issue 1 Pages 1007-1017

https://doi.org/10.70517/ijhsa47187

M2S-GAN: A Cross-Track Attention-Based Transformer Model for Multi-Track Music Generation with Music Theory Constraints

Chen Zihao[1] and Pan Jie[1,*]
[1] Communication University of China, Nanjing, Jiangsu, 210000, China
Corresponding authors: (e-mail: Panjie@cucn.edu.cn).

Abstract As multi-track music creation jobs become more complicated, it has become more difficult to capture the interdependencies between tracks and produce high-quality, music theory-compliant music. In order to address the issues of co-generation and music theory restrictions in multi-track music production, we provide in this study a novel framework, M2S-GAN, based on Generative Adversarial Networks (GAN) and Transformer. In order to achieve collaborative multi-track generation, we present the Cross-Track Attention method, which uses cross-track self-attentive learning to capture the intricate relationships and long-term dependencies between various tracks. To guarantee that the produced music satisfies the standards of conventional music theory in terms of harmony, melody, and rhythm, the model also integrates music theory rules to direct the production process in the form of mathematical models. M2S-GAN improves the diversity and caliber of the generated music in addition to increasing generation stability through the skillful architecture of several generating and discriminative networks. According to experimental results, the suggested model performs better than current approaches in terms of the generated multi-track music's quality, stability, and musical rationality. It can also continue to produce outstanding results across a variety of datasets and assessment criteria. Our work offers a fresh perspective on multi-track music generating and robust support for automated music generation and creation.

Index Terms Multi-track music generation, Generative Adversarial Networks (GANs), Transformer, Cross-Track Attention, Music Theory Constraints, Music Composition, Deep Learning, Generative Modeling

I. Introduction

Generative Adversarial Networks (GANs) and Transformer-based music production models have emerged as a research hotspot in the field of music composition and autonomous composing due to the ongoing advancements in deep learning technology [1]. These models mimic the structure and rules of music to produce excellent musical compositions. Research on multi-track music production has progressively received interest, particularly in recent years. This is especially true when combining music theory rules with multi-track co-creation, which presents both potential and obstacles [2], [3]. However, producing degrees of difference between real samples, matching music theory requirements, and capturing complicated musical structures remain major challenges for current generative models [4].

Complex time-series data, including multi-dimensional correlation information like melody, harmony, and rhythm, are used in the music production task [5]. Although conventional rule-based music generation techniques are capable of producing music of a particular caliber, their fixation on a predetermined set of rules prevents them from being flexible and innovative, making it challenging to overcome the rigid framework [6]. By automatically learning the implicit laws in the data, deep learning-based generation techniques like Transformers, Long Short-Term Memory Networks (LSTM), and Recurrent Neural Networks (RNN) can produce more varied music that satisfies the requirements of creative production [7], [8]. A significant problem in contemporary research is how to efficiently manufacture multi-track music, particularly to preserve a harmonious and cooperative link between tracks, as the task of generating a single track has increasingly fallen short of the demands of complex musical works [9].

The game mechanism between generators and discriminators allows GAN-based generative models, which have been widely utilized in the field of music generation in recent years, to effectively increase the quality of generated music. Existing research, however, mostly concentrates on learning a single track for the task of creating multi-track music, frequently neglecting the intricate relationships and synergistic production between tracks. Higher restrictions are placed on the model's design because music generation must take into account not only each track's independence but also make sure that the various songs blend in with one another [10].

Particularly for instrumental recordings like piano, guitar, and bass, whose melodic, rhythmic, and harmonic information is entwined with one another, the interactions between various songs are complex. Therefore, a significant difficulty in multi-track music production is how to guarantee that each track is generated separately while yet being able to capture the connections between tracks [11], [12]. Although deep learning models are capable of producing music of excellent quality, the output of these algorithms frequently lacks musical reason. Note arrangements, chord progressions, and rhythmic assignments are a few examples of elements that might not follow conventional music theory guidelines. As a result, it is now vital to figure out how to apply knowledge of music theory to the generation process so that the resulting music adheres to specific musical standards while maintaining the flexibility of artistic expression. Long temporal dependencies are common in musical compositions, particularly in multi-track production, when note selection must consider the dynamic changes of other tracks in addition to the present track's state [13]. Improving the quality of the generation depends on how well these long-standing dependencies are captured, particularly across tracks.

A challenge for deep learning models, particularly GANs, is training instability. Due to the interactions between the many tracks, the game process between the discriminator and generator may make it difficult for the model to converge, particularly when creating multi-track music [14]. Thus, one of the main challenges in model optimization is how to balance the training of discriminators and generators while designing the loss function efficiently.

Several Transformer and GAN-based methods are available for creating multi-track music. For instance, the Cross-Track Attention mechanism is used to improve the synergistic generation between tracks, and the multi-head Self-Attention mechanism (Self-Attention) is used to capture the long-term interdependence between individual notes in music [15]. However, the majority of previous research has ignored the intricate relationships between tracks in favor of concentrating on straightforward linkages. Furthermore, while some methods have tried to include music theory principles into the generation process, the majority of researchers are still figuring out how to properly integrate music theory rules with generative modeling, and this field of study is still in its infancy.

Complex musical structures, long-term relationships, and cross-track learning are still difficult for current multi-track music creation algorithms to handle. For instance, the majority of methods fail to efficiently use inter-track dependencies to direct creation when it comes to inter-track interactions. Furthermore, even while certain techniques may identify intricate patterns inside tracks, they still struggle with how to synchronize generation across tracks in a multi-track setting.

The Cross-Track Attention mechanism, which we introduce in this study, is the main innovation of the multi-track music generation adversarial network (M2S-GAN) based on the CT-Transformer architecture. This mechanism effectively captures the intricate relationships between tracks and co-generation in a multi-track environment. Furthermore, this work incorporates the rules of music theory into a mathematical model of a generative network, rewards and penalizes the generative process based on the relative importance of the knowledge of music theory, and directs the model to produce musical compositions that adhere to the norms of music theory [16].

