MUSIC PROOFREADING WITH REFINPAINT: WHERE AND HOW TO MODIFY COMPOSITIONS GIVEN CONTEXT

Pedro Ramoneda[♭]
Universtat Pompeu Fabra
Barcelona
pedro.ramoneda@upf.edu

Martin Rocamora
Universtat Pompeu Fabra
Barcelona
martin.rocamora@upf.edu

Taketo Akama
Sony Computer Science Laboratories
Tokyo
taketo.akama@sony.com

ABSTRACT

Autoregressive generative transformers are key in music generation, producing coherent compositions but facing challenges in human-machine collaboration. We propose RefinPaint, an iterative technique that improves the sampling process. It does this by identifying the weaker music elements using a feedback model, which then informs the choices for resampling by an inpainting model. This dual-focus methodology not only facilitates the machine's ability to improve its automatic inpainting generation through repeated cycles but also offers a valuable tool for humans seeking to refine their compositions with automatic proofreading. Experimental results suggest RefinPaint's effectiveness in inpainting and proofreading tasks, demonstrating its value for refining music created by both machines and humans. This approach not only facilitates creativity but also aids amateur composers in improving their work.

1. INTRODUCTION

Advanced autoregressive models [1, 2] have enabled the automatic generation of complex musical performances [3–7]. However, while autoregressive models generate music in a strictly forward-moving manner, human composers often follow a more iterative approach, frequently revisiting and refining earlier sections of a piece before proceeding [8–10]. Although there are some iterative methods for music generation [11–13], there are still areas for improvement in terms of controllability and human-in-the-loop aspects, such as inferring where to modify composition and inpainting capability to enable partial modification.

Iterative refinement proved effective for image generation; in particular, Lezama's Token-Critic [14] shows how feedback mechanisms can enhance image synthesis. Similarly, such feedback could benefit music composition for iteratively refining generated music. Within the spectrum of music composition tools, the Piano Inpainting Application (PIA) [15] stands out for its capabilities for automatic music generation that addresses the missing parts of musical performances, a technique referred to as inpainting. We highlight their handling of the musical context both before and after the selected gaps, enabling precise note-level inpainting. On account of that, inspired by image generation's success with iterative feedback and how PIA handles music context, our research explores applying these concepts to enhance controllability, human-in-the-loop functionality, and iterative refinement capability in automatic music generation.

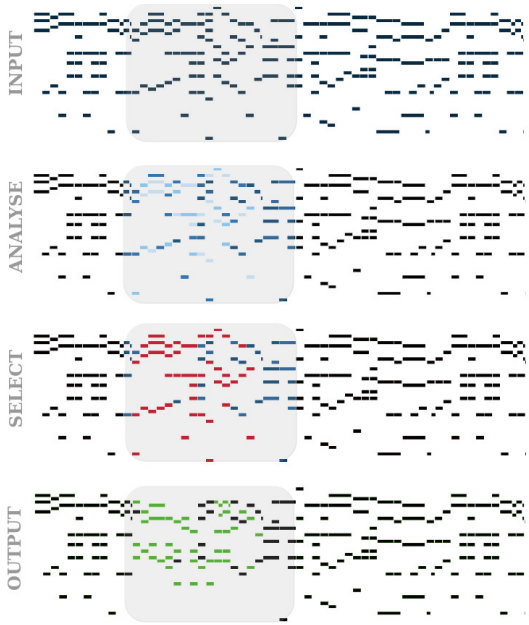

Figure 1: A user selects a MIDI section for enhancement (gray rectangle). Our methodology uses token-level feedback (blue) to highlight critical notes or sequences (red) for regeneration (green). This cycle repeats iteratively.

In this work, drawing from Token-Critic and PIA, we propose RefinPaint, which aims to boost automatic inpainting and proofreading in music generation. Our approach includes an iterative process of identifying areas in a composition needing modification and applying inpainting techniques to these areas. In this context, proofreading refers to automatically identifying and correcting errors or inconsistencies in a music composition. This dual-focus methodology facilitates the machine's ability to improve its automatic inpainting generation through repeated cycles, and offers a valuable tool for humans seeking to refine

♭ Work conducted at Sony Computer Science Laboratories, Inc. Tokyo.

