Learning to Compose Symbolic Jazz

Anonymous submission

Abstract

How should a language model acquire *compositional* skills: by context or by consequence? We contrast prompt evolution with reinforcement learning for jazz composition with the `pretty_midi` symbolic engine. Symbolic jazz is a domain where harmonic conventions can be verified programmatically, yet true skill demands knowing when to violate them. Both methods adapt the same language model and symbolic reward schema to generate 4-bar ensemble pieces (sax, bass, piano, drums). They differ, however, in how reward signal propagates. Prompt evolution refines the Composer system prompt via iterative self-reflection whereas reinforcement learning shapes weights through policy gradient descent, without changes to the original prompt. Our framework clarifies conceptual differences between language-mediated control and gradient-based adaptation to study the fundamental capabilities of learning creative generation under verifiable constraints.

1 Introduction

Jazz occuipes the space between structure and freedom. A pianist knows the ii-V-I progression, but artistry lies in how she voices the chords, when she anticipates the resolution, or whether she substitutes a tritone or plays it straight. The conventions are real—checkable, even—but blindly following them produces mechanical music. Creativity emerges from knowing when structure serves expression and when to break free.

This tension makes jazz an ideal domain for studying how language models acquire creative skills under enabling constraints. Unlike open-ended text generation, musical conventions can be verified programmatically: we can check whether chords are voiced correctly, whether rhythms syncopate, whether the bass walks. Yet verification alone does not capture quality. The same chord progression played mechanically sounds lifeless; played with intention, it swings.

Two paradigms have emerged for adapting language models for task-specific learning. Prompt evolution (Agrawal et al. 2025) refines natural-language instructions through self-reflection and selection. This approach has demonstrated strong results on structured tasks, such as arithmetic, sometimes matching gradient-based methods while using far fewer training samples. Reinforcement learning (Sutton and Barto 1998; Wang et al. 2025) updates weights through reward signals, embedding task knowledge into internal representations. Despite theoretical interest in comparing these approaches, no prior work has evaluated them head-to-head on the same creative task with the same metrics.

We implement both in-context evolution and reinforcement learning on the same model (Qwen3-14B-Instruct) to generate 4-bar jazz ensemble compositions in executable Python code using the `pretty_midi` library (Raffel and Ellis 2014). Our primary contributions are threefold:

(1) **Comparative training dynamics:** Our work is the first direct comparison of prompt evolution vs. RL for creative generation. Prior work evaluates methods in isolation; we offer a comparative look into the tradeoffs in training dynamics, evaluated head-to-head on the same task with identical model, metrics, and random seeds.

(2) **Verifiable reward infrastructure:** We avoid learned reward models entirely, using only symbolic MIDI analysis. Every metric is deterministic and re-runnable from the released code and 10,000+ generated MIDI files.

(3) **Methodological blueprint:** Our framework generalizes to any domain where conventions are programmatically checkable but true competence requires knowing when to violate them: code synthesis with unit tests, protein design with folding constraints, game level generation with playability checks.

2 Background

LLMs for Music Music Transformer (Huang et al. 2019) demonstrated self-attention can capture long-term structure. Music-specialized LLMs like ChatMusician (Yuan et al. 2024) achieve high format correctness through pretraining on symbolic music. Even GPT-4 produces ill-formed compositions when prompted naively; ComposerX addressed this via multi-agent decomposition (Deng et al. 2024). Prompt-controlled generation (Zhu et al. 2024) achieves strong chord accuracy through explicit attribute control. For jazz, datasets remain limited: PiJAMA provides annotated piano performances (Plaja et al. 2024), while the Weimar Jazz Database (Pfleiderer et al. 2017) offers transcriptions used in prior work (Wu and Yang 2020).

Adaptation Methods RL Tuner (Jaques et al. 2017) pioneered using reinforcement learning for music generation, reducing note repetition from 63% to near-zero through reward shaping. RL-Duet (Jiang et al. 2020) applied RL to

interactive accompaniment. Recent work highlights reward hacking risks in music generation (Corcoran and Matsubara 2024). Wang et al. (Wang et al. 2025) demonstrate RLVR for math reasoning with verifiable correctness checks; we extend this to music via deterministic symbolic metrics.

Alternatively, prompt optimization adapts LLMs without weight updates. GEPA (Genetic Prompt Evolution) maintains a population of prompts, uses LLM self-reflection to propose edits, and selects via Pareto fitness (Agrawal et al. 2025). Agrawal et al. report GEPA outperforming traditional RL fine-tuning on multiple tasks (summarization, code generation, etc.), beating a tuned GRPO implementation while using orders of magnitude fewer queries (Agrawal et al. 2025).