In particular, this paper's primary contributions consist of:

(1) The CT-Transformer is suggested, which enables the collaborative creation of multi-track music and efficiently learns the relationships between tracks via the cross-track attention method.

(2) Making sure that the created works adhere to music theory, integrating music theory rules into the model, and using a reward system to direct the harmony of the music.

(3) The adversarial training approach is used to successfully overcome the generation process's instability and enhance the model's stability and generation quality through the carefully planned generation and identification network.

II. Transformer-based adversarial generative networks

II. A.Data presentation

Eight features from the MIDI files must be retrieved and encoded into an event sequence depending on the features in order to adapt the MIDI data to the model's generating task. The MIDI files for the experiments in this paper are organized and separated using hexadecimal notes based on the start time of each event type. The bar indicates the bar. Position indicates where each type of event is located. The chord progression used in this work is indicated by the letter "chord." TempoClass indicates the type of tempo (slow, moderate, or fast). Tempo Value indicates the tempo type's quantized value. Duration indicates how long each note lasts. The pitch onset (pitch quantization range "0~127") is indicated by Note On. Velocity indicates how loud a note is perceived to be. Duration indicates how long each note lasts.

II. B.Overall structure

As shown in Fig. 1, all of the music data used in this work is in MIDI format, and each file only contains three tracks: bass, guitar, and piano. Three generators (G_p 、G_g 、G_b) first encode the three tracks into a time sequence, then they learn the internal information of a single track sequence and generate the state y_t of the subsequent instant.After learning the guitar and bass sequences, the piano sequences are spliced to create the new piano sequences; learning the guitar and bass sequences is identical to learning the piano sequences; final steps include learning the real sample sequences as the piano sequences and splicing the real sample sequences to the piano sequences. Finally, the discriminator D_φ discriminates between the created sample sequence and the real sample sequence after the guitar and bass sequences are learned in the same manner as the piano sequence.

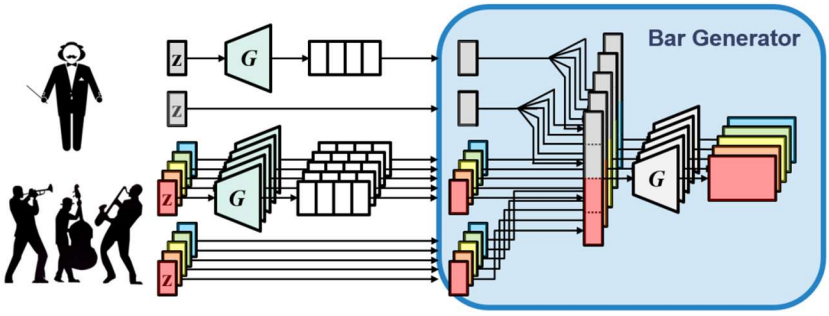


Figure 1: Adversarial network architecture for creating multi-track music

II. C.Generation Network

The Transformer's decoding component is used in the mono-track section as part of the mono-track sequence generating network, as shown in Fig. 2, where an embedding matrix transferred into an Embedding represents a sequence of input letters. This embedding sequence is then combined with a positional embedding, which makes sure the character can only learn the knowledge before to the kth moment by masking out the information after the kth moment using $N_g(N_g=5)$ self-attention modules.

To determine the output character distribution, a softmax layer first maps the output of the last self-attention module to the lexical space before activating it. The mono-track generator is taught to minimize the cross-entropy loss between the predicted characters and the inputs, which are genuine characters, during the pre-training stage. The generator creates characters in an autoregressive fashion throughout the generation phase. The generation can either be provided a starting sequence or start from the beginning.

The module's basic component is the Cross-Track attention mechanism, and in the multi-track section, it enhances the Transformer-based self-attention mechanism, or CT-Transformer. The numbers $\mathbf{X}_{\mathrm{p}} \in \mathbf{R}^{T_{\mathrm{p}} \times d_{\mathrm{p}}}, \mathbf{X}_{\mathrm{g}} \in \mathbf{R}^{T_{\mathrm{g}} \times d_{\mathrm{g}}}$ stand for the piano and guitar sequences, respectively, as the learning objects.The sequence length is indicated by T(-), and the feature dimension by d(-).

First, the key-value pairs are $W_{Q_{\mathrm{p}}} \in \mathbf{R}^{d_{\mathrm{p}} \times d_Q}, W_{K_{\mathrm{g}}} \in \mathbf{R}^{d_{\mathrm{g}} \times d_\kappa}, \mathbf{W}_{V_{\mathrm{g}}} \in \mathbf{R}^{d_{\mathrm{g}} \times d_{\mathrm{V}}}$, with $W_{Q_{\mathrm{p}}} \in \mathbf{R}^{d_{\mathrm{p}} \times d_Q}, W_{K_{\mathrm{g}}} \in \mathbf{R}^{d_{\mathrm{g}} \times d_\kappa}, \mathbf{W}_{V_{\mathrm{g}}} \in \mathbf{R}^{d_{\mathrm{g}} \times d_{\mathrm{V}}}$ denoting the weights of the piano and guitar, and the query is defined as $Q_p = X_p W_{Q_p}$. Equation (1) displays the state acquired at time I, which is $Z^{[i]}_{\mathrm{P}\to\mathrm{g}}$, as illustrated in Fig. 3.

$$Z^{[i]}_{\mathrm{P}\to\mathrm{g}} = \mathrm{f}^{[i]}_{\theta_{\mathrm{p}}\to\mathrm{g}}\left(\ln\left(\hat{Z}^{[i]}_{\mathrm{p}\to\mathrm{g}}\right)\right) + \ln\left(\hat{Z}^{[i]}_{\mathrm{P}\to\mathrm{g}}\right) \tag{1}$$

As shown in Eq. (2), the piano sequence $\hat{Z}^{[i]}_{\mathrm{p}\to\mathrm{g}}$ is used to learn the guitar sequence following i layers of the multi-head Cross-track attention output sequence. In Equation (3), CT represents the multi-head attentions value between two tracks, while in Equation (4), headh represents a single attentions value.

