Generative Melody Composition with Human-in-the-Loop Bayesian Optimization

Yijun Zhou[1], Yuki Koyama[2], Masataka Goto[2], and Takeo Igarashi[1*]

[1] The University of Tokyo
[2] National Institute of Advanced Industrial Science and Technology (AIST)

Abstract. Deep generative models allow even novice composers to generate various melodies by sampling latent vectors. However, finding the desired melody is challenging since the latent space is unintuitive and high-dimensional. In this work, we present an interactive system that supports generative melody composition with human-in-the-loop *Bayesian optimization* (BO). This system takes a mixed-initiative approach; the system generates candidate melodies to evaluate, and the user evaluates them and provides preferential feedback (i.e., picking the best melody among the candidates) to the system. This process is iteratively performed based on BO techniques until the user finds the desired melody. We conducted a pilot study using our prototype system, suggesting the potential of this approach.

Keywords: Melody composition; deep generative model; Bayesian optimization

1 Introduction

Deep generative models, such as generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational auto-encoders (VAEs) (Kingma & Welling, 2014), have been applied to support creative work, showing great potentials in images, videos, and music (Vondrick et al., 2016; Zhu et al., 2016; Dong et al., 2018). When we focus our scope on the music domain, many research projects have tried to learn latent spaces of melodies (Yang et al., 2017; Fernández & Vico, 2013; Simões et al., 2019; Herremans et al., 2017). By sampling latent vectors from the smooth latent space produced by one of these generative melody models, even novice composers can quickly generate various meaningful melodies (Ghosh et al., 2019; Roberts, Engel, et al., 2018).

However, end-users may face significant challenges when exploring the latent space to find a desired melody, since the latent space is often unintuitive and high-dimensional. Researchers have developed many interaction techniques (e.g., mixing two melodies) to assist users in exploring the latent space to achieve their creative goals in music (Vogl & Knees, 2016; Roberts, Hawthorne, & Simon,

2018). These techniques have shown opportunities to enhance the user experience of music composition.

In this work, we propose a new approach to facilitate generative melody composition for novices, which is based on human-in-the-loop *Bayesian optimization* (BO) techniques (Brochu et al., 2007; Koyama et al., 2017, 2020). Specifically, we present a melody-composition system that employs a human-AI mixed-initiative interaction: the system generates candidate melodies to evaluate, and the user evaluates them and provides preferential feedback (i.e., picking the best melody among the observed candidates) to the system. This process is iteratively performed until the user finds the desired melody. The generation of candidate melodies by the system is based on BO, which tries to make the query to the user as effective in terms of the number of necessary iterations as possible.

We evaluated our approach through both a simulated experiment and a pilot user study, suggesting the potential of this approach. To the best of our knowledge, this work is the first to investigate the potential of human-in-the-loop BO in the context of generative melody composition and AI music creativity.

2   Related Work

2.1   Human-in-the-Loop Bayesian Optimization

BO is a black-box optimization technique (Shahriari et al., 2016). It is especially useful when handling expensive-to-query objective functions since it tries to minimize the number of necessary iterations by selecting the most effective query in each iteration. Thus, it is widely used for, for example, hyperparameter tuning of deep learning models (Akiba et al., 2019).

Human's preference is expensive to query. Thus, a search problem with an objective function defined by human perceptual assessment (e.g., preference) should be solved with a minimal number of queries. This motivates researchers to develop human-in-the-loop BO methods (Brochu et al., 2007; Koyama et al., 2017, 2020; Chong et al., 2019). As preferential assessment needs to be "relative" (e.g., which option is the better?) rather than "absolute" (e.g., how good is this option?) (Brochu et al., 2010; Tsukida & Gupta, 2011), these methods take relative preferential feedback as input, unlike typical BO algorithms. Thus, this approach is also called as *preferential BO* (PBO). The PBO method proposed by Brochu et al. (2007) takes pairwise-comparison results as input, which is the simplest form of preferential feedback. To make the necessary number of queries even smaller, the methods proposed by Koyama et al. (2017, 2020) take subspace-search results as input, where human is iteratively asked to find the best possible option from a one- or two-dimensional search subspaces.

