Composer Vector: Style-steering Symbolic Music Generation in a Latent Space

Xunyi Jiang, Xin Xu[*]
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92092
`{xuj003, xinxucs}@ucsd.edu`

Abstract

Symbolic music generation has made significant progress, yet achieving fine-grained and flexible control over composer style remains challenging. Existing training-based methods for composer style conditioning depend on large labeled datasets. Besides, these methods typically support only single-composer generation at a time, limiting their applicability to more creative or blended scenarios. In this work, we propose **Composer Vector**, an inference-time steering method that operates directly in the model's latent space to control composer style without re-training. Through experiments on multiple symbolic music generation models, we show that Composer Vector effectively guides generations toward target composer styles, enabling smooth and interpretable control through a continuous steering coefficient. It also enables seamless fusion of multiple styles within a unified latent-space framework. Overall, our work demonstrates that simple latent-space steering provides a practical and general mechanism for controllable symbolic music generation, enabling more flexible and interactive creative workflows. [2] [3]

1 Introduction

Symbolic music generation can now produce convincing melody, harmonies, and large-scale structures [Lu et al., 2023, Bhandari et al., 2025, Wang et al., 2025]. Despite recent advances, controllable and fine-grained generation remains challenging [Tian et al., 2025]. Among various control dimensions, composer-style generation stands out as a particularly difficult case [Yao and Chen, 2025]. Unlike music attributes such as tempo, key signature, and velocity, a classical composer's style is often data-scarce since each composer typically has only a limited number of available pieces. Apart from data, the style of a composer contains a high-level combination of multiple musical factors, including melody, harmony, texture, and rhythm to make them unique [Zhang et al., 2024]. This combination of data sparsity and representational complexity makes it difficult for traditional supervised or conditioning-based methods to stably control composer style while maintaining model generality. Existing approaches for composer-controlled generation (focusing on classical music in this work) predominantly rely on training-based methods [Zhang et al., 2024, Yao and Chen, 2025], which is computationally intensive. Meanwhile, their controllability is mostly limited to single-composer conditioning. As a result, current text-based controls [Le and Yang, 2024] cannot generalize to more flexible scenarios, where, for example, generating music that blends stylistic traits of Bach and Chopin, or explicitly excluding the influence of Mozart.

[*]Corresponding Author.
[2]Codes are available in `https://github.com/JiangXunyi/Composer-Vector`.
[3]Demo page: `https://jiangxunyi.github.io/composervector.github.io/`

Recent studies show that high-level concepts and behaviors admit differentiable structure in latent spaces. Symbolic music score embeddings support multi-class and multi-label classification [Strepetov and Kovalev, 2025]. SEAL finds that reasoning behavior is separable in LLM latent space [Chen et al., 2025]. Motivated by this evidence, we hypothesize that composer style is differentially represented in symbolic music models and we empirically validate composer structure in Appendix B.

Therefore, building upon the limitations of prior training-based approaches and our validated representational hypothesis, we introduce **Composer Vector**, a latent-space steering method to control the composer style for symbolic music generation. Instead of training, Composer Vector operates directly within the model's internal representation space to capture a stylistic vector corresponding to individual composers. By adding or interpolating these vectors to music language models at inference time, we can continuously modulate the generated music toward or away from specific composer styles, allowing not only style transfer but also style fusion and style suppression. Overall, our contributions are: (i) We introduce Composer Vector, a latent-space steering method that provides fine-grained control over composer style in symbolic music generation. (ii) We show that this approach enables continuous control over composer-style intensity, allowing smooth and interpretable modulation along a steering. (iii) We show that Composer Vector can be linearly combined to achieve multi-style fusion, enabling controllable mixtures of multiple composer styles.

2 Composer Vector

2.1 Style-steering Symbolic Music Generation

We study composer style symbolic music generation with music language models such as NotaGen [Wang et al., 2025] and ChatMusician [Yuan et al., 2024]. Given a textual prompt x (i.e., "%Romantic %Chopin, Frederic %Keyboard" for NotaGen, or "Provide a musical piece that draws inspiration from Chopin's compositions." for ChatMusician), let $r \in \mathcal{C}$ denote the *prompt composer* described in x, and let $c \in \mathcal{C}$ denote the **target composer**, where $\mathcal{C}$ is the set of composers. Our proposed method allows controllable steering toward a *target* composer style c at inference time, without additional training or fine-tuning, as illustrated in Figure 1.

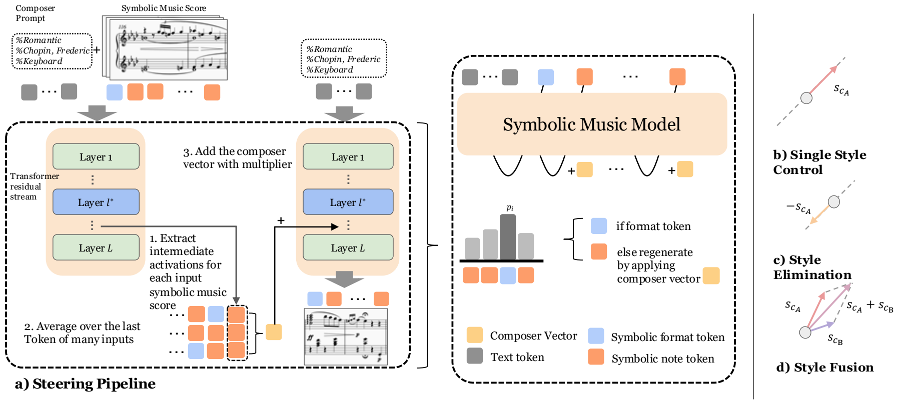


Figure 1: Inference-time control pipeline and style control diagrams

2.2 Method

Composer Vector Construction. To obtain diverse stylistic contexts for each composer c, we first construct a prompt-piece corpus $\mathcal{D}_c = \{x_i^c \oplus p_i^c\}_{i=1}^{N_c}$ for each composer, where x_i^c is a textual prompt describing the composer and p_i^c is a symbolic score with ABC notation format[4]. The symbol $\oplus$ denotes concatenation of sequence. Both components contain composer style context, with the text providing explicit composer cues and the symbolic score encoding intrinsic musical style. Given a transformer-based symbolic music generation model f_θ, the hidden representation at layer l for input

[4]`https://abcnotation.com/`

$x_i^c \oplus p_i^c$ is denoted as $h^{(l)}(x_i^c \oplus p_i^c) \in \mathbb{R}^{T\times d}$, where T is the sequence length and d is the hidden dimension. Following a layer-wise analysis (Appendix D.1), we identify an optimal layer l^* that most strongly encodes composer-specific information, as determined by supervised linear probing and unsupervised clustering metrics.