$$\hat{Z}^{[i]}_{\mathrm{p}\to\mathrm{g}} = CT^{[i]}_{\mathrm{p}\to\mathrm{g}}\left(\ln\left(Z^{[i-1]}_{\mathrm{p}\to\mathrm{g}}\right), \ln\left(Z^{[0]}_{\mathrm{p}}\right)\right) + \ln\left(Z^{[i-1]}_{\mathrm{p}\to\mathrm{g}}\right) \tag{2}$$

$$CT^{i}_{\mathrm{p}\to\mathrm{g}}\left(X_{\mathrm{p}}, X_{\mathrm{g}}\right) = \mathrm{Multihead}(h) = W\left[\mathrm{head}_1; \mathrm{head}_2; \cdots; \mathrm{head}_h\right] \tag{3}$$

$$\mathrm{head}_h = \mathrm{CTattention}\left(Q_{\mathrm{p}}, \mathbf{K}_{\mathrm{g}}, \mathbf{V}_{\mathrm{g}}\right) = \mathrm{softmax}\left(\frac{\mathbf{Q}_{\mathrm{p}}\mathbf{K}_{\mathrm{g}}^{\mathrm{T}}}{\sqrt{d_k}}\right)V_{\mathrm{g}} \tag{4}$$

After the output sequence is obtained, it is layer normalized to make it the same dimension as the input sequence. It is then fed into the feed-forward sublayer, which creates the output sequence following the module by residually concatenating the normalized output sequence $Z_{\mathrm{p}\to g}^{[i]}$'s i-th layer. Likewise, the bass track information is learned by the piano track as $Z_{\mathrm{p}\to b}^{[i]}$.

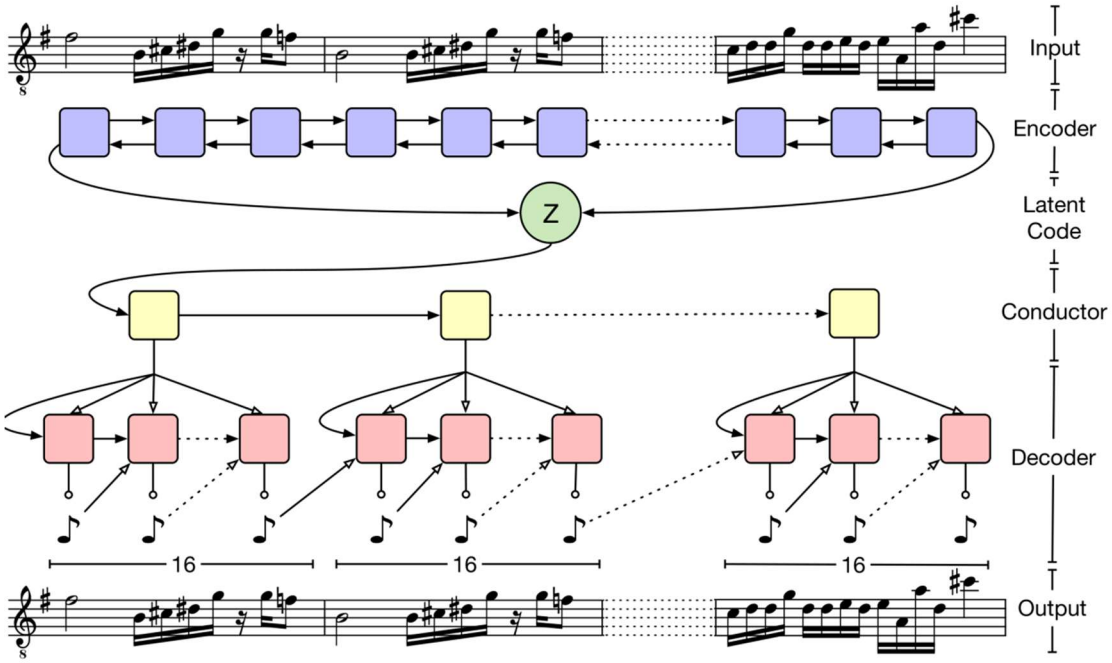


Figure 2: Single-track music creation structure

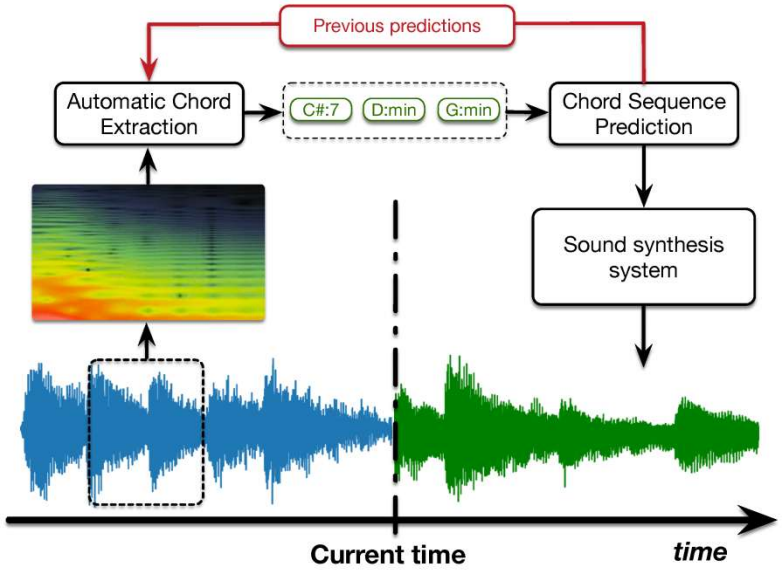


Figure 3: Guitar Sequence Module for Piano Sequence Learning

The piano sequence Z_{p} , this contains the guitar and bass sequence information found in equation (5),is finally obtained by splicing $Z_{\mathrm{p}\to g}^{[i]}$ and $Z_{\mathrm{p}\to b}^{[i]}$ together. The piano sequence Z_{p} is comparable to the bass sequence Z_b and the guitar sequence Z_g .

$$Z_{\mathrm{p}} = \mathrm{Concat}\left(\mathbf{Z}_{\mathrm{p}\to g}^{[i]}, \mathbf{Z}_{\mathrm{p}\to b}^{[i]}\right) \tag{5}$$

II. D.Authentication Networks

In this study, the goal value $\hat{y}_{t+1}$ at this moment is treated as the input note vector X_{t+1} at the following moment, i.e.,It is an environment for supervised learning. The original sample and the predicted value are used to update the model parameters. This is because the loss function allows the deep learning network to effectively learn the connection between feature information and allows the model network to accomplish the set task. Since the output layer in this article is the softmax layer—that is, the notes' probability distribution—the loss function is built using cross entropy [15].