3 Methods

Our constructed task requires generating 4-bar jazz ensemble (sax, bass, piano, drums) compositions. The LLM generates executable Python code that constructs `pretty_midi.PrettyMIDI` objects directly. Both approaches to learning share the same evaluation using deterministic, re-runnable checkers that compute rewards from symbolic features (detailed in §3.3 and Appendix A).

3.1 RLVR: Weight Updates via Policy Gradient

Training uses **group-relative policy optimization (GRPO)** (Hilton et al. 2025) with $G = 32$ rollouts per step for 100 steps: for each prompt x we sample a group $\{y_j\}_{j=1}^{G} \sim \pi_\theta(\cdot \mid x)$, compute rewards $r_j = r(x, y_j)$, and form group-normalized advantages

$$A_j = \frac{r_j - \mu_G}{\sigma_G + \varepsilon}, \quad \mu_G = \tfrac{1}{G}\sum_{g=1}^{G} r_g, \;\; \sigma_G^2 = \tfrac{1}{G}\sum_{g=1}^{G}(r_g - \mu_G)^2 \tag{1}$$

which reduce variance without requiring a value network. Advantages drive weight updates (learning rate 5×10^{-6}) that shape π_θ toward higher-reward compositions.

3.2 GEPA: Prompt Evolution via Pareto Selection

We optimize the prompt via **Genetic-Pareto Prompt Evolution (GEPA)** with population size 32 for 100 generations. Each generation t maintains a population $\mathcal{P}_t$ of prompts. For each prompt p, we sample compositions and evaluate a 4-objective score vector:

$$\mathbf{s}(p) = \big(z_{\text{sax}}(p), z_{\text{bass}}(p), z_{\text{piano}}(p), z_{\text{drums}}(p)\big) \tag{2}$$

where $z_{\text{instr}}(p) = \mathbb{E}_{y\sim p}[z_{\text{instr}}(y)]$ averages per-instrument z-scores over sampled compositions. The next generation is formed via Pareto selection and mutation:

$$\begin{aligned} \mathcal{S}_t &= \{p \in \mathcal{P}_t : \nexists p' \in \mathcal{P}_t, \mathbf{s}(p') \succ \mathbf{s}(p)\}, \\ \mathcal{P}_{t+1} &= \textstyle\bigcup_{p\in\mathcal{S}_t^{(\alpha)}} \mu(p, \rho_c, n_p) \end{aligned} \tag{3}$$

where $\mathcal{S}_t^{(\alpha)}$ denotes the top $\alpha = 0.5$ fraction of $\mathcal{S}_t$ via diversity-preserving sorting, and μ applies LLM self-reflection to propose edits with crossover probability $\rho_c = 0.3$ over $n_p = 2$ parents. This evolves the prompt to generate higher-quality compositions without updating model weights.

3.3 Shared Reward Function

Both methods use the same deterministic reward function computed from symbolic MIDI analysis. Features are extracted per instrument: sax (melodic entropy, syncopation, blue note ratio, rhythm 2-grams, peak timing), bass (duration variety, velocity variance, non-chromatic ratio), piano (unique chords, average chord duration, cluster avoidance, out-of-key ratio, chord duration variety), drums (voice independence, density arc, ghost note ratio, kick sparseness). Invalid MIDI receives $z_{\text{instr}} = 0$ for all instruments. Structural constraint violations (e.g., lack of rhythmic variety or rests) trigger gating penalties that scale instrument z-scores. Complete feature definitions and gating rules are provided in Appendix A.

Each instrument contributes a z-score aggregating standardized features:

$$z_{\text{instr}}(y) = \sum_{i=1}^{n_{\text{instr}}} [\![z_i]\!], \quad z_i = \frac{f_i(y) - \mu_i}{\sigma_i}, \tag{4}$$

where $[\![\cdot]\!]$ denotes clamping to ± 3 standard deviations to avoid reward hacking on outlier features, and $n_{\text{instr}} \in \{3, 4, 5\}$ features per instrument are extracted symbolically and standardized via reference statistics (μ_i, σ_i). The base reward applies temperature-scaled sigmoid ($\tau = 5$) to the total z-score:

$$r_{\text{base}}(y) = \frac{1}{1 + \exp(-z_{\text{total}}(y)/\tau)}, \quad z_{\text{total}} = \sum_{\text{instr}} z_{\text{instr}}(y). \tag{5}$$

An ensemble multiplier $m_{\text{ens}}(y) \in [1.0, 1.1]$ rewards sax-piano interaction through rhythmic independence and note overlap. The final reward combines base and ensemble multiplicatively:

$$r(x, y) = r_{\text{base}}(y) \cdot m_{\text{ens}}(y). \tag{6}$$

All computations are deterministic, symbolic, and re-runnable. Feature extractors are fixed.