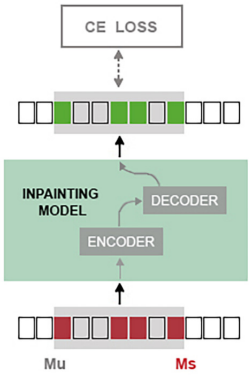

Figure 2: Encoder-decoder architecture for inpainting, given a user-provided mask M_u with a subset mask M_s.

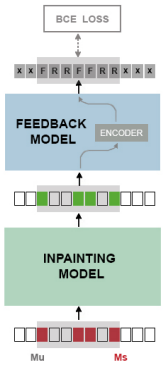

Figure 3: The Feedback algorithm identifies the most realistic tokens by training it to discern between real and synthetic music tokens.

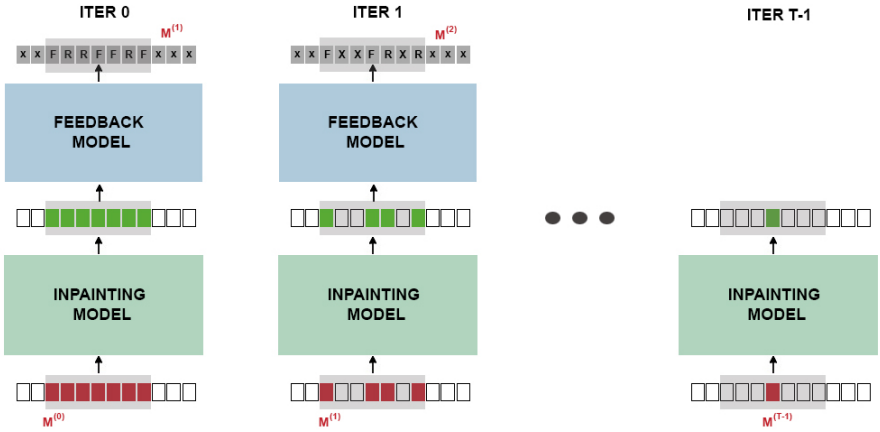

Figure 4: RefinPaint uses inpainting and feedback models to iteratively suggest changes, based on specific note feedback. It reduces the selected tokens in each iteration.

their compositions with automatic proofreading.

Our RefinPaint method is grounded in an autoregressive inpainting model to generate synthetic music tokens and a feedback model trained to distinguish between original and synthetic tokens. This differentiation is key during the sampling stage when deciding on token retention or revision. RefinPaint takes an iterative approach, integrating feedback into the inpainting model for selectively regenerating parts in each iteration, as Figure 1 shows. In contrast to Token-Critic, RefinPaint focuses on modifying a specific part of a composition using a contextual model and exposes the intermediate outputs of the autoregressive inpainting model to human inspection in each iteration.

The human-in-the-loop approach we propose allows for selecting the number of tokens to modify and revise the analysis heatmap at each iteration, as described in the following section. Through experimentation, we confirm RefinPaint's effectiveness in inpainting and proofreading tasks, demonstrating its utility for enhancing music created by both machines and humans. Finally, we provide a companion page featuring examples [1] and the code along with the trained models of RefinPaint for reproducibility [2].

[1] At: `https://refinpaint.github.io/`
[2] At: `https://github.com/ta603/RefinPaint`

2. METHODOLOGY

Our proposed methodology employs two models: an inpainting model $\mathcal{I}$, and a feedback model $\mathcal{F}$, alongside our iterative algorithm RefinPaint. Initially, $\mathcal{F}$ identifies areas within a MIDI file that need improvement based on the specific criteria described in Section 2.2. It uses a heatmap for detailed MIDI token-level feedback, allowing one to assess the context and relevance of each note in the selected region. Then, model $\mathcal{I}$ can regenerate the selected tokens considering the feedback, as described in Section 2.1. The methodology involves using both models iteratively with RefinPaint and encompasses three main stages: training the inpainting model (Section 2.1.1), training the feedback model (Section 2.2.1), and finally executing the iterative process for MIDI sequence generation (Section 2.3).