In this work, we apply this PBO[3] approach to generative melody composition. Our system iteratively asks the user to compare multiple melody candidates sampled from a one-dimensional subspace (Koyama et al., 2017) and to select the preferred one. In addition, we allow the user to directly edit the selected one to provide more controllability to the user.

---

[3] In the following sections, we simply call it BO when no confusion can arise.

2.2   Tools for Deep Music Generation

The expanding use of deep generative models encourages researchers to work on interfaces and tools to make the interactive music generation experience more accessible (Huang et al., 2018; Roberts et al., 2019). Vogl et al. (2016) presented an AI-powered software prototype for generating drum pattern variations. Esling et al. (2019) applied a VAE to increase the controllability of sound synthesizers. Roberts et al. (2018) proposed a recurrent VAE, *MusicVAE*, which utilized a hierarchical decoder for improved modeling sequences with long-term structure. Our system uses the pre-trained MusicVAE, considering its smooth latent space. *Magenta.js* (Roberts, Hawthorne, & Simon, 2018) provides developers with APIs to design and deploy web-based interface of generative melody models. Louie et al. (2020) investigated *AI-steering* tools to give users more control over exploring latent spaces of deep generative models, and presented a human-AI co-creation tool. While we share a similar motivation, we focus on investigating the approach of formulating the task as a human-in-the-loop optimization.

3   System Overview

Our system is web-based, and it consists of two main modes: composing mode (Figure 1 (left)) and searching mode (Figure 1 (right)). The searching mode works as a dialog of the system. The front-end is implemented in JavaScript, while the back-end is managed by a Flask (Python) server (Grinberg, 2014). All the musical data represented on the interfaces and transferred between front-end and back-end are on the symbolic MIDI representation (Rothstein, 1996). A video demo of the system can be found at our web source[4].

Fig. 1. System interface. (Left) Composing mode: (A) metronome on/off and BPM; (B) note sheet; (C) add/subtract more bars and play; (D) select two bars and open the searching mode. (Right) Searching mode: (E) current candidates that BO generated; (F) editing widget; (G) back to the initial state, go to the next iteration, finish, or cancel the searching.

---

[4] yijunzhou.xyz/index/generative-melody-composition-with-human-in-the-loop-bo

Composing Mode. The composing mode allows the user to compose melodies by inputting notes. The interface resembles standard composing software interface, providing a piano-roll timeline and essential functions. Vertical axis represents pitch, and horizontal axis represents timeline. Each column represents a 16th note duration, and a set of 16 columns represents a bar. The user can change the length of the melody by adding and removing bars. Each row represents a pitch of piano key. Since we focus on the main melody composition, we limit the pitch range into three octaves, from C3 to B5. The user can draw and delete notes, and change notes' pitches on the note sheet. The composed melody is used as the starting point of the search when entering the searching mode.

Searching Mode. The searching mode is designed to iteratively show multiple candidates generated by BO based on the user's preference. In each iteration, it shows n candidates ($n = 4$ in our case), from which it allows the user to select the one they prefer. After selecting a candidate, they can edit it in the editing widget before going to the next iteration. The user is not allowed to edit the other bars from the composing mode in the editing widget. They can always go back to the initial state of the current iteration. The iteration goes on by showing the user another batch of candidates until the user finds the desired one.

Interaction Procedure. (1) Every time the user gets stuck at composing, they can select a two-bar melody clip that they wish to see variations and then click the "Search" button in the composing mode. (2) After the searching mode pops up, the user will see the original melody clip in the editing widget and n candidates on the top, which are drawn by BO from the latent space[5]; the user can listen to them and select the most favorite one among the candidates and the current melody. (3) The candidate the user selected will be shown in the editing widget, and the user can tweak details if they want. (4) After editing the candidate, the user can click the "Next Iteration" button to go to the next iteration, where they will see a new batch of candidates generated by BO based on their preferential input; the edited melody in the last iteration stays at the editing widget. (5) The user can go back to the initial state of the current iteration by clicking "Back". (6) Once the user finds the desired one, they can click the "Finish" button to finish the iteration, and then the searching result will be applied to their melody in the composing mode; Or, they can also click the "Cancel" button to discard all the iterations and go back.