When creating multi-track music, cross entropy training of the model can significantly optimize the parameters and improve the quality of musical compositions, as seen in Eq. (6).

$$D_{\varphi} = -\frac{1}{I}\sum_{t=1}^{I}\left[y_t \log \hat{y}_t + (1-y_t)\log(1-\hat{y}_t)\right] \tag{6}$$

This paper mathematically models the rules of music theory in order to produce music that complies with the knowledge of music theory [16]. The generative network is then fed varying reward and punishment values based on the significance of the music theory knowledge in the music. This allows the music generation to be guided by the rules of music theory. The notes in popular music tracks for the piano, guitar, and bass must be in A2-C5, E-C3, and E-1-E1, respectively.

Equation (7) illustrates this, with $y_{\min}$ and $y_{\max}$ being the lowest and highest notes predetermined based on the music's key; y_t is the t-moment pitch, and $R^{\mathrm{m}_1}(S_{1:t}, y_t)$ is the reward value of the t-moment state in the preceding t-moments.

$$R^{\mathrm{m}_1}(S_{1:t}, y_t) = \begin{cases} 0.1, & y_t \in [y_{\min}, y_{\max}] \\ -0.6, & y_t \notin [y_{\min}, y_{\max}] \end{cases} \tag{7}$$

According to Equation (8), where n is the most notes that can be heard and a_t is the number of notes at instant t.and $R^{\mathrm{m}_2}(a_t)$ displays the bonus value of the number of notes that fit the required range. One, six, and eight notes are the most that the bass, guitar, and piano can all play simultaneously.

$$R^{\mathrm{m}_2}(a_t) = \begin{cases} 0.2, & a_t,, n \\ -0.5, & a_t > n \end{cases} \tag{8}$$

As shown in Equation (9), where $S_{1:t}$ indicates the number of notes in the predicted sequence of the measure, y is the number of rests, and $R^{\mathrm{m}_3}(y)$ indicates the reward value for the number of rests in the measure that satisfies the condition, the number of rests per measure in the piano and guitar tracks should not be excessively high as this will impair the legibility of the entire track.

$$R^{\mathrm{m}_3}(y) = \begin{cases} 0.1, & y,, 0.4S_{1:t} \\ -0.3, & y > 0.4S_{1:t} \end{cases} \tag{9}$$

This paper uses triad chords, like F-G-Am-F. The rising music's strong beats are essentially odd-numbered beats for the chord intonation. According to equation (10), if C_1^t, C_2^t, C_3^t is the chord in-tone at time t, then y_t is the note that the generating network selected at that moment, and $R^{\mathrm{C}}(S_{1:t}, y_t)$ is the reward value that the rule provides.

$$R^{\mathrm{C}}(S_{1:t}, y_t) = \begin{cases} 0.7, & y_t \in \left(C_1^t, C_2^t, C_3^t |\ t\%2 = 1\right) \\ 1, & y_t \notin \left(C_1^t, C_2^t, C_3^t |\ t\%2 = 1\right) \end{cases} \tag{10}$$

Equation (11), where $R_{\mathrm{G}}(S_{1:t}, y_t)$ indicates the value of the reward that fulfills the set rule and α_i (i = {1, 2, 3, 4 }) indicates the weight occupied by the ith rule, shows how various weight ratios are given according on the importance of different musical rules.

$$R_{\mathrm{G}}(S_{1:t}, y_t) = \alpha_1 R^{\mathrm{m}_1}(S_{1:t}, y_t) + \alpha_2 R^{\mathrm{m}_2}(a_t) + \alpha_3 R^{\mathrm{m}_3}(y) + \alpha_4 R^{\mathrm{C}}(S_{1:t}, y_t) \tag{11}$$

Equation (12) illustrates how the discriminative network determines the model's goal function by giving the reward function and the cross-entropy loss function varying weights. The objective function's value is J_{G_θ}, This results from assigning weight values to the reward value and the loss function value β_1 and β_2, respectively.