2.1 Inpainting model ($\mathcal{I}$)

The inpainting model aims to predict, or fill in, missing parts of a MIDI sequence based on a given mask. We adopt an encoder-decoder architecture for sequence-to-sequence tasks, as shown in Figure 2, inspired by the PIA study for music generation [15]. This model involves an encoder converting input data into a latent representation and a decoder predicting the final output.

With an anti-causal mask, self-attention within the encoder prevents future data access, while with a causal mask, self-attention within the decoder limits access only to previous data. With an identity mask, cross-attention enforces positional alignment between the encoder and decoder outputs, which is helpful for aligned sequence tasks.

The attention mechanisms are defined as follows, where M_{type} is the mask type (anti-causal, causal, or identity):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} \odot M_{\text{type}}\right) V. \tag{1}$$

This structure enhances the capability of the model to handle bidirectional input-output relationships, essential for inpainting, where future context influences the generation process. Furthermore, we add an extra binary embedding to the encoder input with information about the mask M_s–the tokens to regenerate–for the inpainting model.

2.1.1 Training the Inpainting Model ($\mathcal{I}$)

The training process is outlined in Algorithm 1. A batch x is sampled from the MIDI dataset $\mathcal{D}$, and a random fragment M_u is chosen for each sample in x with a length determined by t_1. It is important to note that t_1 refers to the length in terms of the token sequence, rather than the MIDI duration. Consequently, a random mask M_s, with the masking ratio controlled by $\gamma(t_2)$, is then applied to M_u. The forward pass of the model calculates the loss using the batch x, the mask M_s, and the Cross Entropy (CE) loss function to evaluate the difference between the predicted outputs and the actual labels. The model is subsequently updated via gradient descent. The function γ, a cosine scheduler, dynamically adjusts the masking ratio. It operates on a domain defined by a random variable t_2 within the interval $[0, 1]$. Specifically, for any chosen value t_2 drawn uniformly from the interval $[0, 1]$, the value undergoes a cosine transformation γ to determine the masking ratio, where $\gamma(t_2) = \cos\left(\frac{\pi t_2}{2}\right)$.

Algorithm 1 Training the Inpainting model ($\mathcal{I}$)

Require: MIDI dataset $\mathcal{D}$, Inpainting model $\mathcal{I}$
while convergence **do**
  $x \sim \mathcal{D}$ ▷ Sample batch
  $t_1 \sim U(0.1, 0.6)$, $t_2 \sim U(0, 1)$
  $M_u \leftarrow \text{Fragment}(x, t_1)$
  $M_s \leftarrow \text{Random Masking}(M_u, \gamma(t_2))$
  $L \leftarrow \text{ForwardInpaintingModel}(\mathcal{I}, x, M_s)$
  ▷ model forward and compute loss
  GradientDescent(L)
end while

2.2 Feedback model ($\mathcal{F}$)

We employ an encoder-only transformer architecture for the feedback phase that classifies music tokens as fake or real. We use this output distribution to select the k most realistic tokens to retain while the others are regenerated. Unlike the encoder-decoder inpainting model, $\mathcal{I}$, this model processes the input through a parallel and bidirectional attention mechanism without employing any attention masks, thus facilitating an unrestricted analysis of the musical context. Additionally, we add an extra binary embedding to the encoder input with information about the mask M_u–the selected fragment–for the feedback model.

2.2.1 Training the Feedback model ($\mathcal{F}$)

Algorithm 2 Training the Feedback model ($\mathcal{F}$)

Require: MIDI dataset $\mathcal{D}$, Inpainting model $\mathcal{I}$, Feedback model $\mathcal{F}$
while convergence **do**
  $x \sim \mathcal{D}$ ▷ Sample batch
  $t_1 \sim U(0.1, 0.6)$, $t_2 \sim U(0, 1)$
  $M_u \leftarrow \text{Fragment}(x, t_1)$
  $M_s \leftarrow \text{Random Masking}(M_s, \gamma(t_2))$
  $\hat{x} \leftarrow \mathcal{I}(x, M_u)$
  $L \leftarrow \text{ForwardFeedbackModel}(\mathcal{F}, \hat{x}, M_u)$
  ▷ model forward and compute loss
  GradientDescent(L)
end while

After training the inpainting model $\mathcal{I}$, we train an encoder-only feedback model $\mathcal{F}$. This model aims to evaluate the output from $\mathcal{I}$, offering feedback on the composition quality of each music fragment denoted by M_u.