4   Algorithms and Implementations

4.1   Generative Model of Melodies

Our implementation uses the pre-trained MusicVAE model (Roberts, Engel, et al., 2018) for melody generation, whose latent space is 512-dimensional. Before we

---

[5] At the first iteration, we generate candidates by random sampling since BO does not have data yet.

feed 2-bar melody to MusicVAE, we covert it from MIDI format to *NoteSequence* representation (Magenta, 2013). As it is a VAE architecture, it can encode a melody (which we denote by $\mathbf{x} \in \mathcal{X}$) to a latent vector (which we denote by $\mathbf{z} \in \mathcal{Z}$) as well as decode a latent vector to a melody. We denote the encoder by $E : \mathcal{X} \to \mathcal{Z}$ and the decoder by $D : \mathcal{Z} \to \mathcal{X}$. That is, given a latent vector $\mathbf{z}$, the decoder maps it to a melody $\mathbf{x}$; similarly, given a melody $\mathbf{x}$, the encoder maps it to a latent vector $\mathbf{z}$. Note that any other deep generative models can be used as long as they support both encoding and decoding.

4.2   Dimension Reduction of Latent Space

It is known that BO does not work efficiently with very high-dimensional problems (e.g., more than 20 dimensions) (Wang et al., 2016; Chiu et al., 2020), and thus we need to apply dimensionality reduction to obtain a reasonably lower-dimensional latent space. We tested several techniques for this purpose, including principle component analysis, auto-encoders, and VAEs. We empirically found that VAEs worked well in terms of minimizing reconstruction errors and generating convincing and various melodies. We also tested several different dimensionalities and found that reducing into a 4-dimensional latent space would be a reasonable choice. Thus, we decided to train a VAE whose latent space is 4-dimensional. Note that this choice is consistent with the previous work by Dinculescu et al. (2019). For the training, we selected ten songs (resulting in 284 two-bar melody fragments) from the MusicVAE training dataset, the genres of which cover pop, rock, country, and R&B.

In the following sections, we will still denote the (reduced) latent vector by $\mathbf{z}$ and the (reduced) latent space by $\mathcal{Z}$; but notice that the dimension is 4, not 512. Also, we will suppose that the encoder E and decoder D include the dimensionality conversion process.

Note that this dimensionality reduction would somewhat sacrifice the original expressiveness of the generative model; we consider this is a limitation in our approach. However, we found that the reduced model can still produce various melodies and is sufficiently inspiring and useful for helping novices find the desired melodies.

4.3   Problem Formulation

Following the previous work (Brochu et al., 2007; Koyama et al., 2017, 2020), we formulate the search of the user's desired melody as an optimization problem:

$$\mathbf{z}^* = \arg \max_{\mathbf{z} \in \mathcal{Z}} f(D(\mathbf{z})), \tag{1}$$

where the function $f : \mathcal{X} \to \mathbb{R}$ represents the user's subjective preference over melodies, or how good the melody is (thus it is called a "goodness function" (Koyama et al., 2017)), and $\mathbf{z}^*$ is the latent vector that generates the most satisfying melody. Note that we do not know the landscape of f, which is the

challenge of this problem. The goal here is to solve this optimization problem by iteratively presenting candidate solutions to the user and asking them to choose preferred ones. In the current implementation, we assume a boundary $\mathcal{Z} = [-2.5, 2.5]^4$ to make the search problem tractable.

We generate candidate melodies by a preferential variant of BO (Brochu et al., 2007; Koyama et al., 2017, 2020). In short, new candidates are determined based on all the user's feedback obtained so far so that every new candidate is as effective as possible (subsection 4.4). As in the previous work, we assume a Gaussian process (GP) prior on the goodness function. We jointly estimate the underlying goodness values and the hyperparameters in GP by *maximum a posteriori* (MAP) estimation; refer to (Koyama et al., 2017) for details.

