RNINFER: A LARGE LANGUAGE MODEL APPROACH TO FUNCTIONAL HARMONIC REASONING IN SYMBOLIC MUSIC

Anonymous Authors
Anonymous Affiliations
anonymous@ismir.net

ABSTRACT

Existing machine learning models for Roman Numeral Analysis (RNA) treat the task as a classification problem, providing labels without the explanatory reasoning that is central to music theory. This "black box" approach misaligns with the goal of harmonic analysis, which is to deepen musical understanding. To address this, we introduce RNInfer, a novel framework that bridges a pre-trained symbolic music encoder with a Large Language Model (LLM) to perform interpretable RNA. Our architecture uses a lightweight projector to align musical features with the LLM's embedding space, enabling it to reason about harmonic content. We propose Octuple+, an enhanced tokenization scheme that incorporates crucial enharmonic spelling information into the music encoder. The model is trained in two stages: supervised fine-tuning to learn the analysis task, followed by reinforcement learning with Group Relative Policy Optimization (GRPO) to generate human-readable reasoning traces without requiring annotated examples. Our experiments show that RNInfer achieves competitive accuracy on the primary analysis task, and we demonstrate its capability to generate structured explanations for its predictions, marking a critical step toward more transparent and pedagogically useful models for computational musicology.

1. INTRODUCTION

Roman Numeral Analysis (RNA) is a foundational task in music theory, providing a systematic method for labeling the harmonic function of chords within a tonal context [1]. Automating this process has been a long-standing goal in computational musicology. Early machine learning approaches have made significant strides, with models like AugmentedNet [2] and RNBert [3] demonstrating the potential of deep learning for this task. These systems have explored various architectures, from multi-task learning frameworks that predict related tonal features simultaneously [2, 4] to methods that enforce label consistency using autoregressive models [5].

However, these prior works are fundamentally limited by treating RNA as a classification task. Music analysis is subjective and allows multiple valid interpretations - its key purpose to deepen understanding of musical structure and capture its full richness. Producing harmonic labels without explanation falls short of this goal.

To address these challenges, we turn to Large Language Models, leveraging architectures from vision-language models [6–8] to bridge symbolic music and natural language reasoning. We employ Group Relative Policy Optimization (GRPO) [9] to elicit reasoning capabilities via reinforcement learning with rule-based rewards. allowing the model to generate detailed interpretations without depending on human-annotated reasoning traces.

We introduce RNInfer, a novel framework to perform interpretable Roman Numeral Analysis. RNInfer leverages the symbolic music understanding of a customized MusicBERT, the generative power of an instruction-tuned LLM, and instills reasoning ability via GRPO. Our primary objective is to produce accurate RNA predictions accompanied by human-readable explanations that illuminate the underlying harmonic logic.

While some studies suggest that general-purpose LLMs may underperform specialized models on specific benchmarks [10, 11], their value lies in their capacity for cross-task generalization rather than narrow task-specific accuracy. Our goal is to equip LLMs with deep domain understanding that enables synergistic performance across multiple related tasks. For instance, a model with robust symbolic music comprehension could not only analyze musical structures but also leverage this understanding for controllable generation and interactive music education. Since current LLMs still lack nuanced symbolic music understanding, systems like RNInfer represent important steps toward interpretable and generalizable music AI.

In summary, our contributions are threefold:

- We are the first to apply an LLM to the task of functional harmonic analysis, combining symbolic music encoders with natural language reasoning.
- We propose Octuple+, an enhanced tokenization that integrates spelled pitch into the MusicBERT framework while retaining its pre-trained knowledge, yielding improved performance.
- We develop a model that generates human-readable reasoning traces alongside RNA predictions, taking an initial step toward interpretable models that align with the explanatory purpose of harmonic analysis.

© Anonymous Authors. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Anonymous Authors, “RNInfer: A Large Language Model Approach to Functional Harmonic Reasoning in Symbolic Music”, submitted to *ISMIR*, 2025.