One ideal way of training $\mathcal{F}$ would involve a vast dataset of computer- or human-generated music compositions and human experts' revisions for inpainting and proofreading applications. Instead, we propose a more feasible synthetic training strategy, described in Algorithm 2. The inpainting model $\mathcal{I}$ generates tokens within the selected fragment of a music piece, M_u, which we label as 'Fake', while we label as 'Real' the original unchanged tokens. We utilize these labels to instruct $\mathcal{F}$, following the process illustrated in Figure 3.

The training of $\mathcal{F}$ is based on the output of $\mathcal{I}$. We begin by sampling a batch x from the dataset $\mathcal{D}$, then apply masking M_s and M_u. Model $\mathcal{I}$ regenerates specific tokens within x, yielding a modified output $\hat{x}$. Model $\mathcal{F}$ then assesses each token of $\hat{x}$ against M_s, categorizing them as 'Real' or 'Fake'. The loss L for $\mathcal{F}$ is computed using the Binary Cross Entropy (BCE) loss function, and is minimized through gradient descent. The outcome is a heatmap for M_u, which indicates the probability of each token being 'Real' or 'Fake', determined by the sigmoid activation of the model output.

2.3 Generation of MIDI sequences (RefinPaint)

We capitalize on the strengths of the inpainting and feedback models for the iterative MIDI sequence generation. The process shown in Figure 4 begins with a MIDI sequence x introduced by the user, setting the stage for a loop that spans a predetermined number of iterations T.

Initially, the user selects the fragment to be modified $x_m^{(0)}$ and sets the initial selection rate $k = 0$ for complete inpainting. Alternatively, different values for k allow the

user to control how much of the content to keep in the selected fragment when proofreading.

In the proposed Algorithm 3, at each iteration t, the inpainting model $\mathcal{I}$ generates a new version of the sequence $\hat{x}$, based on the current masked input $x_m^{(t)}$. In the human-in-the-loop scenario, the user can then adjust this generated sequence. The feedback model $\mathcal{F}$ evaluates $\hat{x}$ and provides a new mask $M^{(t+1)}$, which the user may also modify. This mask highlights the tokens that are deemed most realistic. The number of selected realistic tokens k follows a decreasing function γ of the iteration t, which models the increasing confidence in the tokens produced over time. Moreover, we add an extra binary embedding to the encoder input with information about the mask M–the given context–where M changes over iterations.

Refining the music sequence through each iteration aims to achieve a compositional process that closely aligns with that of a human composer so that the user intervention becomes interpretable and natural. It fosters a collaborative environment between the user and the machine and tailors the generation process to the user's specific directives and preferences.

Algorithm 3 Generation Algorithm (RefinPaint)

Require: Inpainting model $\mathcal{I}$, Feedback model $\mathcal{F}$, masked MIDI $x_m^{(0)}$, No. masked tokens N, No. iterations T

for $i = 0$ to $T - 1$ **do**
  $k = \lceil \gamma\left(\frac{i}{T}\right) \cdot N \rceil$
  $\hat{x} \leftarrow \mathcal{I}(x_m^{(i)})$
  if $i \neq T - 1$ **then**
    $M^{(i+1)} \leftarrow \mathcal{F}(\hat{x})$
    $x_m^{(i+1)} \leftarrow$ k-realistic tokens$(\hat{x}, M^{(i+1)}, k)$
  end if
end for

3. RELATED WORK

Automatic music generation has rapidly advanced recently. Significant progress has been made [4–6], especially in solo piano compositions [3, 7, 15], through the capabilities of autoregressive models in producing coherent musical outputs. However, several challenges remain for creating successful interactions with humans [3, 11, 15–22].

Previous work has explored various approaches to generate music iteratively and allowed for partial modification—often referred to as inpainting—, which enhances controllability. Among them, sequential handling of musical elements has been a common strategy, as in models like DeepBach [11] and Coconet [12]. Although these models allow for inpainting and iterative generation, they often rely on random iterations without a mechanism for discriminative feedback to guide improvements. This lack of directed refinement contrasts with the human compositional process, which typically involves iterative improvements based on evaluative feedback. Our proposed approach addresses this limitation by incorporating a feedback model that identifies areas for improvement for both humans and machines to refine the composition.