The algorithm takes the user's relative assessment as input. In our system, the set of candidates involved at the i-th iteration consists of (1) $\mathbf{x}_i^+$, which is the current-best candidate (i.e., the user's selection at the last iteration) and (2) $\mathbf{x}_i^{(j)} = D(\mathbf{z}_i^{(j)})$ $(j = 1, 2, 3, 4)$, which are candidates sampled by latent-space interpolation between the current-best candidate and a BO-generated candidate (see subsection 4.4), where the corresponding interpolation weights are 0.25, 0.5, 0.75, and 1.0. We use Slerp for the interpolation following (Roberts, Engel, et al., 2018). In addition, (3) $\mathbf{x}_i^{\text{edited}}$, which is the candidate after the user's editing, is also involved here. We interpret this situation as relative-comparison feedback in latent space as

$$E(\mathbf{x}_i^{\text{edited}}) \succ \{E(\mathbf{x}_i^+), \mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{z}_i^{(3)}, \mathbf{z}_i^{(4)}\}, \tag{2}$$

which means that the latent vector $E(\mathbf{x}_i^{\text{edited}})$ is preferred over the other latent vectors on the right-hand side. This preferential feedback can be probabilistically modeled by the Bradley–Terry–Luce (BTL) model (Tsukida & Gupta, 2011).

4.4   Candidate Selection

In BO methods, the next candidates are determined by maximizing an *acquisition function*, which calculates the effectiveness of a candidate as the next query. There exist several acquisition functions (Shahriari et al., 2016), but the most popular choice is probably the one called the *expected improvement* (EI), as used in (Brochu et al., 2007; Koyama et al., 2017, 2020; Chong et al., 2019). It is defined in our problem setting as

$$a^{\text{EI}}(\mathbf{z}) = \mathbb{E}[\max\{0, f(D(\mathbf{z})) - f^+\}], \tag{3}$$

where f^+ is the estimated goodness value of the current-best melody. The maximization of the EI function is efficiently solved by standard optimization algorithms (e.g., L-BFGS).

5   Algorithm Evaluation

We conducted an experiment from the algorithmic viewpoint, where we artificially simulated user behaviors. The goal is to verify whether our algorithm can find the

desired melody in a reasonable number of iterations. We designed this experiment based on the assumption that there is one specific melody, which is the user's desired melody. In the experiment, we used *soft dynamic time warping* (s-DTW), an improvement of the original DTW, as the quantitative similarity metric between two melodies in latent space (Cuturi & Blondel, 2017).

We simulated user behaviors as follows. First, we generated a desired melody, $\mathbf{x}^{\text{target}}$, by random sampling on the latent space. Given the set of candidate melodies in the t-th iteration, $\{\mathbf{x}_t^+, \mathbf{x}_t^{(1)}, \ldots, \mathbf{x}_t^{(4)}\}$, we selected the one with the smallest s-DTW to $\mathbf{x}^{\text{target}}$ as the best preferred one at the iteration. We iterated this procedure for $t = 1, \ldots, 50$. We performed 100 trials in total. In addition to the s-DTW values, we also recorded the 4-dimensional latent vectors of all the selected candidate melodies and calculated the Euclidean distances to the latent vector of the desired melody. Note that we regenerated the desired melody for every trial.

Figure 2 (left) shows the plot of the s-DTW values, and Figure 2 (right) shows the plot of the corresponding latent-space Euclidean distances. Each dot on the plot represents the mean of 100 trials. We could observe that the s-DTW values decreased as iterations went on, which was our expectation. We found the Euclidean distances in the latent space also decreased consistently, which indicates the smoothness of the latent space. The results suggest that BO could effectively find melodies that are reasonably similar to the target one after around 20 iterations in the 4-dimensional latent space.
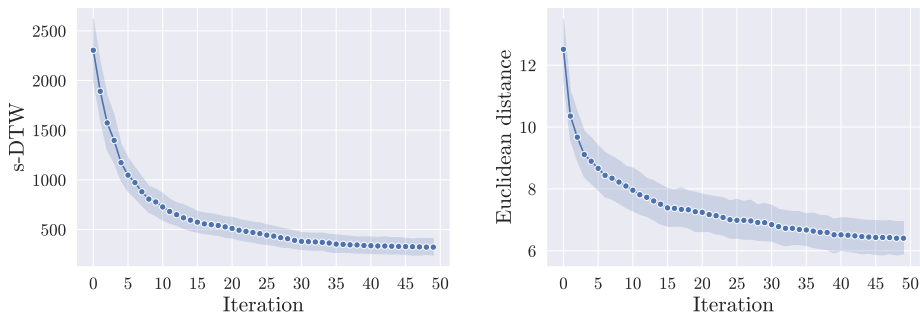

Fig. 2. Simulation experiment results. Each plot shows the mean of 100 trials with the colored region showing the standard deviation. The left shows s-DTW and the right shows the Euclidean distance in the 4-dimensional latent space.