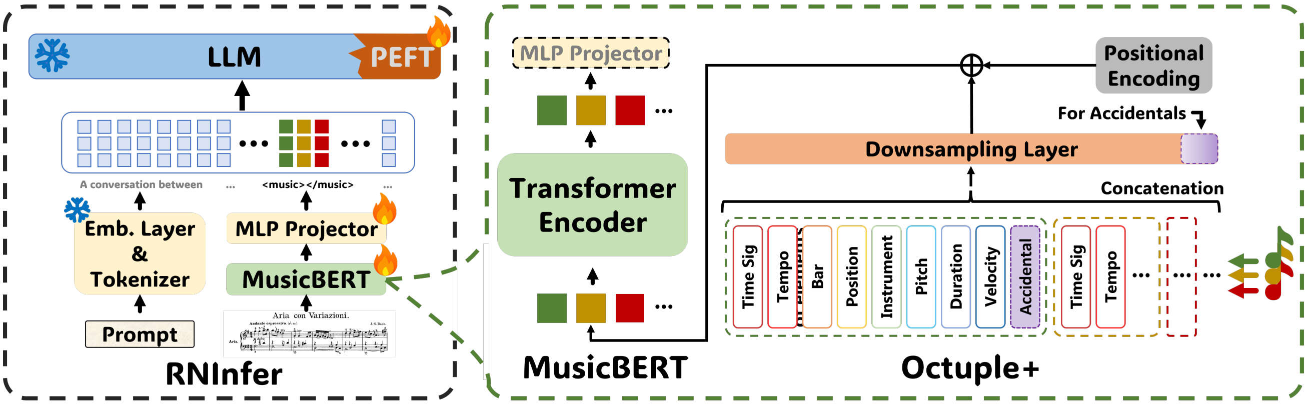

Figure 1. The overall architecture of RNInfer, showing the music encoder, projector, and LLM components.

2. RELATED WORK

Automated Roman Numeral Analysis has evolved significantly with machine learning. Early works established two main research trajectories. The first began with multi-task BiLSTM models [12] and progressed to Transformer-based architectures like Harmony Transformer [13, 14], which, however, were limited by their reliance on piano-roll representations that lack enharmonic information.

A second, parallel line of research emphasized the importance of spelled-pitch representations [4, 15] and explored various architectures to improve performance. These include CRNNs with multi-task learning [2] and autoregressive models designed to enforce label consistency [5]. In addition to these two lines of work, RN-Bert [3] achieved state-of-the-art results by fine-tuning MusicBERT [16], but it inherited MusicBERT's limitation of being unable to take spelled pitch as input.

Despite these advances in accuracy, existing models share fundamental limitations. Most approaches treat RNA at the onset level, assigning a label to every note onset [2, 3]. This contrasts with human analysis, where harmonic spans are identified first and non-chord tones are excluded. Onset-level models, by forcing harmonies onto such tones, create an unnatural and less faithful workflow. Most importantly, they function as "black boxes," providing labels without the explanatory reasoning that is central to the purpose of music analysis. Our work addresses these gaps by reframing RNA as a reasoning task for a multimodal LLM, while retaining crucial spelled-pitch information.

We adopt an architecture inspired by vision-language models (VLMs) like LLaVA [6–8, 17], which use a simple projector to efficiently align features from a specialized encoder with a LLM. To instill reasoning capabilities without requiring human-annotated reasoning traces, which is a key limitation of methods like Chain-of-Thought (CoT) [18], we employ Group Relative Policy Optimization (GRPO) [9, 19]. GRPO uses rule-based rewards to train the model to generate explanatory outputs, making it suitable for specialized domains like music analysis where annotated reasoning data is scarce.

3. METHODOLOGY

We introduce RNInfer, a novel framework designed for interpretable Roman Numeral Analysis. Our architecture bridges a pre-trained symbolic music encoder with a LLM using a lightweight projector, enabling the LLM to reason about musical content.

3.1 System Architecture and Workflow

Figure 1 illustrates the overall architecture of our proposed RNInfer framework, which consists of three core components: First, a customized MusicBERT or RNBert [3, 16] encoder processes symbolic music tokenized with our proposed Octuple+ scheme (see 3.2) to generate contextual embeddings. Next, a lightweight two-layer MLP projector maps these music embeddings into the latent space of the Qwen2.5-7B-Instruct LLM. The LLM then receives the projected music embeddings alongside a task prompt and autoregressively generates the final analysis, including both the RNA labels and a natural-language explanation.

3.2 Octuple+ Spelled-Pitch Tokenization

A key limitation of MusicBERT and RNBert is their reliance on MIDI pitch, which ignores enharmonic distinctions essential for nuanced harmonic analysis. To address this, we propose **Octuple+**, an enhanced tokenization scheme that adds a ninth attribute for accidental spelling to MusicBERT's original eight-tuple representation.

To incorporate this new attribute without disrupting the pre-trained knowledge, we draw inspiration from the initialization strategy of LoRA [20]. The embedding for each note event, originally composed of eight token embeddings E, is now extended with a ninth embedding for the accidental, E_{acc}, forming a new sequence $E' = [E; E_{\text{acc}}]$. This sequence is projected to a single note embedding by a downsampling layer. We carefully initialize this layer to 0 so that its output for E' is identical to its output for E at the start of training. This ensures the contribution from the new accidental token is initially zero, allowing the model to begin fine-tuning from its pre-trained state and gradually learn to incorporate the new spelling information, which stabilizes the training process.

3.3 Two-Stage Training Procedure

Our training process consists of two phases: Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL).

Phase 1: Supervised Fine-Tuning (SFT). In this phase, we train the model to produce correct analytical outputs using a two-stage curriculum. We first fine-tune on local key prediction (SFT-Key), a foundational sub-task of RNA, and then continue training on the full Roman Numeral Analysis task (SFT-RNA). This curriculum is crucial for performance (c.f. Section 4.4). The model is trained with a standard cross-entropy loss. For parameter efficiency, we only train the music encoder, projector, and LoRA [20] adapters in the LLM, which constitute 2% of the total parameters.