Although it is not designed as an inpainting model, ES-Net's approach to music generation integrates generative and discriminative capabilities in one model [13], with a feature for correcting past errors for iterative refinement. Our model differs significantly: it takes into account the context of the selected fragment, could improve any existing inpainting model, and can handle general MIDI formats. In [23], the authors propose a GAN model for piano music composition with a discriminator model that discerns real and fake compositions in the training process. However, it does not give feedback on which generated parts are good or bad and does not create compositions iteratively. Yet, the application of discriminative feedback in music generation, particularly in a manner that mimics human iterative refinement, remains largely unexplored.

Finally, inpainting models in music have seen various approaches but remain less studied compared to their counterparts in image generation [24]. They typically focus on quantized scores, with significant contributions like Gibbs sampling for Bach chorales [11] and RNN-based melodies inpainting [25]. Studies on transformers for multitrack inpainting have advanced the field, such as MMM [26], which utilizes a decoder architecture akin to GPT2 [2], and PIA [15], which uses a specialized transformer design. We chose PIA over MMM as a ground element in this work, given it is capable of working in the token level or larger contexts and inpainting multiple little fragments at the same time, similar to Token-Critic's generator [14].

4. EXPERIMENTAL SETUP

4.1 Data preparation

Our study utilizes the Lakh MIDI dataset (LMD), an extensive collection of approximately 170k unique multitrack MIDI files, compiled by Colin Raffel for music research [27]. The dataset offers a wide variety of music, albeit with varying quality due to its internet-sourced nature. Despite this, the volume and diversity of the LMD dataset make it a valuable asset for our proofreading task. We extracted only the piano parts, totaling 120,000 tracks.

We tokenize the piano tracks using REMI (REvamped MIDI-derived events) [16], a music representation method that converts MIDI events into a structured format optimized for Transformer-based models that significantly enhances their ability to comprehend and produce music. REMI categorizes music elements into distinct event types, including timing for rhythm and note events for melody, but we exclude velocity events for simplicity. Specifically, we use a modified version of REMI tailored for handling single-track piano performances, as implemented in [28]. The dataset was split into training (hashes 0–d), validation (hash e), and testing (hash f) segments, based on each file's MD5 hash's leading digit, akin to previous methods [5, 6]

4.2 Model development

We train the inpainting and feedback models with the AdamW optimizer, using eighty per cent of the dataset for training and the remainder for validation. Each epoch consists of a randomly selected fragment from the training set,

512 tokens in length. We also employ an augmentation procedure that transposes the pitch tokens of a sequence by adding or subtracting up to 6 semitones. For the inpainting model, we apply a cross-entropy loss and use the maximum batch size that our system can handle; a single V100 GPU with 16GB allows for 48 samples. The encoder-decoder inpainting model comprises 12 layers: 4 encoder layers and 8 decoder layers, similar to the original PIA, with 8 heads and an embedding dimension of 512. We employ a cosine scheduler for training, with 16,000 warmup steps, reaching up to a 0.0006 learning rate. The feedback model consists of 6 layers, with an embedding dimension of 512, a dropout rate of 0.1, 8 heads, and the same cosine scheduler. Finally, we acknowledge that optimizing these models was not the main focus of this paper, so there might be better hyperparameter values.

In the particular case of proofreading without human intervention, i.e. for evaluation purposes, the final output is the iteration that maximizes the feedback model probability distribution. Using a sigmoid function, the model determines whether each token in a sequence is fake or real. By averaging the output probabilities, we calculate a global feedback score (GFS) for the sequence's overall realism and select the best regeneration output based on it.