4 Experimental Setup

4.1 Models & Inference Settings

Composer: OpenPipe/Qwen3-14B-Instruct via OpenPipe ART backend, temperature 0.75, max tokens 12000. **RLVR:** Learning rate 5e-6, group size $G = 24$ rollouts/step, KL coefficient 0.1, advantage balance 0.3 (GRPO). **GEPA:** Population size 24, crossover probability 0.3, Pareto selection via non-dominated sorting. [TODO: Add final hyperparameter values after tuning]

4.2 Training/Evolution Budgets

Our training loop for the RL agent uses a small number of parallel rollouts per iteration (six in our experiments) and applies standard policy gradient updates (advantage actor-critic style). Population size N, rollouts per prompt $M \times G$ with fixed $\{x_k\}$ and seeds.

4.3 Baselines & Oracles

[MISSING: Baseline results - MIPROv2, Reflexion, Self-Refine, naive prompting]

4.4 Common-Random-Numbers Protocol

Using CRN (shared $\{x_k\}$ and seeds) ensures that score differences between prompts are attributable to the prompts, not sampling noise.

5 Results

5.1 Main Tables (Pass Rates & Scalar Scores)

[MISSING: Table 3 - Main quantitative results comparing all methods on test set with statistical significance]

We evaluate both RLVR and GEPA on held-out test inputs, measuring validity rates and per-instrument metric scores. Note that judge scores are used *only for evaluation and comparison*—neither RLVR nor GEPA training uses learned judge feedback. All training rewards are purely verifiable metrics.

[MISSING: Figure 2 - Training/evolution curves showing convergence and sample efficiency]

[MISSING: Figure 3 - Example compositions side-by-side (baseline, GEPA, RLVR)]

5.2 Sample Efficiency & Compute

Prior work by Agrawal et al. (2025) demonstrated that prompt-evolution can achieve up to 20% higher accuracy than RL (GRPO) while using $35\times$ fewer rollouts on diverse tasks (Agrawal et al. 2025). We test whether this sample-efficiency advantage holds for structured music composition.

[MISSING: Table/Figure 4 - Sample efficiency comparison: rollouts, wall-clock time, compute cost]

5.3 Human Preference Study

[MISSING: Human preference evaluation results (optional but recommended)]

6 Discussion

[TODO: Interpret results - which approach performs better? Sample efficiency trade-offs? Failure modes?]

7 Conclusion

In summary, our work sits at the intersection of neuro-symbolic music generation and LLM adaptation techniques. By comparing prompt-based in-context skill acquisition with traditional reinforcement learning, we shed light on which paradigm is more sample-efficient and effective for teaching an AI to "jam" within rule-heavy creative domains. Both approaches leverage the same verifiable reward infrastructure, enabling a controlled comparison on the same task with identical evaluation metrics. This framework establishes a foundation for understanding the trade-offs between language-guided self-learning (GEPA) and weight-based optimization (RLVR) in compositional generation tasks where success can be programmatically verified.

Acknowledgments

[MISSING: Acknowledgments (for camera-ready only)]

References

Agrawal, L. A.; Tan, S.; Soylu, D.; Ziems, N.; Khare, R.; Opsahl-Ong, K.; Singhvi, A.; Shandilya, H.; Ryan, M. J.; Jiang, M.; Potts, C.; Sen, K.; Dimakis, A. G.; Stoica, I.; Klein, D.; Zaharia, M.; and Khattab, O. 2025. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv:2507.19457*.

Corcoran, P.; and Matsubara, T. 2024. GAPT: Generative Adversarial Post-Training for Reward Hacking Mitigation in Music Generation. *arXiv:2411.xxxxx*.

Deng, Q.; Deng, Q.; Wang, R.; Liu, Z.; Deng, Y.; Wang, X.; Song, H.; Xia, Y.; Zhu, Y.; Gai, Z.; et al. 2024. ComposerX: Multi-agent symbolic music composition with LLMs. *arXiv:2404.18081*.