6   Pilot Study

To obtain initial qualitative feedback on the proposed approach, we conducted an informal pilot study with two novice composers, who had wished to compose music but had no formal composition training. We asked one of the two participants

(P_1) to compose a 32-bar melody, and the other one (P_2) to compose a 16-bar melody with our system. Each participant was first tutored by the experimenter about the features of the system. During the experiment, the participants were free to compose melodies by themselves or use the "Search" function to get help from the system. When they used the "Search" function, they were asked to go for at least 20 iterations.

The composed melodies can be found at the our web source[4]. P_1 finished the composing within 2.5 hours, while P_2 finished it within 1.5 hours. In the experiment, P_1 used the "Search" function 8 times, while P_2 used it 6 times.

In the follow-up interview, both the participants explicitly expressed that the system had effectively provided them more ideas than they had, which inspired their creativity. *"Before I used the 'Search' function, I didn't realize that there could be so many possibilities developed from my idea"* (P_1). *"After I heard the candidates suggested, I got some fantastic ideas based on them"* (P_2). Both of the participants also mentioned that the system made the process of composition easier. *"I tried to compose before but got stuck in extending my idea. But in this experiment, just 'Search'. It saves my life"* (P_1). *"I liked it because it seemed that those four candidates in each iteration were sorted from similar to different, as compared to my original idea. It made it easier to select one to fit my melody"* (P_2).

However, they also mentioned room for improvement or drawbacks they noticed. First, P_1 mentioned that she wanted more control on the search function, saying *"It would be better if I can set the degree of how different, or how aggressive the candidates to be. Because I already had some ideas on how the melody goes"*. Besides, P_2 wanted the system to take the sequential musical information into account when searching for candidates, saying *"Even though it found some variations on the melody clip I composed in a previous paragraph, sometimes the candidates found did not fit a new paragraph"*.

7   Conclusion

In this paper, we demonstrated that human-in-the-loop BO had the potential to work effectively and efficiently on searching for the user's desired melody from the latent melody space. We also showed that BO could suggest satisfying candidates within a reasonable number of iterations. The pilot study result suggested that it also inspired the user's creativity and lowered the learning barriers of novice composers. In the future, we plan to investigate how to apply BO to a higher dimensionality (e.g., 512) efficiently and effectively in an interactive system. We also want to provide more control in terms of the searching behavior, and to consider the sequential information, as indicated by the pilot study participants. Finally, we hope our work can inspire researchers to more deeply investigate designing music creation tools, or even creativity support tools in a broader context, by incorporating the human-AI mixed-initiative interaction mechanism.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proc. KDD '19* (pp. 2623–2631). doi: 10.1145/3292500.3330701

Brochu, E., Brochu, T., & de Freitas, N. (2010). A Bayesian Interactive Optimization Approach to Procedural Animation Design. In *Proc. SCA '10* (pp. 103–112). doi: 10.2312/SCA/SCA10/103-112

Brochu, E., Freitas, N. d., & Ghosh, A. (2007). Active Preference Learning with Discrete Choice Data. In *Proc. NIPS '07* (pp. 409–416).

Chiu, C.-H., Koyama, Y., Lai, Y.-C., Igarashi, T., & Yue, Y. (2020). Human-in-the-Loop Differential Subspace Search in High-Dimensional Latent Space. *ACM Transactions on Graphics*, *39*(4), 85:1–85:15. doi: 10.1145/3386569.3392409

Chong, T. L. H., Shen, I.-C., Sato, I., & Igarashi, T. (2019). Interactive Subspace Exploration on Generative Image Modelling. *ArXiv*, *abs/1906.09840*.

Cuturi, M., & Blondel, M. (2017). Soft-DTW: a Differentiable Loss Function for Time-Series. In *Proc. ICML '17* (Vol. 70, pp. 894–903). doi: 10.5555/3305381.3305474

Dinculescu, M., Engel, J., & Roberts, A. (2019). MidiMe: Personalizing a MusicVAE model with user data. In *Proc. NeurIPS Workshop on Machine Learning for Creativity and Design '19.*

Dong, H., Hsiao, W., Yang, L., & Yang, Y. (2018). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaninproceedingsiment. In *Proc. AAAI '18* (pp. 34–41).

