In *Proceedings of the 16th International Symposium on Computer Music Multidisciplinary Research* (CMMR 2023),
Tokyo, Japan, November 2023

VaryNote: A Method to Automatically Vary the Number of Notes in Symbolic Music

Juan M. Huerta[1] and Bo Liu[1] and Peter Stone[1,2]

[1] Department of Computer Science, The University of Texas at Austin
[2] Sony AI
jmhuer@utexas.edu, {bliu, pstone}@cs.utexas.edu

Abstract. Automatically varying the number of notes in symbolic music has various applications in assisting music creators to embellish simple tunes or to reduce complex music to its core idea. In this paper, we formulate the problem of varying the number of notes while preserving the essence of the original music. Our method, *VaryNote*, adopts an autoencoder architecture in combination with a masking mechanism to control the number of notes. To train the weights of the pitch autoencoder we present a novel surrogate divergence, combining the loss of pitch reconstructions with chord predictions end-to-end. We evaluate our results by plotting chord recognition accuracy with increasing and decreasing number of notes, analysing absolute and relative musical features with a probabilistic framework, and by conducting human surveys. The human survey results indicate humans prefer VaryNote output (with 1.5, 1.9 $\times$ notes) over the original music, suggesting that it can be a useful tool in music generation applications. [3] [4]

Keywords: Pitch Autoencoder, Harmonic Analysis, Arrangement Generation, Automatic Ornamentation, Symbolic Music Generation, Chord Predictions

1 Introduction

Automating the process of varying the number of notes in a musical arrangement can have many applications. In the case of increasing notes, we can apply this technology to enhance compositions. This application has previously been explored, to some degree, when discussing automatic melody harmonization, arrangement generation, or automatic ornamentation [3, 4, 14–16]. However most of those methods require supervision in the form of labeled data such as Wang et al. POP909 MIDI dataset with segmented melody, arrangement, and bridge notes [17]. The other direction of reducing the number of notes is considered a useful research area relevant to voicing information, automatic melody extraction, and feature extraction in general. However, similar limitations exist

[3] Project page and listening examples: https://varynote.github.io
[4] Code: https://github.com/varynote/varynote-code

for this case: all the methods require segmented data and do not allow for continuous control over the number of notes. In the field of music theory, Schenkerian analysis, or similar variants, can be used to uncover the underlying hierarchical structure of music and use this information to both reduce and add notes to music. However to implement this process automatically, a corpus of analyzed examples is needed, in addition to heuristics to determine how to add or remove notes based on the analysis [7, 13].

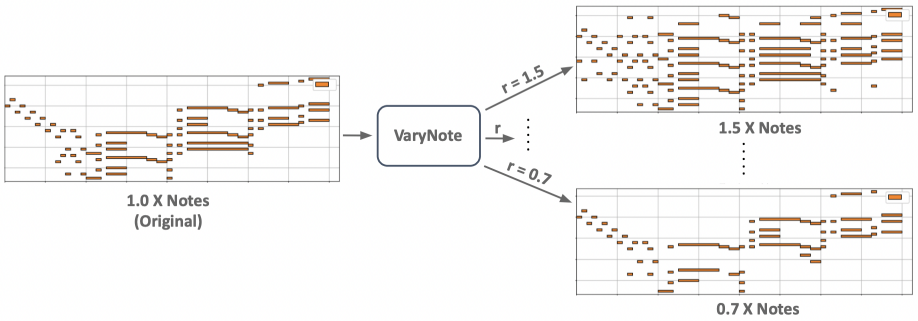

Fig. 1. VaryNote example usage: given a piece of MIDI music we varying the number of notes according to a desired input-output ratio: r.