Phase 2: Reinforcement Learning (RL). Following SFT, we apply the GRPO [9] algorithm to train the model to generate human-readable reasoning traces. In this phase, the model learns to produce a structured output containing a `<think>` block with step-by-step reasoning, followed by an `<answer>` block with the final RNA label. The model is rewarded based on rule-based functions, obviating the need for human-annotated reasoning datasets.

3.4 Data Processing

To create musically coherent training samples and reduce computational cost, we employ a "phrase-slicing" strategy, segmenting music along phrase boundaries. This required curating a new dataset, P.DCML, from pieces with existing phrase annotations in the DCML Corpora [21], the composition of which is detailed in Appendix A.1. As our ablation study in Section 4.4 confirms, this musically-informed segmentation significantly improves model performance.

Our data preprocessing and augmentation largely follow the procedures of RNBert [3], but with two key enhancements to create a more musically complete dataset: First, we add support for augmented sixth chords, which are omitted in RNBert. Second, we expand the annotation of secondary chords to include their quality in addition to their scale degree, as this information is necessary for a complete and accurate RN label.

4. EXPERIMENTS

We evaluate RNInfer's architecture and training strategy through a series of experiments. We first report the model's performance on the supervised tasks of local key prediction and full Roman Numeral Analysis. We then analyze the effects of the subsequent RL stage, which is designed to instill reasoning capabilities.

4.1 Validating the Projector Module

To validate the projector's ability to bridge the music encoder and LLM, we designed a simple toy task requiring the model to identify simultaneous notes, training only the projector while keeping the other components frozen. The near-perfect accuracy shown in Table 1 confirms that the projector effectively maps musical features into the LLM's latent space, establishing the viability of our architecture for more complex harmonic analysis.

Model	Position	Group	Both
RNInfer	99.51%	99.74%	99.35%

Table 1. Accuracy (%) on the simultaneous note determination task for note position, note group, and both.

Model	Dataset	Acc. (%)
RNInfer	BPS	80.33
RNBert	BPS	**81.50**
RNInfer	P.DCML	79.36
RNBert	P.DCML	**85.76**

Table 2. Accuracy (%) for the SFT-Key task. We report RNBert results from our own training run, following the original authors' instructions.

4.2 Supervised Fine-Tuning (SFT)

Our SFT phase follows a two-stage curriculum. We first fine-tune on local key prediction (SFT-Key), a foundational sub-task that stabilizes training. As shown in Table 2, RNInfer achieves competitive accuracy on the BPS and P.DCML datasets. A confusion matrix (see Appendix A.4) confirms that most errors are musically plausible, occurring between closely related keys.

Next, we fine-tuned the model on the full RNA task. Unlike prior work that uses an onset-level classification approach, RNInfer adopts a more human-aligned, harmony-level perspective, identifying harmonic spans and assigning a single label to each. This approach naturally handles non-chord tones and better reflects analytical practice.

Table 3 shows that RNInfer outperforms strong baselines, though it does not surpass the state-of-the-art RNBert. The performance gap may be explained by differences in the task setup: RNInfer tackles a more challenging open-ended generation task, whereas RNBert performs classification. Furthermore, RNInfer operates on a more granular output space, distinguishing between 42 enharmonically distinct keys versus RNBert's 24, and supports the recognition of augmented sixth chords and secondary chord qualities, which RNBert cannot handle.

4.3 Reinforcement Learning for Reasoning (RL-RNA)

Finally, we employ GRPO to train the model to generate reasoning traces. We design three rule-based reward functions: one to enforce the output format, one to encourage an appropriate reasoning length, and one to reward the accuracy of the final answer.

As shown in Table 4, though the average reward for all three reward functions increases throughout RL training (see Appendix A.6) the final task accuracy is decreased.

Model	Training set	Test set	Acc. (%)
RNInfer	P.DCML	BPS	**58.3**
AugN [2]	AugNData	BPS	45.4
Mi20 [4]	Mi20Data	BPS	42.8
RNInfer (8B)	P.DCML	P.DCML	52.8
RNInfer	P.DCML	P.DCML	49.5
RNBert	P.DCML	P.DCML	**64.5**

Table 3. Results for the SFT-RNA task. The P.DCML dataset is comparable in size to those used for AugN and Mi20 (Appendix A.1). RNInfer (8B) employs Qwen3-8B, while other RNInfer variants use Qwen2.5-7B-Instruct, showing that stronger backbones improve performance.

Model	Training Stage	Dataset	Acc. (%)
RNInfer	SFT-RNA	BPS	**58.28**
RNInfer	RL-RNA	BPS	38.67

Table 4. Accuracy (%) comparison before and after RL-RNA. Due to the high computational cost of GRPO, this experiment was conducted on the BPS subset.