5. INPAINTING RESULTS

5.1 Divide and conquer with the inpainting model

We conducted an experiment to explore how the model's inpainting performance is affected by the percentage of tokens to inpaint in a selected fragment. We hypothesize that the more tokens to inpaint, the harder the problem is, so the model performance is lower. The experiment uses the inpainting model trained as detailed in section 2.1.1, and we report its Negative Log-Likelihood (NLL) loss and perplexity of the next predicted token. The evaluation covered the entire test set, with masking ratios ranging from 1 (fully masked) to 0 (no tokens to inpaint) and a fixed 30% fragment size rate of the 512 tokens sequence. Results shown in Table 1 indicate better performance with reduced masking, confirming our hypothesis. Notably, the average Perplexity value is less than half at 0.05 compared to the 1.0 masking ratio. This finding is crucial for RefinPaint's effectiveness as it reduces the number of tokens to be inpainted in subsequent iterations, considering the iterative process as a top-to-bottom strategy.

5.2 Objective evaluation of proofreading inpainting

This section conducts a comparative analysis between the reference inpainting output, as described in [15] (PIA), and our enhanced method. Our method applies the RefinPaint proofreading process to the initial PIA's inpainting output over ten iterations and is referred to as 'Ours'. For fragment sizes of 50%, 30%, and 10% of the 512-token test sequences, we computed 1,000 instances each. It is important to note that the PIA method discussed is our reimplementation, since the original code was not available.

Table 2 shows the average global feedback score (GFS), computed as explained in Section 4.2, and the number of evaluations in which each algorithm outperforms the other (Wins) and in which their scores are the same (Ties). Table 3, on the other hand, focuses on the comparison between PIA and Ours, employing the NLL loss, a metric of the next token prediction in generated music. This metric, derived from an autoregressive model we trained explicitly from scratch to assess the inpainting results, is a benchmark metric in our evaluation. Similar evaluations have been employed in previous studies in natural language processing [29] and music generation [30]. Consequently, our study employs a 12-layer Transformer-based autoregressive model with REMI representation. Our goal is to assess the similarity between the distribution of musical elements in inpainted sections and those in the original dataset, including aspects such as rhythms, harmony, or melodies. A lower NLL loss indicates a more accurate prediction of the next token, reflecting a closer approximation to the dataset's inherent musicality. Note we assess this metric over the entire output sequence.

masking ratio	NLL	AVG PPL
0.05	0.56	0.31
0.10	0.58	0.33
0.15	0.58	0.34
0.20	0.58	0.33
0.40	0.64	0.41
0.60	0.70	0.49
0.80	0.77	0.59
1.00	0.86	0.73

Table 1: Summary of the inpainting experiment with different masking ratios. A masking ratio of 1.0 corresponds to being fully masked, and 0 indicates no masking. The standard deviation is less than 0.01 in all the experiments.

	GFS (↑)		Wins		Ties
	PIA	Ours	PIA	Ours	
50%	0.458	**0.696**	0	**870**	130
30%	0.515	**0.730**	0	**886**	114
10%	0.650	**0.803**	0	**891**	209

Table 2: Comparison of global feedback scores (GFS) between PIA and the proposed RefinPaint methodology, Ours. Higher values indicate better performance.

	NLL (↓)		Wins		Ties
	PIA	Ours	PIA	Ours	
50%	2.01	**1.97**	330	**541**	129
30%	1.68	**1.66**	347	**533**	120
10%	1.63	**1.62**	321	**457**	222

Table 3: Comparison of Negative Log Likelihood (NLL) between PIA and the proposed RefinPaint methodology, Ours. Lower values indicate better performance.

Results in Table 2 indicate that our model's GFS score is generally better than the baseline, suggesting that the optimization goal of the RefinPaint iterative process is met.

The PIA model never wins because this experiment selects the best GFS of all the iterations, as mentioned in Section 4.2. Although dynamic programming or genetic algorithms could enhance the process, this study uses a simpler method, focusing on the iteration with the highest GFS.

In Table 3, RefinPaint consistently achieves a slightly lower average NLL loss than PIA, suggesting that the inpainted content by RefinPaint is more consistent with the original dataset used for training. Furthermore, RefinPaint wins more evaluations than PIA across all the percentages of fragment size evaluated. This further underscores the enhanced performance of RefinPaint in producing sequences more akin to human compositions. However, comparing both tables, we acknowledge that higher GFS does not always imply a better NLL loss, calling for other types of evaluation, as addressed in the next section.

5.3 Listening test of proofreading inpainting

While computational metrics provide valuable insights into the quality of our inpainted music sections, human perception adds another perspective for evaluating musical quality and appeal. A user-based evaluation was conducted to capture a holistic view of the inpainted outputs' musicality.