To approach the problem of varying the number of notes in symbolic music automatically, we introduce VaryNote, a novel method that uses an autoencoder trained on pitch reconstructions that preserve chord structure. This design is considering several studies that have surveyed human listeners and discovered maintaining harmonic chord structure, while removing other aspects can still allow human listeners to recognize the original tune [6]. An example is the common practice of describing the chord progression I-V-vi-iii-IV-I-IV-V in terms of Pachelbel's Canon in D. In addition, VaryNotes' design effectively preserves rhythmic features, which we believe is beneficial as listeners can recall a song based on its melody, even with different instrumentation or tempo [2, 5, 18]. In summary, this paper makes the following contributions:

1. Formulate the task of varying the number of notes in music as an optimization problem.
2. Introduce VaryNote, a novel deep learning method consisting of an autoencoder trained with a combined loss of pitch reconstructions and chord predictions. We demonstrate that VaryNote can significantly outperform a baseline based on heuristics from music theory at the task of varying the number of notes on the BTL MIDI dataset.

2 Problem Formulation and Background

In this section, we introduce a general problem formulation of varying the number of notes. Then, in Section 3, we provide a description of our proposed strategy to solve this problem.

2.1 Problem Formulation

Symbolic music information is a type of sequential information. We represent music using a piano roll representation, a frame-wise representation, where every time step is a multi-hot encoding of the pitches that are played at time t. Assuming a time-length H, with P possible note pitches, we denote $\mathcal{X} = \{0,1\}^{P\times H}$ as the input space. We define a piano roll matrix $X \in \mathcal{X}$, and quantify the number of notes as the sum of non-zero elements[5] in X:

$$\textit{Number of Notes}: \; m := ||X||_0. \tag{1}$$

The goal is to learn a mapping $f_\theta(X \mid r) \to \hat{X} \in \mathcal{X}$ parameterized by θ such that $\hat{X}$ increases or decreases the number of notes in a piano roll X, given an *output-input ratio*, $r \in \mathbb{R}^+$ of notes that controls the relative sparsity of the output. Formally, we view the problem of automatically varying the complexity of harmonies as the following optimization problem:

$$\min_\theta \mathcal{D}\Big(f_\theta(X \mid r),\; X\Big) \;\text{ s.t. }\; \frac{||f_\theta(X \mid r)||_0}{||X||_0} = r. \tag{2}$$

Conceptually, $\mathcal{D}$ is a divergence that measures how similar the reconstructed $f_\theta(X \mid r)$ is to the original piece of music X. Informally, it can be characterized as the degree to which an average human listener would consider the two passages to be "the same tune" and is related to cover song identification [9]. While both melodic contour and harmonic contour can be used to quantify music similarity in music theory, they may not provide a complete picture due to subjective differences in interpretation and other factors. Ultimately, this divergence is based on human judgement and is not easily measurable so we resort to using a surrogate loss defined in Section. 3.2 that estimates the ability for a reconstructed piano roll to identify the original chords. We assume that when this loss is small, people will consider the passages to be the same tune. This assumption is considering the importance of harmonic structure in perceptual similarity [5, 6, 18]. However, we are not making any claim that this surrogate loss is the best possible quantitative estimation of the true divergence.

3 Method

The general problem presented in Eq. (2), is to conditionally generate music based on r. A straightforward approach is to first apply representation learning on the music and then reconstruct it conditioned on r, similar to autoencoder style models in machine learning. In this section, we introduce a novel autoencoder, named VaryNote. Specifically, VaryNote consists of two parts. The first is a pitch autoencoder (Section 3.1) where the encoder compresses a piece of music into a latent representation and the decoder reconstructs music from the latent representation. The second is a threshold mask (Section 3.1) that controls the sparsity in the output music. To train the weights of the pitch autoencoder we define a novel divergence in Section 3.2. This divergence is a combination of error on reconstruction and error on symbolic chord predictions.

[5] This definition is not exactly aligned with the music theory concept of a note, since we are ignoring note length, but it captures the amount of pitch information and is simple to calculate.