At this layer, the **Composer Vector** for composer c is defined as the mean embedding over their corpus: $s_c = \frac{1}{N_c}\sum_{i=1}^{N_c} h_T^{(l^\star)}(x_i^c \oplus p_i^c) \in \mathbb{R}^d$, This vector characterizes the latent direction corresponding to composer c. For any subset of composers $\mathcal{C}_k = \{c_1, \ldots, c_k\}$, we compute a composite style vector by linear combination: $s_{\mathcal{C}_k} = \sum_{i=1}^{k} w_{c_i}\, s_{c_i}, w_{c_i} \in \mathbb{R}$. This enables smooth multi-style fusion and interpolation between composer styles. Positive weights amplify the associated styles, whereas negative weights induce contrastive steering, suppressing certain stylistic influences.

Composer Style Steering. Given a target composer c and prompt composer r, let x as the text prompt corresponding to r. We inject the corresponding Composer Vector s_c into the model's residual stream at layer $l^\star$ during generation. Let $\alpha \in \mathbb{R}$ denote the steering coefficient controlling the intensity of stylistic modulation. At each time step t, $y_{<t}$ denotes the output before t of f_θ. We modify the hidden state and rescale to preserve original norm:

$$\widehat{h}_t^{(l^\star)}(x, y_{<t}) = h_t^{(l^\star)}(x, y_{<t}) + \alpha\, s_c, \qquad \widehat{h}_t^{(l^\star)}(x, y_{<t}) \leftarrow \frac{\|h_t^{(l^\star)}(x, y_{<t})\|_2}{\|\widehat{h}_t^{(l^\star)}(x, y_{<t})\|_2}\, \widehat{h}_t^{(l^\star)}(x, y_{<t}).$$

Finally, the model generates the output sequence under steering: $y = f_\theta(x;\, \alpha, s_c)$. This norm-preserving steering allows continuous, directionally interpretable control over composer style.

Format Preservation. To maintain the structural integrity of symbolic notation, we incorporate a simple format preservation rule that limits steering to musically relevant tokens. At each decoding step t, if the predicted token belongs to a predefined set of format tokens (e.g., bar lines, measure separators, or control symbols), we preserve the model's hidden state: $\widehat{h}_t^{(l^\star)}(x, y_{<t}) = h_t^{(l^\star)}(x, y_{<t})$. Otherwise, when the token corresponds to musical content (e.g., notes, durations, dynamics), we apply the composer vector steering as described above. This mechanism ensures that Composer Vector only affects expressive music content while preserving the score's structural validity.

3 Results

We evaluate Composer Vector on NotaGen [Wang et al., 2025] and ChatMusician [Yuan et al., 2024] on CLAP [Wu et al., 2023b] and CLaMP [Wu et al., 2025a] similarity. Our experiments evaluate the effectiveness of the proposed **Composer Vector** for inference-time control of composer style in symbolic music generation. We assess three key aspects: (1) the effectiveness of style steering, (2) the continuity of control strength, and (3) the ability to perform multi-style fusion. Experiments setup details are in Appendix C.2

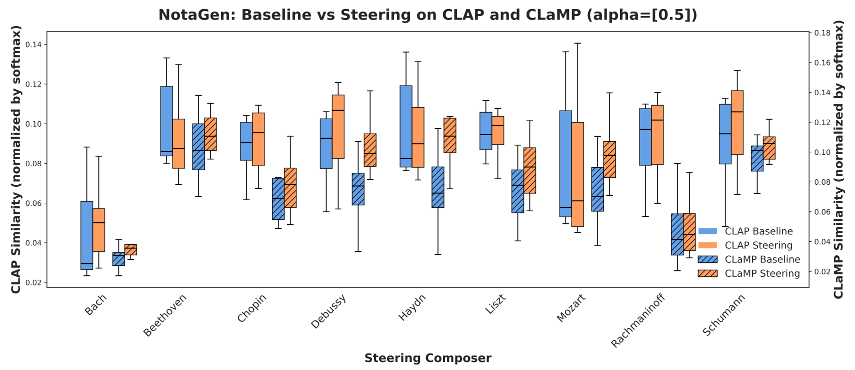


Figure 2: NotaGen: CLAP and CLaMP similarity before and after steering.

Controlling Symbolic Music Generation with Composer Vector. In three objective evaluation metrics over two symbolic music generation models, we validate that the composer vector can guide composer style generation for symbolic music. In similarity-based evaluation, we measure the similarity between generated scores and the target composer style before and after applying the Composer Vector. For both models, similarity to the target composer style is consistently greater after steering than under the no-steering baseline, as shown in Figure 2 and Figure 5. This demonstrates that the Composer Vector shifts generation toward the target composer style. This result confirms that composer vector approach enables controllable composer-style generation across different symbolic generation models. Beyond similarity metrics, we evaluate style control with a supervised composer classifier for music scores. In Table 1, applying Composer Vector consistently increases the classifier's predicted probability for the corresponding composer. Moreover, the vector's control exceeds the effect of the prompt: in ChatMusician, when the **target composer** is **Bach**, several prompt composers, such as Beethoven, Schumann, and Chopin, yield prediction probabilities exceeding 50%. This indicates that the Composer Vector encodes composer-specific information and dominates the prompt conditioning. In conclusion, with similarity and classification metrics, we demonstrate Composer Vector can achieve fine-grained control across different composer styles.

Table 1: Prediction probabilities (%) of **target composer**, where the target composer is the one specified by the Composer Vector. Columns are ordered by prompt era (Classical to Romantic). Headers show the *prompt* composer r (italic) and the **target** composer c (bold). Each value indicates the classifier's predicted probability that the output matches the target composer.

	Bach	*Haydn*	*Haydn*	*Beethoven*	*Beethoven*	*Schubert*	*Schumann*	*Chopin*	*Debussy*
Method	**Mozart**	**Beethoven**	**Liszt**	**Mozart**	**Bach**	**Ravel**	**Bach**	**Bach**	**Beethoven**
NotaGen	5.9	26.3	18.7	4.9	11.7	6.6	11.2	14.9	33.7
+vector	**14.0**	**34.2**	**26.2**	**16.0**	**17.9**	**14.4**	**29.2**	**22.5**	**52.2**
ChatMusician	3.4	4.5	5.0	6.0	25.9	1.2	29.5	23.0	1.4
+vector	**16.4**	**30.1**	**15.1**	**13.2**	**54.3**	**14.6**	**57.0**	**53.4**	**20.8**

Composer Vector can Provide Continuous Style Control. We examine how the classifier probability of the target composer varies with the coefficient α in Figure 7. With Beethoven, Chopin, and Rachmaninoff as targets, we apply their vectors across multiple prompts and plot one regression line per prompt. Across all prompts, α shows a consistent positive correlation with the target composer probability. Relative to the unsteered baseline (dashed lines), the curves lie higher for $\alpha > 0$, showing a clear gain from steering. For instance, in the Rachmaninoff-target case, the probability grows steadily and, by α=0.8, is more than twice the value at α=0.1. Overall, α provides a controllable knob for style-intensity and larger values strengthen target-style features.