$$J_{G_\theta} = \beta_1 R_{\mathrm{G}} + \beta_2 D_{\varphi} \tag{12}$$

III. Experimentation

III. A. Details of implementation

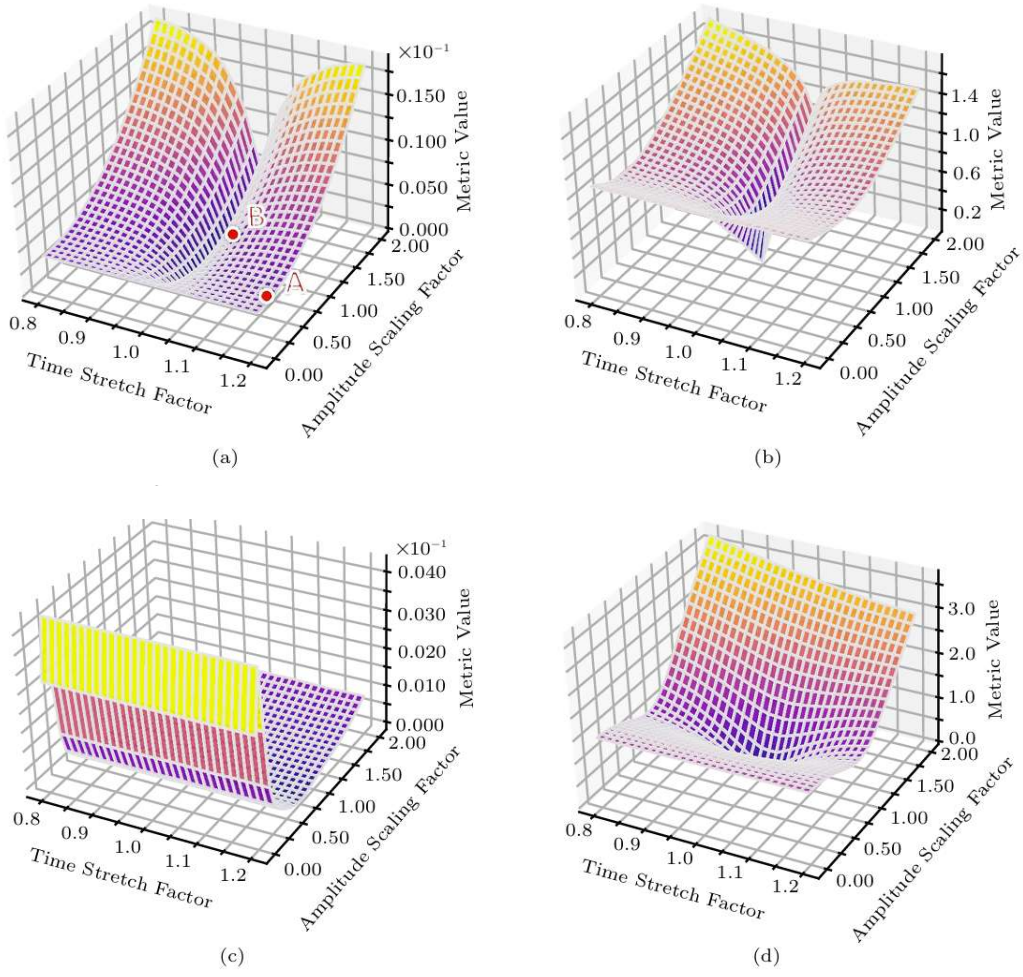
Every model uses a single NVIDIA 2080Ti GPU and is implemented in Pytorch 6.As explained in, the RMSprop optimizer trains our M2S-GAN with a learning rate of 0.0005.The Adam optimizer is used to train M2S-Net and other comparable models at a learning rate of 0.001.The comparison phase has a batch size of 10, and the generative learning phase has a batch size of 3. Our approach (M2SNet + M2S-GAN) requires roughly 48 hours to train in total. The training time needed for each phase is comparable.

Phase of contrast learning. 30-second music-action pairings were sampled for both positive and negative sample pairs. Each pair of samples had a duration of 10 seconds. Easy negative examples were utilized for pre-training in the first epoch of each model since we found that models containing tough or ultra-difficult negative data were hard to train from scratch.

Creating the phase of learning. The sample length was set at 60 seconds to help with long-term dependency learning. We empirically selected wGP = 10 for the loss function's hyperparameters. Following the WGAN GP, we train the generator once and the discriminator five times every step.

III. B. Assessment indicators

Few indicators are now available to measure the results because learning music-driven command movement creation is a relatively new endeavor. Traditionally, Initial Score (IS) or Frechette Initial Distance (FID) have been used as the basis for objective assessments of deep generative models in recent years. Interestingly, these measurements need feature encoders, which are typically acquired by pre-training classification; nevertheless, command actions do not include class labels. Therefore, as explained below, we suggest a number of new criteria in addition to the current ones to better evaluate the caliber of command actions. To further understand the features of these metrics, we present in Fig. 4 the variation of the metric values under spatiotemporal disturbances. To perform spatial perturbations, a magnitude scaling factor is applied to the motion.