We identify three reasons for this: First, the model struggled to consistently adhere to the strict JSON format, leading to parsing errors. Unlike the token-level supervision in SFT, the sample-level reward signal from RL is less efficient for enforcing specific syntactic rules. Second, unlike other domains, we lack formal verifiers for the correctness of the reasoning trace itself, allowing the model's behavior to deviate from the intended analytical path. Third, the computational cost of RL limited training to a single epoch on the smaller BPS dataset (see Appendix A.1), which was likely insufficient and led to underfitting. We provide reward curves and generation examples in Appendix A.6.

4.4 Ablation Study

We conduct a series of ablation studies to validate our key design choices. All experiments in this section were performed on the P.DCML dataset.

Impact of Slicing Strategy. We compare two data slicing strategies: a simple measure-based approach and a more musically-informed phrase-based strategy. As shown in Table 5, phrase-based slicing yields a significant improvement in accuracy on the SFT-Key task. This confirms our hypothesis that providing musically coherent segments aligns better with the nature of harmonic analysis and improves model performance.

Slicing Strategy	Acc. (%)
Measure-based	73.90
Phrase-based	**79.46**

Table 5. Impact of slicing strategy on the SFT-Key task using the RNBert encoder.

Impact of Octuple+ Tokenization. We compare models trained with and without the additional accidental bit. Table 6 shows that explicitly providing spelled pitch information improves key prediction accuracy, while it may be feasible for a model to implicitly learn to infer spelled pitch from MIDI sequences [22].

Tokenization	Key Acc. (%)
Octuple (w/o accidental)	77.04
Octuple+ (w/ accidental)	**79.46**

Table 6. Impact of Octuple+ on the SFT-Key task using the RNBert encoder.

Music Encoder	Acc. (%)
MusicBERT (pre-trained)	21.69
RNBert (fine-tuned)	**49.51**

Table 7. Impact of music encoder on the SFT-RNA task.

Music Encoder: Pre-trained vs. Fine-tuned. We compare a pre-trained MusicBERT encoder and an RNBert encoder fine-tuned on RNA tasks. As shown in Table 7, the fine-tuned RNBert provides a substantial performance boost on the SFT-RNA task. This demonstrates that using a domain-adapted music encoder is crucial for capturing important features for detailed harmonic analysis.

Impact of SFT Curriculum. Finally, we validate our two-stage SFT curriculum. We compare training the SFT-RNA model from scratch versus initializing it from a checkpoint pre-trained on the SFT-Key task. Table 8 shows that the curriculum approach dramatically improves performance. This confirms that pre-training on the foundational subtask of key prediction is a necessary step to effectively learn the more complex full RNA task.

SFT-RNA Training	Acc. (%)
From Scratch	29.44
With SFT-Key curriculum	**49.51**

Table 8. Impact of the two-stage SFT curriculum.

5. CONCLUSION & FUTURE WORK

In this paper, we introduced RNInfer, a novel framework for interpretable Roman Numeral Analysis that bridges a symbolic music encoder with an LLM. Our approach, featuring Octuple+ tokenization and a two-stage training curriculum, achieves competitive accuracy and marks a critical step toward transparent, pedagogically useful models for computational musicology. In future work, we plan to use external LLMs as automated evaluators to provide a direct reward signal for the reasoning traces. We also intend to explore methods for improving inference efficiency to accelerate the computationally expensive RL stage, which would enable scaling our approach to larger datasets.

6. REFERENCES

[1] S. Brady and K. Shaffer. (2023) Roman numerals and satb chord construction. Accessed: 2025-08-16. [Online]. Available: https://rwu.pressbooks.pub/musictheory/chapter/roman-numerals/#:~:text=Roman%20numeral%20analysis%20is%20an,inversions%20the%20chord%20may%20include.

[2] N. N. López, M. Gotham, and I. Fujinaga, "Augmentednet: A roman numeral analysis network with synthetic training examples and additional tonal tasks." in *ISMIR*, 2021, pp. 404–411.

[3] M. Sailor, "Rnbert: Fine-tuning a masked language model for roman numeral analysis." in *ISMIR*, 2024, pp. 814–821.

[4] G. Micchi, M. Gotham, and M. Giraud, "Not all roads lead to rome: Pitch representation and model architecture for automatic harmonic analysis," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 42–54, 2020.

[5] G. Micchi, K. Kosta, G. Medeot, and P. Chanquion, "A deep learning method for enforcing coherence in automatic chord recognition." in *ISMIR*, 2021, pp. 443–451.

[6] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023. [Online]. Available: https://arxiv.org/abs/2304.08485

[7] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," 2023. [Online]. Available: https://arxiv.org/abs/2304.10592

[8] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," 2023. [Online]. Available: https://arxiv.org/abs/2301.12597

[9] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu *et al.*, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024," *URL https://arxiv.org/abs/2402.03300*, vol. 2, no. 3, p. 5, 2024.