Multi-style Fusion through Composer Vector. We study style fusion by linearly combining two composer vectors and measuring classifier probabilities across coefficient pairs in Figure 8. For instance, in the Debussy–Mozart fusion pair, the two regression lines diverge strongly in opposite directions as the coefficient changes. As coefficient of Debussy increases, the corresponding probability increases while the Mozart's decreases, indicating continuous and interpretable interpolation. This opposite–slope pattern recurs across panels with different prompt and steering coefficient. In summary, across all fusion pairs, one composer's probability consistently increases while the other decreases, demonstrating that linear composition of composer vectors enables continuous, interpretable style interpolation.

4 Conclusion

In this work, we proposed **Composer Vector**, an inference-time latent-space method for composer-style generation. It operates in the model's latent space to steer, blend, and fine-grained control styles without fine-tuning. Our analysis showed that composer identity is explicitly encoded in deep layers, providing reliable directions for control. Across two symbolic generation models and multiple metrics, Composer Vector can shift outputs toward a target composer. It supports continuous strength control via a scalar coefficient, and enables linear fusion of multiple composer styles. Overall, these results establish a simple and interpretable mechanism for flexible composer-style control in symbolic music generation.

References

Lukasz Bartoszcze, Sarthak Munshi, Bryan Sukidi, Jennifer Yen, Zejia Yang, David Williams-King, Linh Le, Kosi Asuzu, and Carsten Maple. Representation engineering for large-language models: Survey and research challenges. *ArXiv*, abs/2502.17601, 2025. URL `https://api.semanticscholar.org/CorpusID:276580063`.

Keshav Bhandari, Abhinaba Roy, Kyra Wang, Geeta Puri, Simon Colton, and Dorien Herremans. Text2midi: Generating symbolic music from captions. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 23478–23486. AAAI Press, 2025. doi: 10.1609/AAAI.V39I22.34516. URL `https://doi.org/10.1609/aaai.v39i22.34516`.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. SEAL: steerable reasoning calibration of large language models for free. *CoRR*, abs/2504.07986, 2025. doi: 10.48550/ARXIV.2504.07986. URL `https://doi.org/10.48550/arXiv.2504.07986`.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Francesco Foscarin, Andrew McLeod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. ASAP: a dataset of aligned scores and performances for piano transcription. In Julie Cumming, Jin Ha Lee, Brian McFee, Markus Schedl, Johanna Devaney, Cory McKay, Eva Zangerle, and Timothy de Reuse, editors, *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020*, pages 534–541, 2020. URL `http://archives.ismir.net/ismir2020/paper/000127.pdf`.

Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In Chang Wen Chen, Rita Cucchiara, Xian-Sheng Hua, Guo-Jun Qi, Elisa Ricci, Zhengyou Zhang, and Roger Zimmermann, editors, *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, pages 1180–1188. ACM, 2020. doi: 10.1145/3394171.3413671. URL `https://doi.org/10.1145/3394171.3413671`.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3651–3657. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1356. URL `https://doi.org/10.18653/v1/p19-1356`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL `https://arxiv.org/abs/2001.08361`.

Dinh-Viet-Toan Le and Yi-Hsuan Yang. METEOR: melody-aware texture-controllable symbolic orchestral music generation. *CoRR*, abs/2409.11753, 2024. doi: 10.48550/ARXIV.2409.11753. URL `https://doi.org/10.48550/arXiv.2409.11753`.

Dinh-Viet-Toan Le and Yi-Hsuan Yang. METEOR: melody-aware texture-controllable symbolic music re-orchestration via transformer VAE. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 10126–10134. ijcai.org, 2025. doi: 10.24963/IJCAI.2025/1125. URL `https://doi.org/10.24963/ijcai.2025/1125`.

Dinh-Viet-Toan Le, Louis Bigo, Dorien Herremans, and Mikaela Keller. Natural Language Processing Methods for Symbolic Music Generation and Information Retrieval: A Survey. *ACM Computing Surveys*, 57(7):1–40, July 2025. ISSN 0360-0300, 1557-7341. doi: 10.1145/3714457.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. Controllable text generation for large language models: A survey. *CoRR*, abs/2408.12599, 2024. doi: 10.48550/ARXIV.2408.12599. URL `https://doi.org/10.48550/arXiv.2408.12599`.

Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, and Jiang Bian. Musecoco: Generating symbolic music from text. *CoRR*, abs/2306.00110, 2023. doi: 10.48550/ARXIV.2306.00110. URL https://doi.org/10.48550/arXiv.2306.00110.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *CoRR*, abs/2407.02646, 2024. doi: 10.48550/ARXIV.2407.02646. URL https://doi.org/10.48550/arXiv.2407.02646.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.828. URL https://doi.org/10.18653/v1/2024.acl-long.828.

Petr Strepetov and Dmitrii Kovalev. Symurbench: Benchmark for symbolic music representations. In *Proceedings of the 3rd International Workshop on Multimedia Content Generation and Evaluation: New Methods and Practice*, McGE '25, page 138–146, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720604. doi: 10.1145/3746278.3759392. URL https://doi.org/10.1145/3746278.3759392.

Sida Tian, Can Zhang, Wei Yuan, Wei Tan, and Wenjie Zhu. Xmusic: Towards a generalized and controllable symbolic music generation framework. *IEEE Trans. Multim.*, 27:6857–6871, 2025. doi: 10.1109/TMM.2025.3590912. URL https://doi.org/10.1109/TMM.2025.3590912.

Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hoffman. FIGARO: generating symbolic music with fine-grained artistic control. *CoRR*, abs/2201.10936, 2022. URL https://arxiv.org/abs/2201.10936.

Yashan Wang, Shangda Wu, Jianhuai Hu, Xingjian Du, Yueqi Peng, Yongxin Huang, Shuai Fan, Xiaobing Li, Feng Yu, and Maosong Sun. Notagen: Advancing musicality in symbolic music generation with large language model training paradigms. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 10207–10215. ijcai.org, 2025. doi: 10.24963/IJCAI.2025/1134. URL https://doi.org/10.24963/ijcai.2025/1134.

Shangda Wu, Dingyao Yu, Xu Tan, and Maosong Sun. Clamp: Contrastive language-music pre-training for cross-modal symbolic music information retrieval. In Augusto Sarti, Fabio Antonacci, Mark Sandler, Paolo Bestagini, Simon Dixon, Beici Liang, Gaël Richard, and Johan Pauwels, editors, *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*, pages 157–165, 2023a. doi: 10.5281/ZENODO.10265247. URL https://doi.org/10.5281/zenodo.10265247.

Shangda Wu, Zhancheng Guo, Ruibin Yuan, Junyan Jiang, Seungheon Doh, Gus Xia, Juhan Nam, Xiaobing Li, Feng Yu, and Maosong Sun. Clamp 3: Universal music information retrieval across unaligned modalities and unseen languages. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 2605–2625. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.findings-acl.133/.