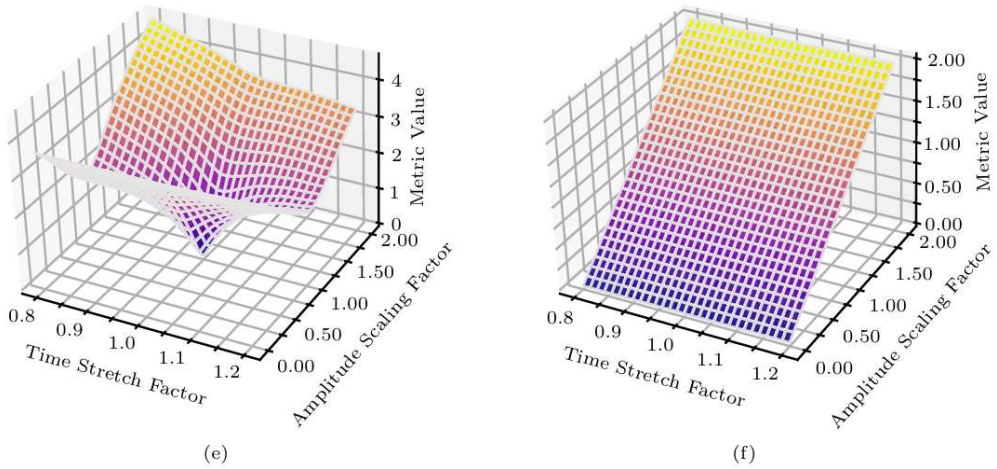


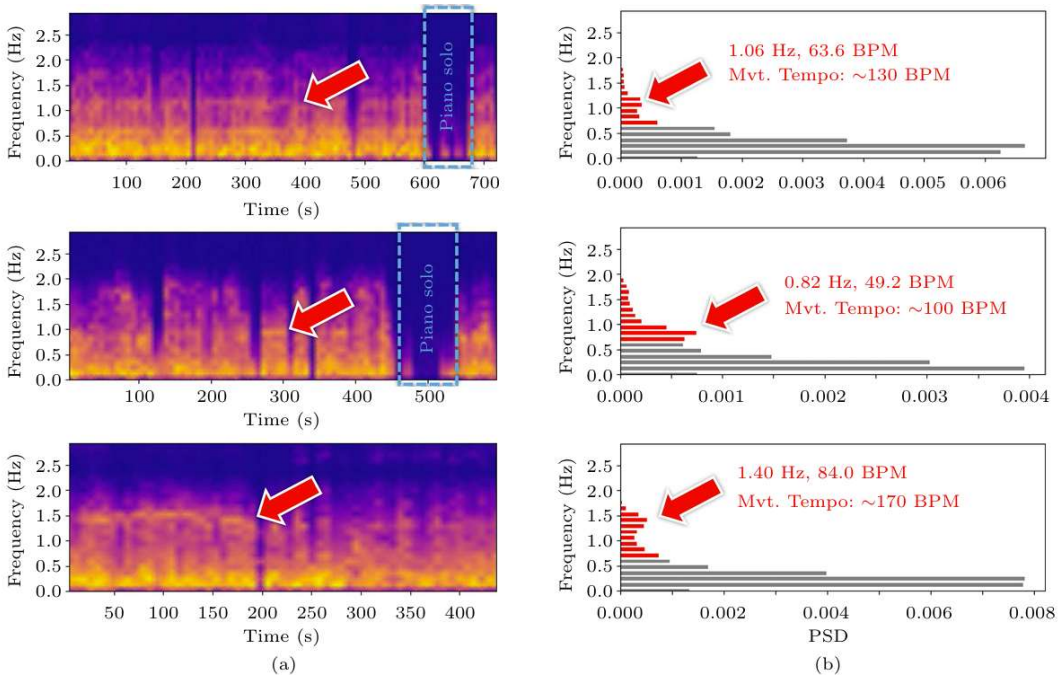
Figure 4: variation of metric values under spatio-temporal perturbations. (a) Mean square error. (b) Synchronization error. (c) Wasserstein distance. (d) Rhythm density error. (e) Intensity profile error. (f) Percentage standard deviation.

III. C. Percentage of standard deviation

The average motion intensity throughout time is measured by the standard deviation (SD). To determine whether the model is overly smoothed, we can compare the standard deviation of the simulated motion with the actual motion. The standard deviation percentage (SDP), as illustrated in Fig. 5, is linearly proportional to the motion's amplitude. SDP is defined in (9), where y is the average keypoint position over time, $\mathbf{y}_t$ is the ith motion frame, and T^y is the total number of frames. An SDP of roughly 100% is ideal for a conductive motion generating model, whereas 0% is ideal for static motion.

$$SDP(\mathbf{Y}) = \frac{SD(\hat{\mathbf{Y}})}{SD(\mathbf{Y})} \times 100\% \tag{13}$$

$$SD(\mathbf{Y}) = \sqrt{\frac{\sum_{t=1}^{T^y} \mathbf{y}_t - \bar{\mathbf{y}}_2^2}{T^y - 1}} \tag{14}$$



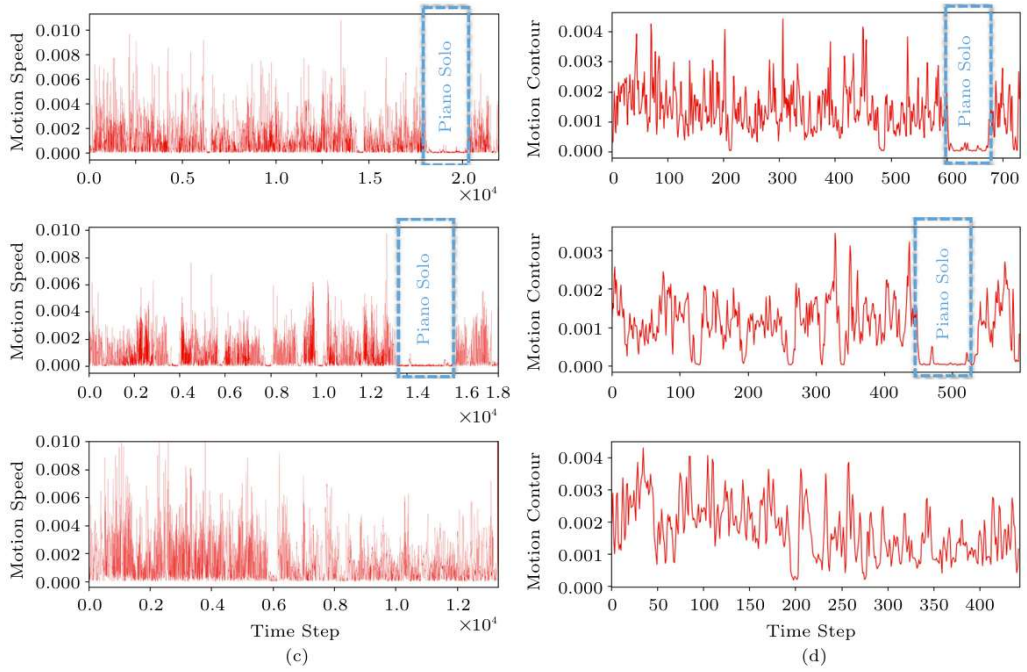


Figure 5: Mozart's Piano Concerto No. 17 in G major, K. 453, in three movements (Mvt.). Motion spectrum (a). Motion PSD (b). The red bars in the PSD represent the frequency intervals above the threshold (0.7 Hz). The RDE is the difference between these red bars. The red arrows highlight the rhythmic element of the guided action. The matching movement frequencies, which are extremely near to half of the musical rhythmic value, can be labeled. (c) Total velocity of motion. (d) Contours of intensity.The difference between these intensity contours is known as SCE. The solo piano parts in Mvt. 1 and Mvt. 2 are distinguished by the intensity contours and the total motion velocity.

III. D. Keeping counter loss and synchronization loss in check

First, we identify the ideal configurations for λ_{sync} and λ_{adv} using the validation set. After fixing λ_{adv} = 1, we compare performance on RDE and SCE using λ_{sync} = {0.001, 0.01, 0.02, 0.05, 0.1, 1}.