Esling, P., Masuda, N., Bardet, A., Despres, R., & Chemla-Romeu-Santos, A. (2019). Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows. *Applied Sciences*, *10*, 302. doi: 10.3390/app10010302

Fernández, J. D., & Vico, F. (2013). AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, *48*(1), 513–582.

Ghosh, P., Sajjadi, M. S. M., Vergari, A., Black, M. J., & Schölkopf, B. (2019). From Variational to Deterministic Autoencoders. *ArXiv*, *abs/1903.12436*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Nets. In *Proc. NIPS '14* (pp. 2672–2680).

Grinberg, M. (2014). *Flask Web Development: Developing Web Applications with Python* (1st ed.). O'Reilly Media, Inc.

Herremans, D., Chuan, C.-H., & Chew, E. (2017). A Functional Taxonomy of Music Generation Systems. *ACM Computing Surveys*, *50*(5). doi: 10.1145/3108242

Huang, A., Chen, S., Nelson, M., & Eck, D. (2018). Towards mixed-initiative generation of multi-channel sequential structure. In *Proc. ICLR Workshop '18.*

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proc. ICLR '14.*

Koyama, Y., Sato, I., & Goto, M. (2020). Sequential Gallery for Interactive Visual Design Optimization. *ACM Trans. Graph.*, *39*(4), 88:1–88:12. doi: 10.1145/3386569.3392444

Koyama, Y., Sato, I., Sakamoto, D., & Igarashi, T. (2017). Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Transactions on Graphics*, *36*(4), 48:1–48:11. doi: 10.1145/3072959.3073598

Louie, R., Coenen, A., Huang, C. Z., Terry, M., & Cai, C. J. (2020). Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models. In *Proc. CHI '20* (pp. 1–13). doi: 10.1145/3313831.3376739

Magenta. (2013). *NoteSequence.* `https://github.com/magenta/note-seq`.

Roberts, A., Engel, J., Mann, Y., Gillick, J., Kayacik, C., Nørly, S., . . . Eck, D. (2019). Magenta Studio: Augmenting Creativity with Deep Learning in Ableton Live. In *Proc. MUME '19.*

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2018). A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. In *Proc. ICML '18* (pp. 4364–4373).

Roberts, A., Hawthorne, C., & Simon, I. (2018). Magenta.js: A JavaScript API for Augmenting Creativity with Deep Learning. In *Joint Workshop on Machine Learning for Music (ICML).*

Rothstein, J. (1996). *MIDI: A Comprehensive Introduction* (2nd ed.). USA: A-R Editions, Inc.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE*, *104*(1), 148–175. doi: 10.1109/JPROC.2015.2494218

Simões, J. M., Machado, P., & Rodrigues, A. C. (2019). Deep Learning for Expressive Music Generation. In *Proc. ARTECH '19* (pp. 1–9). doi: 10.1145/3359852.3359898

Tsukida, K., & Gupta, M. R. (2011). *How to Analyze Paired Comparison Data* (Tech. Rep. No. UWEETR-2011-0004). University of Washington, Department of Electrical Engineering. Retrieved from `https://vannevar.ece.uw.edu/techsite/papers/refer/UWEETR-2011-0004.html`

Vogl, R., & Knees, P. (2016). An Intelligent Musical Rhythm Variation Interface. In *Proc. IUI '16 Companion* (pp. 88–91). doi: 10.1145/2876456.2879471

Vondrick, C., Pirsiavash, H., & Torralba, A. (2016). Generating Videos with Scene Dynamics. In *Proc. NIPS '16* (pp. 613–621).

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., & De Freitas, N. (2016). Bayesian Optimization in a Billion Dimensions via Random Embeddings. *J. Artif. Int. Res.*, *55*(1), 361–387. doi: 10.1613/jair.4806

Yang, L., Szu-Yu, C., & Yang, Y. (2017). MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation. In *Proc., ISMIR '17* (pp. 324–331).

Zhu, J.-Y., Krähenbühl, P., Shechtman, E., & Efros, A. A. (2016). Generative Visual Manipulation on the Natural Image Manifold. In *Proc. ECCV '16* (pp. 597–613).