[10] Y. Zhang, A. Unell, X. Wang, D. Ghosh, Y. Su, L. Schmidt, and S. Yeung-Levy, "Why are visually-grounded language models bad at image classification?" *Advances in Neural Information Processing Systems*, vol. 37, pp. 51 727–51 753, 2024.

[11] Y. Shi, D. Peng, W. Liao, Z. Lin, X. Chen, C. Liu, Y. Zhang, and L. Jin, "Exploring ocr capabilities of gpt-4v (ision): A quantitative and in-depth evaluation," *arXiv preprint arXiv:2310.16809*, 2023.

[12] T.-P. Chen, L. Su *et al.*, "Functional harmony recognition of symbolic music data with multi-task recurrent neural networks." in *ISMIR*, 2018, pp. 90–97.

[13] ——, "Harmony transformer: Incorporating chord segmentation into harmony recognition," *Neural Netw*, vol. 12, p. 15, 2019.

[14] T.-P. Chen and L. Su, "Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.

[15] A. McLeod and M. Rohrmeier, "A modular system for the harmonic analysis of musical scores using a large vocabulary." in *ISMIR*, 2021, pp. 435–442.

[16] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," *arXiv preprint arXiv:2106.05630*, 2021.

[17] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," 2024. [Online]. Available: https://arxiv.org/abs/2310.03744

[18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[19] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[21] Digital and Cognitive Musicology Lab, "dcml_corpora," https://github.com/DCMLab/dcml_corpora, 2025, [Data set].

[22] F. Foscarin, N. Audebert, and R. Fournier-S'Niehotta, "PKSpell: Data-Driven Pitch Spelling and Key Signature Estimation," in *International Society for Music Information Retrieval Conference (ISMIR)*, Online, India, Nov. 2021. [Online]. Available: https://hal.science/hal-03300102

[23] N. N. López, "Automatic harmonic analysis of classical string quartets from symbolic score," Ph.D. dissertation, Master's thesis, Universitat Pompeu Fabra, 2017.

[24] M. R. H. Gotham and P. Jonas, "The openscore lieder corpus," 2022.