Hilton, B.; Corbitt, K.; Corbitt, D.; Gandhi, S.; William, A.; Kovalenskyi, B.; and Jones, A. 2025. ART: Agent Reinforcement Trainer. https://github.com/openpipe/art.

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Simon, I.; Hawthorne, C.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2019. Music Transformer: Generating music with long-term structure. In *International Conference on Learning Representations*.

Jaques, N.; Gu, S.; Bahdanau, D.; Hernández-Lobato, J. M.; Turner, R. E.; and Eck, D. 2017. Tuning recurrent neural networks with reinforcement learning. In *Proceedings of the International Conference on Learning Representations Workshop Track*.

Jiang, N.; Jin, S.; Duan, Z.; and Zhang, C. 2020. RL-Duet: Online Music Accompaniment Generation Using Deep Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 710–718.

Pfleiderer, M.; Frieler, K.; Abeßer, J.; Zaddach, W.-G.; and Burkhart, B. 2017. The Weimar Jazz Database. Jazzomat Research Project, https://jazzomat.hfm-weimar.de/dbformat/dboverview.html.

Plaja, L.; Durand, E.; Lattner, S.; and Grachten, M. 2024. PiJAMA: Piano Jazz with Automatic MIDI Annotations.

Raffel, C.; and Ellis, D. P. W. 2014. Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 84–93.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Wang, Y.; Yang, Q.; Zeng, Z.; Ren, L.; Liu, L.; Peng, B.; Cheng, H.; He, X.; Wang, K.; Gao, J.; Chen, W.; Wang, S.; Du, S. S.; and Shen, Y. 2025. Reinforcement learning for reasoning in large language models with one training example. *arXiv:2504.20571*.

Wu, S.-L.; and Yang, Y.-H. 2020. The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 142–149.

Yuan, R.; Lin, H.; Wang, Y.; Tian, Z.; Wu, S.; Shen, T.; Zhang, G.; Wu, Y.; Liu, C.; Zhou, Z.; et al. 2024. ChatMusician: Understanding and Generating Music Intrinsically with LLMs. *arXiv:2402.16153*.

Zhu, Z.; Zhang, L.; Dai, T.; Liu, B.; Lin, M.; and Zhao, Z. 2024. SymPAC: Scalable Symbolic Music Generation with Prompt-based Attribute Control. *arXiv:2409.16181*.

Appendix

A Reward Function Specification

A.1 Per-Instrument Feature Definitions

Saxophone (5 features):

- Melodic entropy: Shannon entropy of pitch transitions
- Syncopation: Ratio of off-beat note onsets to total onsets
- Blue note ratio: Proportion of notes in blue note regions (b3, b5, b7)
- Rhythm 2-grams: Diversity of consecutive duration pairs
- Peak timing: Whether melodic peak occurs in optimal position (0.4–0.7 of piece)

Bass (3 features):

- Duration variety: Coefficient of variation in note durations
- Velocity variance: Standard deviation of note velocities
- Non-chromatic ratio: Proportion of intervals > 2 semitones

Piano (5 features):

- Unique chords: Number of distinct simultaneities
- Average chord duration: Mean temporal extent of chords
- Cluster avoidance: Penalty for adjacent semitones in chords
- Out-of-key ratio: Proportion of notes outside key signature
- Chord duration variety: Coefficient of variation in chord durations

Drums (4 features):

- Voice independence: Mutual information between kick/snare/hi-hat patterns
- Density arc: Temporal variation in note density (avoids constant patterns)
- Ghost note ratio: Proportion of low-velocity notes (dynamics)
- Kick sparseness: Proportion of time without kick hits

A.2 Gating Rules

Instrument z-scores are multiplied by penalty factors when structural constraints fail:

Saxophone gates:

- $\times 0.5$ if unique durations < 2
- $\times 0.5$ if no rests detected

Ensemble multiplier: The ensemble reward $m_{\text{ens}}(y)$ is computed from two features measuring sax-piano interaction:

- Rhythmic independence: $1 - \frac{|\text{sax\_onsets} \cap \text{piano\_onsets}|}{|\text{sax\_onsets} \cup \text{piano\_onsets}|}$ (quantized to 16th-note grid)
- Note overlap ratio: Proportion of time both instruments are sounding

These features are z-scored with reference statistics (mean, std), weighted by correlation with human ratings, normalized to $[0, 1]$, and mapped to $[1.0, 1.1]$ via $1 + 0.1 \cdot \text{score}$.