Shangda Wu, Yashan Wang, Ruibin Yuan, Zhancheng Guo, Xu Tan, Ge Zhang, Monan Zhou, Jing Chen, Xuefeng Mu, Yuejie Gao, Yuanliang Dong, Jiafeng Liu, Xiaobing Li, Feng Yu, and Maosong Sun. Clamp 2: Multimodal music information retrieval across 101 languages using large language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for*

Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025, pages 435–451. Association for Computational Linguistics, 2025b. doi: 10.18653/V1/2025.FINDINGS-NAACL.27. URL https://doi.org/10.18653/v1/2025.findings-naacl.27.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE, 2023b. doi: 10.1109/ICASSP49357.2023.10095969. URL https://doi.org/10.1109/ICASSP49357.2023.10095969.

Zhengxuan Wu, Qinan Yu, Aryaman Arora, Christopher D. Manning, and Christopher Potts. Improved representation steering for language models. *CoRR*, abs/2505.20809, 2025c. doi: 10.48550/ARXIV.2505.20809. URL https://doi.org/10.48550/arXiv.2505.20809.

Ziwen Xu, Shuxun Wang, Kewei Xu, Haoming Xu, Mengru Wang, Xinle Deng, Yunzhi Yao, Guozhou Zheng, Huajun Chen, and Ningyu Zhang. Easyedit2: An easy-to-use steering framework for editing large language models. *CoRR*, abs/2504.15133, 2025. doi: 10.48550/ARXIV.2504.15133. URL https://doi.org/10.48550/arXiv.2504.15133.

Mingyang Yao and Ke Chen. From generality to mastery: Composer-style symbolic music generation via large-scale pre-training. *CoRR*, abs/2506.17497, 2025. doi: 10.48550/ARXIV.2506.17497. URL https://doi.org/10.48550/arXiv.2506.17497.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23078–23097. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1126/.

Ruibin Yuan, Hanfeng Lin, Yi Wang, Zeyue Tian, Shangda Wu, Tianhao Shen, Ge Zhang, Yuhang Wu, Cong Liu, Ziya Zhou, Liumeng Xue, Ziyang Ma, Qin Liu, Tianyu Zheng, Yizhi Li, Yinghao Ma, Yiming Liang, Xiaowei Chi, Ruibo Liu, Zili Wang, Chenghua Lin, Qifeng Liu, Tao Jiang, Wenhao Huang, Wenhu Chen, Jie Fu, Emmanouil Benetos, Gus Xia, Roger B. Dannenberg, Wei Xue, Shiyin Kang, and Yike Guo. Chatmusician: Understanding and generating music intrinsically with LLM. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6252–6271. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.373. URL https://doi.org/10.18653/v1/2024.findings-acl.373.

Jincheng Zhang, György Fazekas, and Charalampos Saitis. Composer style-specific symbolic music generation using vector quantized discrete diffusion models. In *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2024. doi: 10.1109/MLSP58920.2024.10734713.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Troy Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency. *ArXiv*, abs/2310.01405, 2023. URL https://api.semanticscholar.org/CorpusID:263605618.

A Related work

Symbolic music generation. Symbolic music generation has long been an active area of research [Le et al., 2025]. By representing compositions through abstract notations, symbolic music allows researchers and creators to directly manipulate musical elements, and it enables the analysis and modeling of high-level musical concepts such as harmony, structure, and style [Wang et al., 2025]. To realize these advantages in generation, a variety of symbolic representation systems have been developed, including MIDI [Bhandari et al., 2025, Le and Yang, 2025], REMI [Lu et al., 2023, Huang and Yang, 2020, von Rütte et al., 2022], and ABC notation [Yuan et al., 2024, Wu et al., 2023a, 2025b,a, Wang et al., 2025]. Although differing in granularity and design choices, these representations have each proven effective for symbolic music generation. For example, MIDI-based models can capture detailed performance signals, REMI emphasizes sequential regularity, and ABC notation has shown strong results in modeling melody and structure. Together, these representations make symbolic generation not only suitable for producing melodies but also capable of modeling expressive performance signals, thereby offering both controllability and expressiveness in music generation.

Inference time intervention. With the increasing prevalence of large language models [Kaplan et al., 2020, Yuan et al., 2025] and the growing strength of base models, a line of research has investigated training-free inference-time steering as a way to adapt model behavior without additional fine-tuning ([Wu et al., 2025c, Xu et al., 2025, Rimsky et al., 2024]). Specifically, inference-time steering methods can be broadly categorized into three families. Prompt-based approaches guide generation by designing or augmenting input prompts [Wu et al., 2025c]. Activation-based interventions modify internal activations at inference time to steer latent representations toward desired directions [Zou et al., 2023, Bartoszcze et al., 2025]. Decoding-based techniques instead adjust the sampling or decoding process to bias outputs toward specific attributes [Liang et al., 2024]. Moreover, these steering methods can also support the combination of multiple features, thereby enabling more efficient and fine-grained control of model outputs [Xu et al., 2025].

Mechanistic interpretability. Mechanistic interpretability (MI) seeks to reverse engineer model computations into human-understandable algorithms by decomposing large models into smaller components and elementary operations [Rai et al., 2024]. Following [Olah et al., 2020], MI research is commonly organized around three fundamental objects of study. First, features are human-interpretable properties encoded in model activations (e.g., the token "dog" activating concepts such as "animal" or "pet"), and MI aims to decode these representations. Second, circuits are computational pathways that connect features or transformer components to implement specific behaviors, ranging from toy examples like the induction circuit [Elhage et al., 2021] to generalized notions of information flow. Third, universality investigates whether identified features or circuits recur across different models and tasks, which has critical implications for transferring insights from small-scale studies to large language models [Olah et al., 2020, Rai et al., 2024].

B Composer Styles are Distinguishable in the Latent Space

We hypothesize that composer style, as a high-level musical attribute, is encoded in the latent representations of symbolic music models in a differentiable and structured manner. Therefore, before introducing our steering method, we first examine whether composer styles are inherently encoded within the model's latent representations. Understanding this structure is crucial, as effective steering relies on the existence of disentangled, style-specific directions in the representation space. Following a similar approach to Strepetov and Kovalev [2025], we analyze how composer information is distributed across the layers of NotaGen. For each music piece p in ABC notation [5], we extract its piece-level embedding defined as the hidden state of the final token at position T for each layer l, denoted as $h_T^{(l)}(p)$, for each music piece p. We then visualize these embeddings using t-SNE [Maaten and Hinton, 2008], as shown in Figure 3. Several key observations emerge: (i) Distinct clusters corresponding to different composers are clearly visible in the latent space, indicating that stylistic cues are represented internally. (ii) The separation is weak in lower layers but becomes progressively sharper in deeper ones, with the final layer exhibiting the most pronounced clustering. This aligns

[5] `https://abcnotation.com/`

with prior findings that early layers capture local musical attributes (e.g., pitch transitions), whereas deeper layers encode higher-level structural and stylistic features [Jawahar et al., 2019]. (iii) Certain composers, such as Bach and Liszt, form more compact and well-separated clusters than others, reflecting their stronger stylistic distinctiveness. Overall, these results suggest that composer identity is explicitly manifested in the model's hidden representations—providing a strong foundation for constructing and manipulating Composer Vector in subsequent sections.