Fig. 6 displays the findings. We discover that the greatest results on both RDE and SCE are obtained with λ_{sync} = 0.05 and λ_{adv} = 1. The position is either to the left (λ_{adv} is larger and more realistic) or to the right (λ_{sync} is larger and more consistent), depending on the training synchronization loss and the change in W dis. In both cases, one loss term dominates the other, increasing both RDE and SCE.Consequently, we concluded that λ_{sync} = 0.05 and λ_{adv} = 1 were the ideal values. These parameters will be used in the upcoming trials.

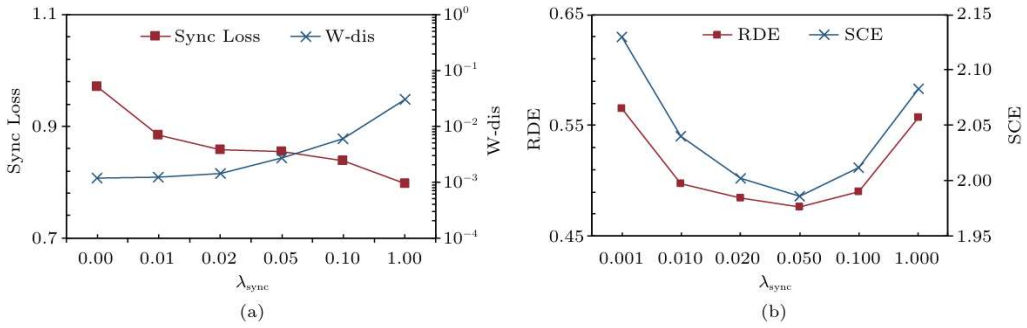


Figure 6: Training of SCE, RDE, W-dis, and synchronization loss with various hyperparameters. The value of λ_{adv} is set at 1. We discover that the best results on RDE and SCE are obtained with λ_{sync} = 0.05. (a) W-dis and synchronization loss. (b) SCE and RDE.

III. E. Performance comparisons

Our approach seeks to produce motions from audio input, whereas the majority of current music-driven conducted motion generation techniques, as covered in Section 2.1, require MIDI data as input. Furthermore, a fair comparison between our learning-based technique and these rule-based systems is hard to come up with. KHMM [12], initially suggested in 2003, is the only learning-based approach for music-driven conductive action creation. Nevertheless, this method necessitates the human selection of models appropriate for the music in question. Their techniques are also computationally inefficient, particularly when compared to our extensive ConductorMotion100 dataset. We selected three deep models that were first created for various audio motion translation tasks: music-driven dance generation, spoken gesture generation, and instrumental movement generation. This is because none of the current music-driven conductive motion generation techniques are appropriate for comparison. These models can be directly trained to produce conducted activities because none of them require any particular prior information [17], [18].

III. F. Impacts of various negative pairings

Difficult negative samples ought to be superior to simple and super-difficult negative samples for M2S learning. We empirically illustrate this here by training M2S-Net with various negative sample types and comparing the test accuracies. Table 1 displays the findings. The best outcomes are bolded. Remarkably, on all three test negative samples, the model that was trained using challenging negative data performs the best. This suggests that challenging negative instances can help M2S-Net acquire semantic correlation (simple negative samples) and temporal synchronization (ultra-difficult negative samples) [19], [20]. Additionally, compared to tough and very difficult negative samples, training with simple negative samples is extremely unstable, as Figure 7 illustrates. The most likely situation, according to our hypothesis, is that M2S-Net occasionally attempts to learn semantic correspondences and occasionally attempts to determine each sample's identity while working with simple negative samples. Simple negative samples can be a powerful baseline in many different multimodal self-supervised learning tasks. This experiment, however, demonstrates that basic negative samples are inappropriate for M2S learning. Furthermore, it was discovered in [15] that optimizing super-hard negative samples is challenging. The training process under ultra-hard negative samples is quite smooth, as illustrated in Fig. 7, even if the ultra-hard negative samples in our tests do not perform as well as the hard negative samples in M2S learning.

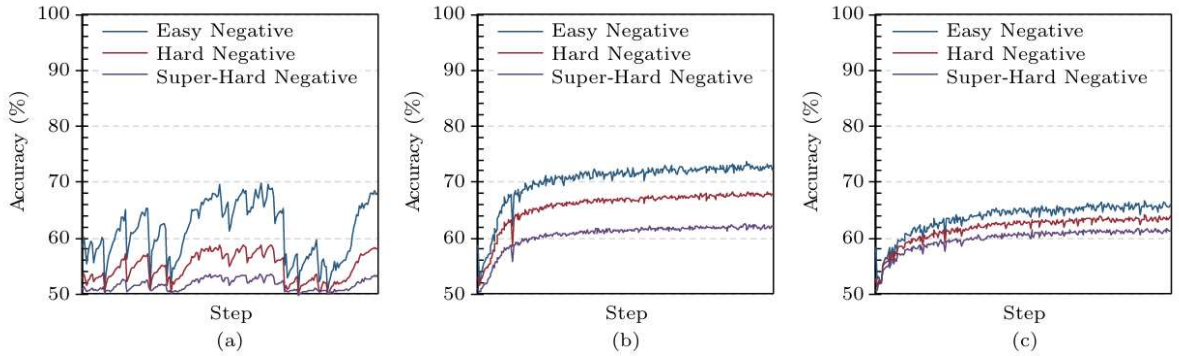


Figure 7: Variation of test accuracy during training in (a) easy, (b) difficult, and (c) ultra-difficult negative sampling conditions.

III. G. The effect of training set size

We then perform an additional experiment to demonstrate the necessity of removing MSE losses: we train the MSE model and our proposed M2S-GAN using training sets of different sizes. In particular, the ConductorMotion100 training set is 90 hours long. In this case, we train both models with {1,4,8,16,32,64,90} hours and evaluate how well they perform. Table 1 displays the findings. We can observe from the MSE model's training (blue dashed line) that, even with a relatively short training set, the model fits the data well. However, the MSE model's capacity to memorize data decreases with increasing training set size, resulting in an increase in the MSE. Simultaneously, the SDP drastically decreases, suggesting that the model is reducing its motion. On the other hand, our M2S-GAN's SDP (red line) stays constant at 100% and is not affected by the size of the training set.