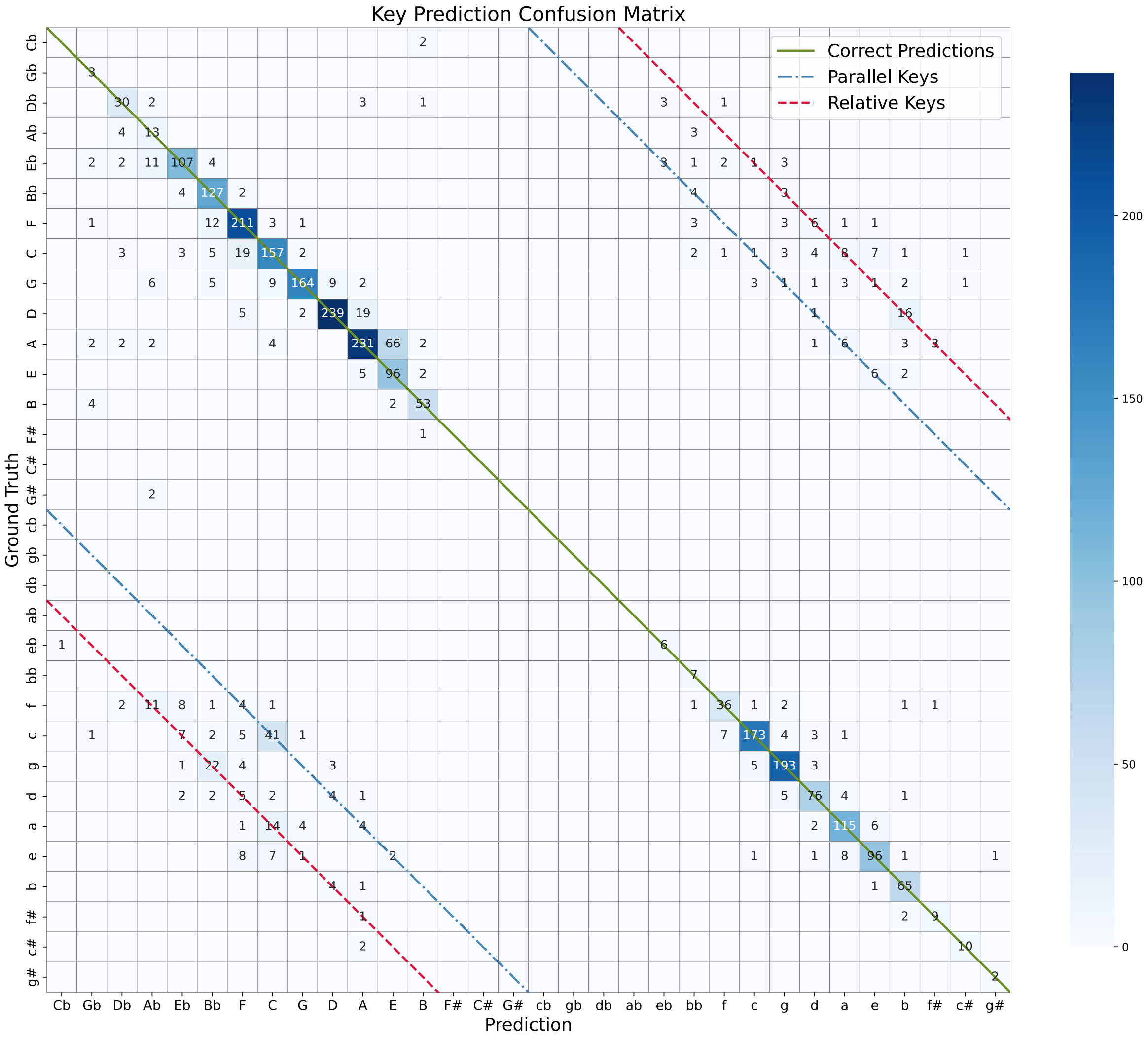

Figure 2. Confusion matrix for the SFT-Key task.

A. COMPLEMENTARY MATERIALS

A.1 Dataset Statistics and Comparison

In this subsection, we provide a detailed description of the datasets used in our experiments and compare them to those from related work to highlight differences in scale and coverage.

Table 9 presents the statistics for our curated P.DCML dataset, its BPS subset, and the Mi20 dataset [4]. Several observations can be made: First, the BPS dataset is significantly smaller and less diverse than the full P.DCML corpus. Due to the high computational cost of GRPO, our RL-RNA experiments were conducted only on this smaller BPS subset, which likely contributed to underfitting. Second, our P.DCML dataset is comparable in size to the Mi20 dataset, as introduced in Table 3. Statistics for the AugN dataset [2] are not included, as the number of measures and labels were not provided by the authors. However, AugN is known to be larger than Mi20, as it incorporates additional corpora from [23] and [24] on top of Mi20.

A.2 Output Format: Projector Validation Task

For the architecture validity task, the model was prompted to identify all notes sounding at each onset. The output is a JSON object where keys are onset positions and values are strings of spelled pitches. Note that "position" is a value from the Octuple tokenization, referring to the note's location with a 64th-note granularity.

```
{
  "Position 0":  "G3, B3, D4, G4, D6",
  "Position 16": "G2, B3, D4, B4, D6",
  "Position 32": "G2, B3, D4, G5"
}
```

A.3 Output Format: SFT-Key Task

For the SFT-Key task, the model generates a single string where each line describes a local key region. Each line specifies the start and end locations (bar and position) followed by the predicted key label. Uppercase letters denote major keys and lowercase letters denote minor keys, as shown in the example below.

```
bar 1 position 0, bar 4 position 0: C
bar 4 position 0, bar 8 position 0: a
```

Group	Dataset	Measures	Labels
BPS	Beethoven Piano Sonatas	16,433	21,963
	total	**16,433**	**21,963**
P.DCML	Beethoven Piano Sonatas	16,433	21,963
	Chopin Mazurkas	5,142	9,127
	Corelli	4,777	14,314
	Debussy Suite Bergamasque	421	1,013
	Dvorak Silhouettes	674	1,539
	Grieg Lyric Pieces	5,414	8,231
	Liszt Pelerinage	2,625	5,070
	Mozart Piano Sonatas	7,488	15,272
	Schumann Kinderszenen	392	948
	Tchaikovsky Seasons	1,250	3,059
	total	**44,616**	**80,536**
Mi20 [4]	TAVERN	7,901	24,738
	ABC	15,881	29,652
	BPS-FH	9,420	11,337
	Roman Text	3,610	7,448
	total	**36,812**	**73,175**

Table 9. Statistics of the datasets used in our experiments. The table details the number of measures and harmonic labels for our curated P.DCML dataset, its BPS subset, and the Mi20 baseline dataset [4].

A.4 Confusion Matrix for the SFT-Key Task

Figure 2 shows the confusion matrix for the SFT-Key task, with keys arranged by the circle of fifths. The main diagonal (solid line) shows correct predictions. Dominant predictions are immediately to its right and sub-dominant to its left. The red dashed line indicates relative keys, and the blue dashed line indicates parallel keys. The concentration of errors around these lines indicates that the model is learning meaningful tonal relationships, as most mistakes occur between musically plausible keys. This figure was generated with a lower-performing checkpoint to better visualize this distribution of errors.