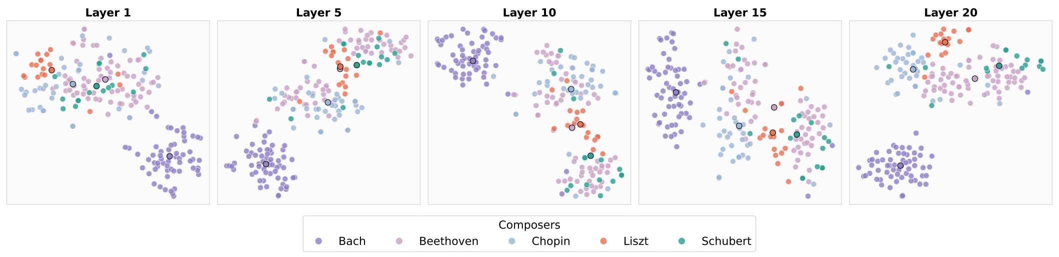


Figure 3: t-SNE visualization of piece-level embeddings across composers.

C Experiments Details

We evaluate the proposed **Composer Vector** method using two symbolic music generation models that operate in ABC notation: **NotaGen** [Wang et al., 2025] and **ChatMusician** [Yuan et al., 2024]. The evaluation dataset is derived from the **ASAP dataset** [Foscarin et al., 2020], which contains symbolic music annotated with composer identities. To ensure consistency across models, we convert all MusicXML files in the dataset into ABC format.

We focus on 11 composers spanning the Baroque, Classical, and Romantic periods. For each composer, we generate symbolic pieces across three instrumental categories: *keyboard*, *chamber*, and *orchestral*. Composer Vector are injected at inference time with steering coefficients $\alpha \in [0, 1]$. This setup enables systematic investigation of both single-composer steering and multi-composer fusion under different control intensities.

The experiments aim to address three research questions (RQs):

- **RQ1:** How effectively can the Composer Vector control the generated composer style?
- **RQ2:** Can the Composer Vector provide *continuous* and interpretable style control?
- **RQ3:** Can the Composer Vector achieve *multi-style fusion* by combining stylistic directions?

C.1 Experiment Design

To answer the above questions, we design three groups of experiments corresponding to RQ1–RQ3.

How effective is the Composer Vector? We assess whether the Composer Vector can guide generation toward the target composer style. For each prompt corresponding to a composer c_p, we apply Composer Vector from all 11 composers $\{c_1, \ldots, c_{11}\}$ during generation. Each model (NotaGen and ChatMusician) produces 11 steered generations per prompt under steering coefficients $\alpha \in \{0.1, 0.3, 0.5, 0.8\}$. Composer Vector are extracted by averaging hidden representations of real ABC samples from the corresponding composer (see Section 2.2). After obtaining the steering vectors, we apply them to the generation process for 11 composers, using steering coefficients of 0.1, 0.3, 0.5, and 0.8.

Can the Composer Vector provide continuous style control? To verify continuous steering effects, we vary the coefficient continuously from 0.0 to 1.0 in 0.05 steps. We perform this test on five composers: Beethoven, Chopin, and Rachmaninoff.

Can the Composer Vector provide fusion of different styles? We test combinations of composers to evaluate fusion effects. Mozart, Debussy, and Rachmaninoff are chosen as steering composers due to their distinct styles. We experiment with different fusion ratios: 0.1, 0.3, 0.5, 0.7, and 0.9.

C.2 Evaluation

Similarity-based Evaluation. We use CLAP [Wu et al., 2023b] and CLaMP [Wu et al., 2025a] scores. For CLAP, we convert generated ABC scores into audio and embed the clips using the CLAP model. For CLaMP, we use the CLAMP3 model to embed symbolic scores directly. After obtaining embeddings from both audio and symbolic domains, we calculate the cosine similarity between the steered and original composer-style generations. An increase in similarity indicates that the steered generation is closer to the target composer's style.

Classification-based Evaluation. We also train a linear classifier to measure the style similarity more precisely. We use CLaMP3 [Wu et al., 2025a] as symbolic music encoder. We train the classifier on MIDI text format on 11 composers. We train the model on a large open source MIDI dataset [6]. We separate the dataset into 70%, 10%, 20%, training, validation, and testing set. The test accuracy is 89.38% over 11 catagories.

Table 2: ASAP dataset summary: 1,067 performances and 236 distinct scores spanning 15 Western classical piano composers.

Composer	MIDI Perf.	Audio Perf.	MIDI/XML Scores
Bach	169	152	59
Balakirev	10	3	1
Beethoven	271	120	57
Brahms	1	0	1
Chopin	289	108	34
Debussy	3	3	2
Glinka	2	2	1
Haydn	44	16	11
Liszt	121	48	16
Mozart	16	5	6
Prokofiev	8	0	1
Rachmaninoff	8	4	4
Ravel	22	0	4
Schubert	62	44	13
Schumann	28	7	10
Scriabin	13	7	2
Total	**1067**	**519**	**222**

D Result Details

D.1 Composer Style Localization

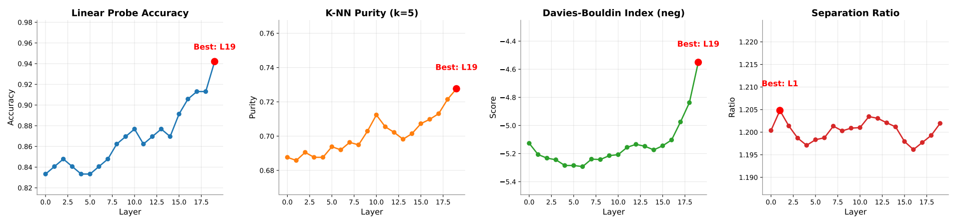
We extract layer-wise hidden states from NotaGen and evaluate their ability to capture composer-specific style information. To this end, we employ a comprehensive evaluation framework that combines both supervised and unsupervised metrics to assess how well different layers organize musical pieces by composer identity.

Supervised evaluation. We employ **linear probe accuracy**, a widely adopted method in mechanism interpretability [Rai et al., 2024], which quantifies how well a simple classifier can recover composer identity from layer-wise embeddings. Specifically, we train a logistic regression classifier with a 75%/25% train–test split (stratified by composer), using the test set accuracy as the evaluation score. This provides a direct measure of the discriminative power of each layer's representation.