3.1 Architecture

Pitch Autoecoder An autoencoder is a model that seeks to learn a compressed representation of an input. It does so by passing the input through an information bottleneck of lower dimensionality than the original input. We apply an autoencoder to a piano roll X_t. We first breakdown the piano roll matrix by time-step, defining a sequence of pitch vectors such as:

$$X_{\mathrm{t}} \triangleq x_{t-H:t} = [x_{t-H}, \ldots, x_{t-1}]\,. \tag{3}$$

The goal is to learn to reconstruct a pitch vector x_t at time t using the encoder with $d = 32$: $E_\phi : \mathbb{R}^{\mathcal{X}} \rightarrow \mathbb{R}^d$ and decoder $D_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^{\mathcal{X}}$, parameterized by ϕ and θ respectively. In addition we test several non-linear activations λ:

- **ReLU:** rectified linear activation function.
- **k-WTA:** the k-largest neurons in the autoencoder's hidden layer (or code) is kept and the rest, as well as their derivatives, are set to zero [11]

$$\lambda_{WTA}(\boldsymbol{y} \mid k)_j = \begin{cases} y_j, & y_j \in \{k\text{-th largest elements of } \boldsymbol{y}\} \\ 0, & \text{Otherwise.} \end{cases} \tag{4}$$

- **Lifetime sparsity:** this is the same as k-WTA constraints Eq. (4) except we apply percent sparsity $\%k$ of the hidden layer across the entire mini-batch. This encourages a wider range of neurons to be active [12].

We encapsulate the autoencoder in a function $\mathcal{A}$, and define the autoencoder reconstruction as $\hat{X}_t$:

$$\hat{X}_t \triangleq \mathcal{A}_{\phi,\theta}(X_t) = D_\theta \circ \lambda[E_\phi(X_t)]. \tag{5}$$

Thresholding Piano Rolls After training the autoencoder, VaryNote reduces or increases the number of notes in a piano roll using a threshold mask. This mask essentially zeros out everything except the top-k values in the autoencoder output. Specifically, consider the autoencoder output of size $S = P \times H$ with $\hat{X}_t \in \mathbb{R}^S$ as defined in Eq. (5). Denote the k-th smallest element of $\hat{X}_t$ as $\hat{x}^{(k)}$. We define a mask M:

$$M(\hat{X}_t) = \mathbf{1}(\hat{x}_{i,j} \geq \hat{x}^{(k)}). \tag{6}$$

For any desired *output-input ratio* r, and m number of notes in the original piano roll X_t, we find a k-th order that achieves r:

$$k(r) = \lfloor S - rm \rfloor. \tag{7}$$

Now we can write Eq. (6) using a target r:

$$M(\hat{X}_t \mid r) = \mathbf{1}(\hat{x}_{i,j} \geq \hat{x}^{(k(r))}). \tag{8}$$

Applying the mask defined in Eq. (8) on $\hat{X}_t$ assures we end up with rm number of notes:

$$||M(\hat{X}_t \mid r)||_0 = S - k \approx rm. \tag{9}$$

VaryNote Architecture At this point we have described all the required components of VaryNote. We have slightly different treatment for increasing or decreasing notes. *Increasing Number of Notes:* to apply a relative increase in number of notes (output-input ratio $r \geq 1$), we add the pitch autoencoder output with the original music and apply the mask in Eq. (8) that assures we meet the desired output-input ratio constraints. *Decreasing Number of Notes:* to apply a relative decrease in number of notes (output-input ratio $r < 1$), we multiply, element-wise, the pitch autoencoder output with the original music and apply the mask in Eq. (8) that assures we meet the desired output-input ratio constraints. In summary:

$$F_{vc}(X \mid r) = \begin{cases} \mathbf{M}(\mathcal{A}_{\phi,\theta}(X) + X \mid r), & \text{if } r \geq 1 \\ \mathbf{M}([X + \mathcal{A}_{\phi,\theta}(X)] * X \mid r), & \text{if } r < 1. \end{cases} \tag{10}$$