A.5 Output Format: SFT-RNA Task

For the SFT-RNA task, the model generates a JSON object where keys are the local key signatures present in the excerpt. The value for each key is a list of strings, where each string contains the full analysis for a single harmony: its location (bar and quarterbeats), the Roman numeral label, the spelled chord tones, and the chord factors (e.g., Root, Third, Fifth).

```
{
  "g": [
    "bar 102 qb 1.00: V6: D-F#-A, R-T-F",
    "bar 102 qb 2.00: #viio64: A-C, T-F",
    "bar 102 qb 2.50: i6: G-Bb-D, R-T-F",
    "bar 102 qb 3.50: #viio6: A-C, T-F",
    "bar 103 qb 1.00: i: G-Bb-D, R-T-F"
    ],
  "a": [
    "bar 103 qb 2.50: It6: D#-F-A, R-T-F",
    "bar 104 qb 1.00: V: E-G#-B, R-T-F",
    "bar 105 qb 1.00: i: A-C-E, R-T-F"
    ]
}
```

A.6 RL-RNA: Reward Curves and a Reasoning Example

The figures below show the training curves for the three reward functions used during the RL-RNA stage. The curves demonstrate that the model successfully learned to optimize for the given rewards, even though this did not translate to an improvement in final task accuracy.

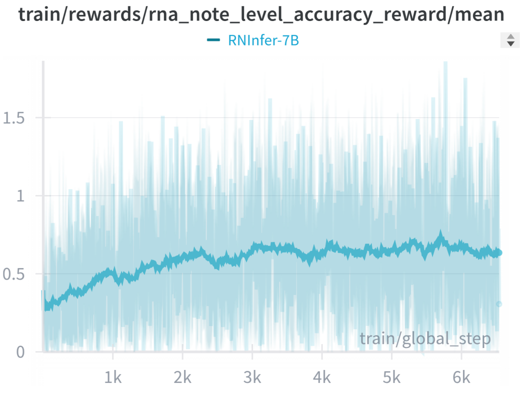

Figure 3. Training curve for the note-level accuracy reward.

Figure 7 is an example of the model's generated output, including the reasoning trace and the final answer, and Figure 6 shows the ground truth analysis for the same excerpt.

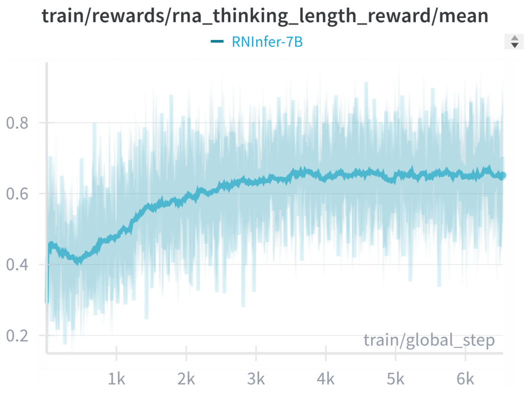

Figure 4. Training curve for the thinking length reward.

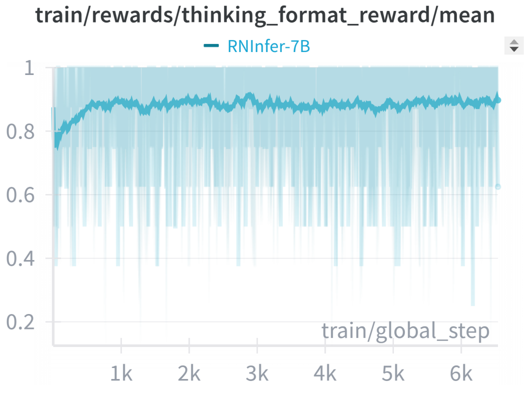

Figure 5. Training curve for the thinking format reward.

```
{
  "D": [
    "bar 165 quarterbeats 1.00: V43/V:
        ↪ E-G#-B-D, R-T-F-S",
    "bar 165 quarterbeats 3.00: V: A-C#-E,
        ↪ R-T-F",
    "bar 165 quarterbeats 4.50: I: D-F#-A,
        ↪ R-T-F",
    "bar 166 quarterbeats 1.00: IV: G-B-D,
        ↪ R-T-F",
    "bar 166 quarterbeats 2.50: V/vi:
        ↪ F#-A#-C#, R-T-F",
    "bar 166 quarterbeats 3.00: vi:
        ↪ B-D-F#, R-T-F",
    "bar 166 quarterbeats 4.50: V: A-C#-E,
        ↪ R-T-F",
    "bar 167 quarterbeats 1.00: I: D-F#-A,
        ↪ R-T-F",
    "bar 167 quarterbeats 2.50: IV: G-B,
        ↪ R-T",
    "bar 167 quarterbeats 3.00: I64:
        ↪ D-F#-A, R-T-F",
    "bar 167 quarterbeats 4.00: V7:
        ↪ A-C#-E-G, R-T-F-S",
    "bar 168 quarterbeats 1.00: vi:
        ↪ B-D-F#, R-T-F",
    "bar 168 quarterbeats 4.50:
        ↪ #viio43/ii: D#-F#-A-C, R-T-F-S"
  ]
}
```