Unsupervised evaluation. To complement the supervised probe, we compute six clustering-based metrics that capture different aspects of organization: (i) **k-nearest-neighbor purity (kNN-P, k=5)**, which evaluates local neighborhood consistency by checking the proportion of nearest neighbors with

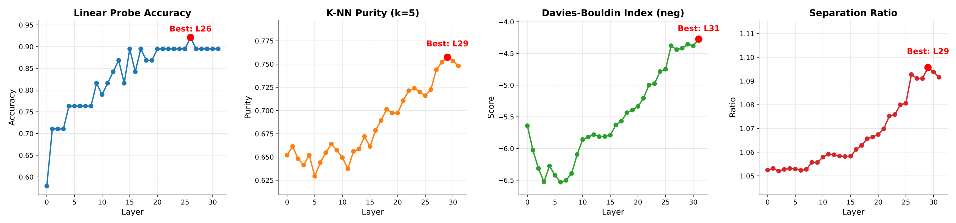
[6] `https://huggingface.co/datasets/drengskapur/midi-classical-music`

NotaGen - Layer-wise Performance Metrics



(a) Layer-wise evaluation of composer style embeddings for NotaGen

ChatMusician - Layer-wise Performance Metrics



(b) Layer-wise evaluation of composer style embeddings for ChatMusician

Figure 4: Comparison of layer-wise composer style localization across different models.

the same composer label (ii) the **Davies–Bouldin index (DBI)**[7], reported in negated form ($-\text{DBI}$) so that larger values indicate better cluster separation; (iii) the **separation ratio (SepR)**, defined as the ratio of inter-cluster to intra-cluster distances, providing a normalized measure of relative separation;

Together, these supervised and unsupervised metrics provide a comprehensive assessment of how well different layers encode composer-specific style.

We select the layer that achieves the highest number of first-place rankings across the evaluation metrics as the final steering layer. In Figure 4a, the linear probe accuracy reaches to 94% at layer 19th, which is the last layer of the NotaGen patch model. As for the ChatMusician, it has the highest K-NN purity (k=5) at layer 29th. These layers are the deeper layers in the model, which is consistent with the previous omposer style analysis. Overall, we choose 19th layer as steering layer in the NotaGen, and 29th layer in the ChatMusician as steering layer.

D.2 Single-style steering

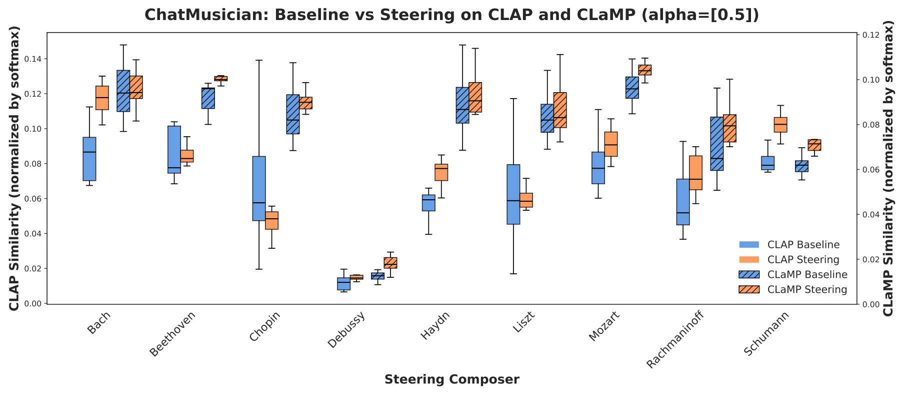
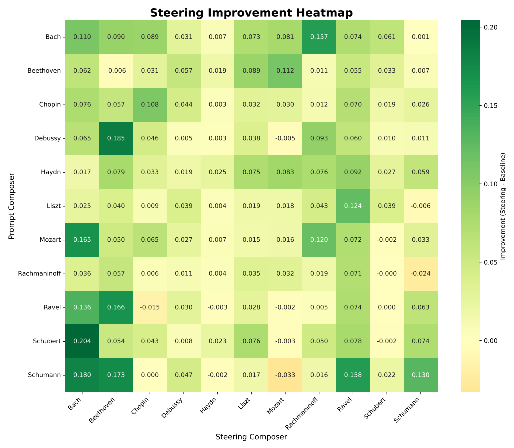


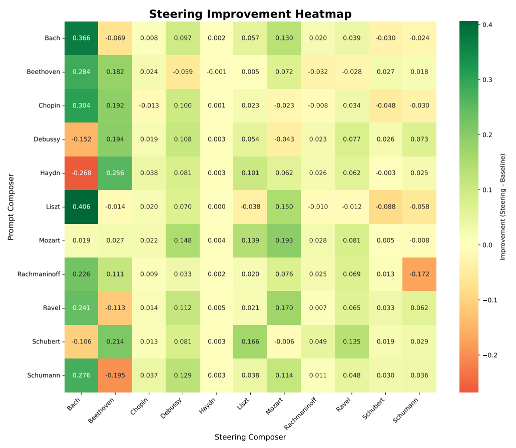
Figure 5: ChatMusician: CLAP and CLaMP similarity before and after steering.

Similarity-based Evaluation.

[7]https://en.wikipedia.org/wiki/Davies-Bouldin_index



(a) NotaGen.



(b) ChatMusician.

Figure 6: Steering improvement heatmaps for (a) NotaGen and (b) ChatMusician.