Table 1: M2S learning outcomes for various negative sample types

Negative Pair	Training Accuracy (%)	Testing Accuracy		
		Easy (%)	Hard (%)	Super-Hard (%)
Easy	74.26	65.28	58.28	52.68
Hard	68.35	72.54	68.21	62.57
Super-Hard	61.78	65.32	62.45	60.54

IV. Conclusion

By merging Generative Adversarial Networks (GANs) with the enhanced Transformer architecture, the M2S-GAN model presented in this paper effectively addresses the co-generation and music theory restrictions issues in multi-track music generation. In order to produce more organic and melodic multi-track music, M2S-GAN successfully captures the intricate relationships and long-term dependencies between tracks by implementing the Cross-Track Attention technique. Furthermore, the application of music theory principles ensures that the music produced satisfies conventional standards for harmony, melody, and rhythm by imposing strict limitations on the process.

According to the experimental results, M2S-GAN can produce musical compositions that are of high quality, varied, and compliant with music theory. It also performs noticeably better than the current approaches in a number of evaluation measures. Specifically, M2S-GAN exhibits significant benefits in terms of musical rationality and generation stability. A new technical avenue for automated music creation is opened by the model design and experimental validation, which show the cross-track self-attention mechanism's enormous potential in multi-track music generation.

Future research could investigate the use of M2S-GAN in more musical genres and styles, improve the model's computational efficiency, or add more instruments and tracks to the generation process to increase the range of possible applications.

References

[1] WU, Guowei; LIU, Shipei; FAN, Xiaoya. The power of fragmentation: a hierarchical transformer model for structural segmentation in symbolic music generation. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2023, 31: 1409-1420.

[2] SHIH, Yi-Jen, et al. Theme transformer: Symbolic music generation with theme-conditioned transformer. IEEE Transactions on Multimedia, 2022, 25: 3495-3508.

[3] CHEN, Haonan, et al. SymPAC: Scalable Symbolic Music Generation With Prompts And Constraints. arXiv preprint arXiv:2409.03055, 2024.

[4] BHANDARI, Keshav; COLTON, Simon. Motifs, Phrases, and Beyond: The Modelling of Structure in Symbolic Music Generation. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). Cham: Springer Nature Switzerland, 2024. p. 33-51.

[5] DE BERARDINIS, Jacopo; CANGELOSI, Angelo; COUTINHO, Eduardo. Measuring the structural complexity of music: from structural segmentations to the automatic evaluation of models for music generation. IEEE/ACM transactions on audio, speech, and language processing, 2022, 30: 1963-1976.

[6] Li, H., Zeng, D., Chen, L., Chen, Q., Wang, M., & Zhang, C. (2016). Immune multipath reliable transmission with fault tolerance in wireless sensor networks. In Bio-inspired Computing–Theories and Applications: 11th International Conference, BIC-TA 2016, Xi'an, China, October 28-30, 2016, Revised Selected Papers, Part II 11 (pp. 513-517). Springer Singapore.

[7] ZHU, Jinlong, et al. MMT-BERT: Chord-aware Symbolic Music Generation Based on Multitrack Music Transformer and MusicBERT. arXiv preprint arXiv:2409.00919, 2024.

[8] YU, Botao, et al. Museformer: Transformer with fine-and coarse-grained attention for music generation. Advances in Neural Information Processing Systems, 2022, 35: 1376-1388.

[9] GUO, Zixun; KANG, Jaeyong; HERREMANS, Dorien. A domain-knowledge-inspired music embedding space and a novel attention mechanism for symbolic music modeling. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2023. p. 5070-5077.

[10] JI, Shulei; YANG, Xinyu; LUO, Jing. A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges. ACM Computing Surveys, 2023, 56.1: 1-39.

[11] ROUARD, Simon, et al. Audio Conditioning for Music Generation via Discrete Bottleneck Features. arXiv preprint arXiv:2407.12563, 2024.

[12] FRADET, Nathan, et al. Impact of time and note duration tokenizations on deep learning symbolic music modeling. arXiv preprint arXiv:2310.08497, 2023.

[13] SHU, Yangyang, et al. MuseBarControl: Enhancing Fine-Grained Control in Symbolic Music Generation through Pre-Training and Counterfactual Loss. arXiv preprint arXiv:2407.04331, 2024.

[14] Zhang, Yan . “An innovative deep learning method for IoT malware identification.” Mari Papel y Corrugado Volume 2025: 29-37, doi:10.71442/mari2025-0004.

[15] Michael Simon, Salwa M. Din. Performance evaluation of self-organizing features in wireless sensor networks[J], TK Techforum Journal (ThyssenKrupp Techforum), Volume 2025 (1). 12-19.

[16] LI, Peike Patrick, et al. Jen-1: Text-guided universal music generation with omnidirectional diffusion models. In: 2024 IEEE Conference on Artificial Intelligence (CAI). IEEE, 2024. p. 762-769.

[17] Zhang, C., Li, M., & Wu, D. (2022). Federated multidomain learning with graph ensemble autoencoder GMM for emotion recognition. IEEE Transactions on Intelligent Transportation Systems, 24(7), 7631-7641.

[18] JI, Shulei; YANG, Xinyu. Emomusictv: Emotion-conditioned symbolic music generation with hierarchical transformer vae. IEEE Transactions on Multimedia, 2023, 26: 1076-1088.
[19] CHEN, Guan-Yuan; SOO, Von-Wun. Controllable Music Loops Generation with MIDI and Text via Multi-Stage Cross Attention and Instrument-Aware Reinforcement Learning. In: Proceedings of the 32nd ACM International Conference on Multimedia. 2024. p. 6851-6859.
[20] WU, Shih-Lun; YANG, Yi-Hsuan. MuseMorphose: Full-song and fine-grained piano music style transfer with one transformer VAE. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2023, 31: 1953-1967.