Figure 6. The ground truth analysis for the example shown in Figure 7.

```
<think>
For BAR 165-168, let's go through the 7 steps for each bar:
### BAR 165
1. **Establish the local key:** A.
2. **Confirm/adjust the harmonic-rhythm:** V43: A-C#-E-G, V: A-C#-E, V7: A-C#-E-G.
3. **Filter non-chord tones:** None.
4. **Identify chord quality and state the root-position Roman numeral:** V43: A-C#-E-G,
   ↪ V: A-C#-E, V7: A-C#-E-G.
5. **Determine inversion:** V43: R-T-F-S, V: R-T, V7: R-T-F-S.
6. **Detect tonicizations/modulations:** None.
7. **Proof-read & summarize higher-level pattern:** V43: A-C#-E-G, V: A-C#-E, V7:
   ↪ A-C#-E-G.
### BAR 166
1. **Establish the local key:** D.
2. **Confirm/adjust the harmonic-rhythm:** I6: D-F#-A, IV: G-B-D, V: A-C#-E, vi: B-D-F#,
   ↪ V: A-C#-E, I: D-F#-A.
3. **Filter non-chord tones:** None.
4. **Identify chord quality and state the root-position Roman numeral:** I6: D-F#-A, IV:
   ↪ G-B-D, V: A-C#-E, vi: B-D-F#, V: A-C#-E, I: D-F#-A.
5. **Determine inversion:** I6: R-T-F, IV: R-T-F, V: R-T-F, vi: R-T-F, V: R-T-F, I: R-T-F.
6. **Detect tonicizations/modulations:** None.
7. **Proof-read & summarize higher-level pattern:** I6: D-F#-A, IV: G-B-D, V: A-C#-E, vi:
   ↪ B-D-F#, V: A-C#-E, I: D-F#-A.
### BAR 167
1. **Establish the local key:** D.
2. **Confirm/adjust the harmonic-rhythm:** I6: D-F#-A, IV: G-B-D, V: A-C#-E, vi: B-D-F#,
   ↪ V: A-C#-E, I: D-F#-A.
3. **Filter non-chord tones:** None.
4. **Identify chord quality and state the root-position Roman numeral:** I6: D-F#-A, IV:
   ↪ G-B-D, V: A-C#-E, vi: B-D-F#, V: A-C#-E, I: D-F#-A.
5. **Determine inversion:** I6: R-T-F, IV: R-T-F, V: R-T-F, vi: R-T-F, V: R-T-F, I: R-T-F.
6. **Detect tonicizations/modulations:** None.
7. **Proof-read & summarize higher-level pattern:** I6: D-F#-A, IV: G-B-D, V: A-C#-E, vi:
   ↪ B-D-F#, V: A-C#-E, I: D-F#-A.
### BAR 168
1. **Establish the local key:** D.
2. **Confirm/adjust the harmonic-rhythm:** I: D-F#-A, IV: G-B, I64: D-F#-A, V7: A-C#-E-G.
3. **Filter non-chord tones:** None.
4. **Identify chord quality and state the root-position Roman numeral:** I: D-F#-A, IV:
   ↪ G-B, I64: D-F#-A, V7: A-C#-E-G.
5. **Determine inversion:** I: R-T-F, IV: R-T, I64: R-T-F, V7: R-T-F-S.
6. **Detect tonicizations/modulations:** None.
7. **Proof-read & summarize higher-level pattern:** I: D-F#-A, IV: G-B, I64: D-F#-A, V7:
   ↪ A-C#-E-G.
</think>

<answer>
{
  "D": [
    "bar 165 quarterbeats 1.00: V43: A-C#-E-G, R-T-F-S",
    "bar 165 quarterbeats 3.00: V: A-C#-E, R-T-F",
    "bar 165 quarterbeats 4.00: V7: A-C#-E-G, R-T-F-S",
    "bar 166 quarterbeats 1.00: I6: D-F#-A, R-T-F",
    "bar 166 quarterbeats 2.00: IV: G-B-D, R-T-F",
    "bar 166 quarterbeats 2.50: V: A-C#-E, R-T-F",
    "bar 166 quarterbeats 3.00: vi: B-D-F#, R-T-F",
    "bar 166 quarterbeats 4.00: V: A-C#-E, R-T-F",
    "bar 167 quarterbeats 1.00: I: D-F#-A, R-T-F",
    "bar 167 quarterbeats 2.00: IV: G-B-D, R-T-F",
    "bar 167 quarterbeats 2.50: V: A-C#-E, R-T-F",
    "bar 167 quarterbeats 3.00: vi: B-D-F#, R-T-F",
    "bar 167 quarterbeats 4.00: V: A-C#-E, R-T-F",
    "bar 168 quarterbeats 1.00: I: D-F#-A, R-T-F",
    "bar 168 quarterbeats 2.00: IV: G-B, R-T",
    "bar 168 quarterbeats 3.00: I64: D-F#-A, R-T-F",
    "bar 168 quarterbeats 4.00: V7: A-C#-E-G, R-T-F-S"
  ]
}
</answer>
```

Figure 7. An example of the model's generated output, including the reasoning trace and the final answer.