Prompt (B / S)	Bach	Beethoven	Chopin	Debussy	Haydn	Liszt	Mozart	Rach.	Ravel	Schubert	Schumann
Bach	48.4% / 59.3%	4.3% / 13.3%	3.1% / 12.0%	7.6% / 10.7%	1.6% / 2.3%	9.3% / 16.5%	5.9% / 14.0%	2.5% / 18.2%	2.6% / 10.0%	11.8% / 17.9%	3.0% / 3.1%
Beethoven	11.7% / 17.9%	35.8% / 35.3%	6.4% / 9.5%	4.5% / 10.3%	1.2% / 3.1%	12.3% / 21.2%	4.9% / 16.0%	7.2% / 8.3%	4.7% / 10.2%	4.3% / 7.6%	7.0% / 7.7%
Chopin	14.9% / 22.5%	28.6% / 34.3%	12.8% / 23.6%	4.9% / 9.3%	0.6% / 0.9%	9.9% / 13.0%	4.0% / 6.9%	10.2% / 11.4%	3.3% / 10.3%	1.1% / 3.1%	9.6% / 12.2%
Debussy	18.5% / 25.1%	33.7% / 52.2%	6.2% / 10.9%	7.5% / 7.9%	0.8% / 1.0%	6.1% / 9.9%	7.2% / 6.7%	6.9% / 16.2%	7.6% / 13.6%	0.6% / 1.5%	4.9% / 6.0%
Haydn	16.8% / 18.5%	26.3% / 34.2%	4.3% / 7.6%	5.2% / 7.1%	1.6% / 4.0%	18.7% / 26.2%	10.7% / 19.0%	3.4% / 11.0%	4.7% / 13.8%	4.3% / 7.1%	4.1% / 10.0%
Liszt	19.6% / 22.1%	26.9% / 30.9%	11.0% / 11.9%	4.1% / 8.0%	1.2% / 1.6%	11.0% / 12.8%	2.9% / 4.7%	7.1% / 11.4%	6.6% / 19.0%	0.9% / 4.8%	8.8% / 8.1%
Mozart	11.1% / 27.6%	23.3% / 28.3%	3.9% / 10.4%	5.4% / 8.2%	3.1% / 3.8%	24.5% / 26.1%	11.2% / 12.9%	3.5% / 15.5%	4.6% / 11.9%	4.3% / 4.1%	4.9% / 8.2%
Rach.	16.0% / 19.6%	27.5% / 33.2%	13.8% / 14.4%	5.1% / 6.2%	0.7% / 1.1%	7.6% / 11.1%	1.7% / 5.0%	9.5% / 11.3%	5.3% / 12.4%	2.0% / 2.0%	10.8% / 8.4%
Ravel	16.2% / 29.8%	22.1% / 38.7%	9.7% / 8.2%	7.5% / 10.4%	1.2% / 0.9%	7.3% / 10.1%	10.5% / 10.3%	10.3% / 10.8%	7.3% / 14.7%	1.5% / 1.5%	6.4% / 12.7%
Schubert	14.4% / 34.9%	30.0% / 35.4%	7.9% / 12.2%	5.8% / 6.7%	0.9% / 3.2%	7.8% / 15.3%	5.8% / 5.5%	9.6% / 14.7%	6.6% / 14.4%	2.9% / 2.7%	8.2% / 15.6%
Schumann	11.2% / 29.2%	15.7% / 33.0%	14.9% / 14.9%	5.1% / 9.8%	1.2% / 1.0%	11.0% / 12.7%	9.5% / 6.2%	8.5% / 10.2%	6.8% / 22.7%	1.0% / 3.2%	14.9% / 28.0%

Table 3: All Classifier Prediction Results for NotaGen. Baseline (top, ends with "/") and Steering (bottom). Numbers with green background indicate steering probabilities greater than 10%. "Rach." = Rachmaninoff.

Prompt (B / S)	Bach	Beethoven	Chopin	Debussy	Haydn	Liszt	Mozart	Rach.	Ravel	Schubert	Schumann
Bach	15.1% / 51.7%	31.9% / 25.0%	0.8% / 1.6%	9.1% / 18.8%	0.1% / 0.4%	12.6% / 18.4%	3.4% / 16.4%	4.4% / 6.5%	9.7% / 13.6%	6.1% / 3.1%	6.7% / 4.3%
Beethoven	25.9% / 54.3%	2.5% / 20.7%	0.9% / 3.2%	22.6% / 16.7%	0.4% / 0.4%	7.4% / 7.9%	6.0% / 13.2%	7.4% / 4.2%	19.5% / 16.7%	2.7% / 5.4%	4.6% / 6.4%
Chopin	23.0% / 53.4%	6.2% / 25.4%	3.8% / 2.5%	9.9% / 19.9%	0.4% / 0.4%	10.4% / 12.7%	15.3% / 13.0%	5.1% / 4.3%	10.3% / 13.7%	7.2% / 2.3%	8.5% / 5.5%
Debussy	63.8% / 48.6%	1.4% / 20.8%	1.1% / 3.1%	9.3% / 20.1%	0.1% / 0.4%	1.5% / 6.9%	17.1% / 12.8%	2.1% / 4.3%	2.0% / 9.7%	1.2% / 3.8%	0.5% / 7.8%
Haydn	63.6% / 36.7%	4.5% / 30.1%	1.5% / 5.3%	5.7% / 13.8%	0.1% / 0.4%	5.0% / 15.1%	3.7% / 9.9%	1.8% / 4.4%	8.5% / 14.7%	3.1% / 2.7%	2.8% / 5.3%
Liszt	20.8% / 61.3%	15.5% / 14.1%	0.5% / 2.5%	10.4% / 17.4%	0.2% / 0.3%	9.2% / 5.4%	2.8% / 17.8%	5.8% / 4.8%	12.8% / 11.7%	12.4% / 3.6%	9.7% / 3.9%
Mozart	53.5% / 55.4%	22.3% / 25.0%	0.9% / 3.1%	3.0% / 17.8%	0.1% / 0.5%	4.5% / 18.4%	2.7% / 21.9%	3.2% / 6.1%	3.1% / 11.2%	1.2% / 1.7%	5.6% / 4.8%
Rach.	29.4% / 52.0%	7.3% / 18.5%	1.1% / 1.9%	15.8% / 19.1%	0.2% / 0.3%	7.2% / 9.2%	6.7% / 14.3%	3.2% / 5.8%	4.5% / 11.4%	2.6% / 3.9%	22.0% / 4.8%
Ravel	40.4% / 64.5%	24.0% / 12.6%	0.9% / 2.3%	6.4% / 17.5%	0.1% / 0.6%	3.6% / 5.7%	8.0% / 24.9%	4.1% / 4.8%	4.2% / 10.7%	1.6% / 4.9%	6.8% / 13.0%
Schubert	71.0% / 60.4%	0.6% / 22.0%	1.1% / 2.4%	6.3% / 14.3%	0.1% / 0.4%	1.9% / 18.4%	14.0% / 13.4%	0.7% / 5.6%	1.2% / 14.6%	1.4% / 3.4%	1.8% / 4.7%
Schumann	29.5% / 57.0%	47.7% / 28.1%	0.7% / 4.5%	3.3% / 16.2%	0.2% / 0.5%	2.8% / 6.6%	5.4% / 16.9%	3.5% / 4.6%	3.8% / 8.6%	0.6% / 3.6%	2.5% / 6.1%

Table 4: All Classifier Prediction Results for ChaMusician. Baseline (top, ends with "/") and Steering (bottom). Numbers with green background indicate steering probabilities greater than 10%. "Rach." = Rachmaninoff.