For each experiment, which involved 50%, 30%, and 10% fragments of inpainted content, 15 different annotators evaluated both the first iteration of inpainted content (PIA) and the complete iterative process of RefinPaint (Ours) for ten iterations. Participants were exposed to two scenarios, Experiment 1 and Experiment 2: one from the PIA model and one from our RefinPaint model. The order in which these pairs were presented was randomized to avoid any bias. Additionally, we provided the original music fragment without the inpainted content for reference. Participants listened to both the PIA and RefinPaint versions before making their evaluations. They were asked to assess the inpainted content's quality by comparing it to the original fragment, focusing specifically on coherence and creativity. To make their choice, participants were given four options to prevent bias: 'Experiment 1,' 'Maybe Experiment 1,' 'Maybe Experiment 2,' and 'Experiment 2'.

Figure 5 shows the listening test results. Firstly, PIA got lower preference scores than RefinPaint for the different fragment size conditions. In addition, RefinPaint's performance for different fragment sizes shows that the coherence scores increase as the fragment size gets larger, even if the creativity varies. This means that as there is more to inpaint, RefinPaint gets better at being coherent. In contrast, PIA does not show such a strong trend.

The quantitative and qualitative evaluations point towards a clear trend: Refinpaint tends to yield superior inpainting results when proofreading machine inpainted sections compared to the baseline. Our methodology produces music sequences that are more consistent, perceptually closer to the original, and preferred by listeners.

6. CASE OF STUDY ON PROOFREADING AMATEUR COMPOSITIONS

We conducted an additional study to explore the proposed system's capabilities for proofreading music compositions

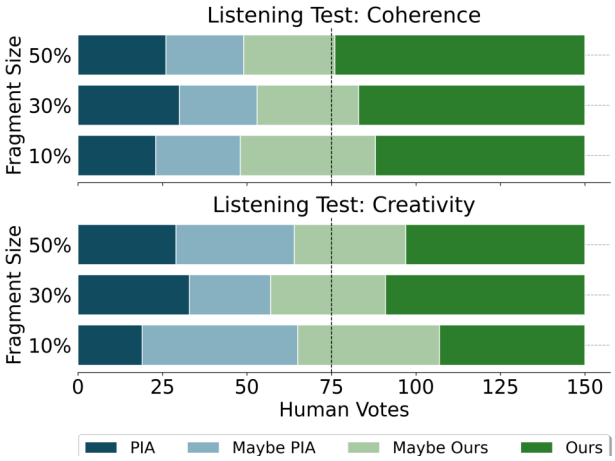

Figure 5: Results of the participants' votes for the listening test comparing PIA and RefinPaint (Ours) along different fragment sizes (50%, 30%, and 10%).

by humans. Given the intrinsic difficulties of such a study and due to practical restrictions, we limited our experiment to four amateur composers–two with classical music training and two with modern popular music training.

Participants used a straightforward proofreading interface that enables bar selection for regeneration, allowing them to choose how much of the content to keep in certain sections of their work, as described in 2.3. Additionally, we allowed the users to change the RefinPaint feedback in the selected area and experiment with the tools by conducting as many trials as they wanted.

After testing our inpainting tool on a 30-second music piece, participants responded to questions about their experience. They evaluated whether the tool (i) enhanced their original draft, (ii) sparked new ideas, (iii) could save time over manual proofreading, and (iv) was something they would use in the future. All chose "yes" for (i), (iii) and (iv) with three "yes" and one "maybe" for (ii), suggesting time efficiency as a key advantage and providing an overall positive view of the tool.

The positive feedback prompted us to showcase the proofread compositions on our companion website. Participants suggested the tool could be particularly effective in overcoming creative blocks, noting that inspiring ideas stemmed from all iterations, not just the last one. Additionally, two participants especially valued the option to alter tokens within the RefinPaint selection.

7. CONCLUSION

In conclusion, our novel approach, RefinPaint, significantly enhances music generation by identifying and improving weaker musical elements through iterative feedback. Its effectiveness in both inpainting and proofreading tasks promises a new direction for creative assistance and quality enhancement in compositions by humans and machines alike. Future work could fruitfully extend the research to multitrack compositions and explore control mechanisms for this model, such as conditioning by harmony, rhythm, genre, or other musical factors.