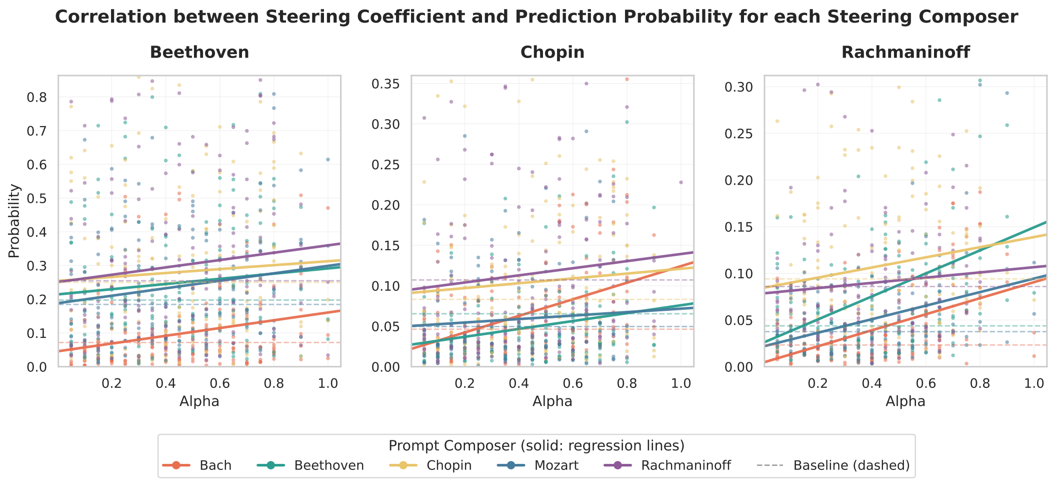


Figure 7: Correlation Between Steering Coefficient and Prediction Probability

Classification results. We also use the naive generation as baseline to compare the prediction probability difference. The Table 3 shows the probability difference for each generated score. For instance, a piece has prompt "Bach" and steered by "Mozart". Then we will compare the Mozart probability versus the Mozart probability of naive "Bach" generation. As shown in the Figure 6. 97.5% of NotaGen probability difference is larger than 0. ChatMusician 84.3% probability difference is larger than 0. The composer vector is more stable on the NotaGen, but in general the steered generation will have higher probability for that steering composer.

D.3 Steering Coefficient Analysis

To further assess the effectiveness of the steering coefficient, we analyze how the classifier-predicted probability of the steering composer varies with the coefficient value α. Figure 7 illustrates the rela-

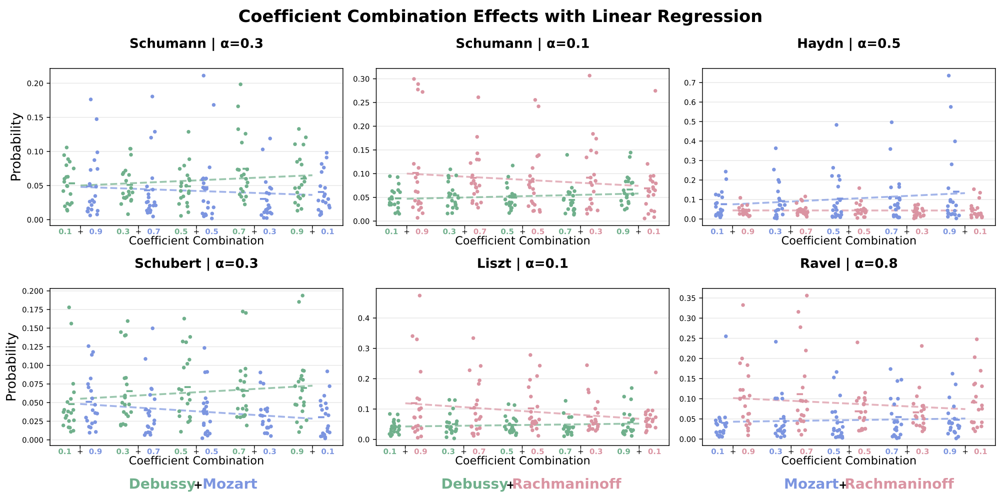


Figure 8: Probability of Steering Composers for Style Fusion

tionship between the steering coefficient and classifier-predicted probability for three representative composers: Beethoven, Chopin, and Rachmaninoff. Each point represents a single steered sample, and the color indicates the prompt composer r used during generation. Each subfigure corresponds to one steering composer, where each point represents an individual steered sample, and the color indicates the prompt composer used during generation. Solid lines denote linear regression fits between α and the classifier-predicted probability, while the dashed line represents the unsteered baseline.

First, we observe a consistent positive correlation between the steering coefficient α and the classifier probability of the corresponding composer. As α increases, the predicted likelihood of the target composer rises across nearly all prompt conditions. For instance, in the Rachmaninoff-steering group, the classifier probability grows steadily from near-baseline levels at α=0.1 to more than double at α=0.8 when base prompts are Bach, Mozart, and Beethoven, demonstrating that stronger steering coefficients effectively enhance stylistic alignment. Second, cross-composer variations highlight meaningful stylistic distinctions among composers. In the Beethoven-steering experiments, samples prompted by Bach yield notably lower classifier probabilities compared to those prompted by Romantic-era composers such as Liszt or Schumann. This pattern suggests that stylistic distance between Bach and Beethoven constrains the model's ability to transfer features through latent steering, whereas stylistically closer composers benefit more from steering amplification.

In summary, these results show that the steering coefficient α provides a controllable and interpretable mechanism for modulating composer-style intensity. Larger coefficients reinforce stylistic features of the target composer, while inter-composer variations in response strength reveal underlying structure in the model's latent style representation.

D.4 Multi-style Fusion through Composer Vector

To further assess the compositional flexibility of the proposed steering method, we investigate style fusion by linearly combining two composer vectors. We focus on three stylistically distinct composers: Mozart, Debussy, and Rachmaninoff, whose contrasting harmonic and textural characteristics allow a clear examination of how linear blending influences stylistic interpolation. This diversity allows us to isolate and contrast the steering effects among stylistically distant composers.

Figure 8 shows the classifier-predicted probabilities for each steering composer across different coefficient combinations (0.1–0.9). Each point represents an individual steered sample, and regression lines capture the trend between coefficient ratios and classifier probabilities.

First, in the Debussy–Mozart fusion pair, the two regression lines diverge strongly in opposite directions as the coefficient changes. The probability of Debussy rises steadily while that of Mozart decreases, forming a clear linear correlation. This pattern indicates that their latent style representations are highly separable, allowing precise control over stylistic balance between impressionistic fluidity and classical clarity. Moreover, even in fusion pairs with weaker global trends, such as Debussy–Rachmaninoff and Mozart–Rachmaninoff in Figure 8, the highest classifier probabilities for individual samples still align with the dominant coefficient in each combination. This pattern suggests that although the overall regression slopes are mild, the model retains localized sensitivity to the steering ratio, capturing the intended stylistic emphasis at the sample level. In other words, the maximum probability values vary systematically with the coefficient ratio, indicating that the model's responses remain well aligned with the intended style proportions.

In summary, across all pairs, one composer's probability consistently increases while the other decreases, demonstrating that linear composition of composer vectors enables continuous, interpretable style interpolation. The degree of slope separation reflects how linearly independent the underlying stylistic manifolds are in the model's latent space, confirming that composer-vector fusion provides a controllable mechanism for generating hybrid musical styles.