8. ETHICS STATEMENT

While RefinPaint can represent a significant leap forward in music composition technology, ensuring ethical deployment and use is crucial. We advocate for a future where such technologies support and enrich the creative process, complementing rather than displacing human creativity. While RefinPaint aims to democratize music creation, making it accessible and achievable for amateurs, there is a risk that professional musicians and composers could feel their roles and contributions are being undermined or replaced by machines. It is essential to strike a balance where this technology serves as a tool for enhancement and learning rather than a substitute for human creativity. Furthermore, it will be vital to establish guidelines that protect the intellectual property rights of original compositions, whether entirely human-made or AI-assisted.

9. REFERENCES

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

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations, (ICLR)*, 2019.

[4] S.-L. Wu and Y.-H. Yang, "Compose & embellish: Well-structured piano performance generation via a two-stage approach," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[5] J. Thickstun, D. Hall, C. Donahue, and P. Liang, "Anticipatory music transformer," *arXiv preprint arXiv:2306.08620*, 2023.

[6] B. Yu, P. Lu, R. Wang, W. Hu, X. Tan, W. Ye, S. Zhang, T. Qin, and T.-Y. Liu, "Museformer: Transformer with fine-and coarse-grained attention for music generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] N. Fradet, J.-P. Briot, F. Chhel, A. E. F. Seghrouchni, and N. Gutowski, "Byte pair encoding for symbolic music," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[8] D. Collins and M. Dunn, "Problem-solving strategies and processes in musical composition: Observations in real time," *Journal of Music, Technology & Education*, vol. 4, no. 1, pp. 47–76, 2011.

[9] P. Burnard, *Musical creativities in practice*. OUP Oxford, 2012.

[10] B. Jacob, "Algorithmic composition as a model of creativity," *Organised Sound*, vol. 1, pp. 157 – 165, 1996.

[11] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *International Conference on Machine Learning (ICML)*, 2017.

[12] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proc. of the 18th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2017.

[13] W. Chi, P. Kumar, S. Yaddanapudi, R. Suresh, and U. Isik, "Generating music with a self-correcting non-chronological autoregressive model," in *Proc. of the 21th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2020.

[14] J. Lezama, H. Chang, L. Jiang, and I. Essa, "Improved masked image generation with token-critic," in *European Conference on Computer Vision (ECCV)*, 2022.

[15] G. Hadjeres and L. Crestel, "The piano inpainting application," *arXiv preprint arXiv:2107.05944*, 2021.

[16] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[17] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019.

[18] ——, "Connective fusion: Learning transformational joining of sequences with application to melody creation," in *Proc. of the 21st Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2020.

[19] ——, "A contextual latent space model: Subsequence modulation in melodic sequence," in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2021.

[20] C. Payne, "Musenet," https://openai.com/blog/musenet, 2019.

[21] G. Hadjeres and L. Crestel, "Vector quantized contrastive predictive coding for template-based music generation," *arXiv preprint*, 2020.

[22] Y.-J. Shih, S.-L. Wu, F. Zalkow, M. Muller, and Y.-H. Yang, "Theme transformer: Symbolic music generation with theme-conditioned transformer," *IEEE Transactions on Multimedia*, 2022.

[23] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola, "Symbolic music generation with transformer-gans," in *35th AAAI Conference on Artificial Intelligence*, 2021.

[24] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image inpainting: A review," *Neural Processing Letters*, vol. 51, pp. 2007–2028, 2020.

[25] G. Hadjeres and F. Nielsen, "Anticipation-rnn: Enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, vol. 32, no. 4, pp. 995–1005, 2020.

[26] J. Ens and P. Pasquier, "Mmm: Exploring conditional multi-track music generation with the transformer," *arXiv preprint arXiv:2008.06048*, 2020.

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

[28] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah Seghrouchni, and N. Gutowski, "MidiTok: A python package for MIDI file tokenization," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[29] A. Wang and K. Cho, "BERT has a mouth, and it must speak: BERT as a Markov random field language model," in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, Jun. 2019.

[30] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning (ICML)*, 2018.