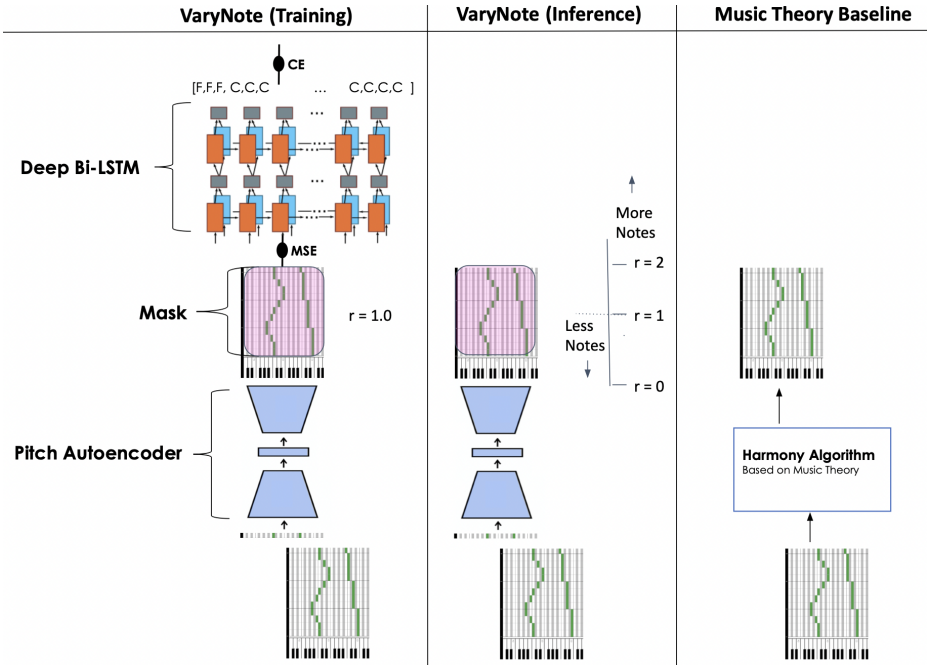

Fig. 2. During training VaryNote combines MSE loss and softmax cross entropy loss. Note the mask requires an output-input ratio r. During training we can fix r; or train without masking, and apply the mask during inference. During inference, r controls the number of notes.

Bi-LSTM Architecture for Chord Recognition To train the weights of the autoencoder $\mathcal{A}_{\phi,\theta}$, VaryNote temporarily attaches a Bi-LSTM [1] that uses the output of the autoencoder to make chord predictions as a downstream task. This addition helps our

pitch reconstructions maintain the original chord structure. The task is to find a mapping from $X_t = [x_{t-H}, \ldots, x_{t-1}] \in \{0,1\}^{P \times H}$ to a corresponding chord sequence per time step $Y_t = [y_{t-H}, \ldots, y_{t-1}] \in \mathbb{Z}^C$ where C is the number of symbolic chord classes. The output sequence is passed through a softmax layer that generates the probability for each pitch vector.

3.2 VaryNote Surrogate Loss: MSE and Chord Recognition

Ideally we want a differential metric, $\mathcal{D}$, that measures music similarity across different arrangement representations as described in Eq. (2). This divergence is not easily quantifiable, so we resort to designing a combined loss that preserves chord structure during pitch reconstructions. Specifically, we propose a combined loss of mean-square error on the pitch autoencoder reconstruction and cross entropy on symbolic chord targets between the Bi-LSTM output $o_t \in \mathbb{R}^{C \times H}$ and target sequence. In our study we reduce the number of pitches to $P = 64$ and predict $N = 24$ possible chords. The total loss can be described as:

$$\begin{aligned} \mathcal{D} = L_{\text{total}} &= L_{\text{MSE}} + cL_{\text{CE}} \\ &= \frac{1}{P}\sum_{t=1}^{P}(x_t - \hat{x}_t)^2 - \frac{c}{N}\sum_{i=1}^{N}\log\frac{\exp\big(o_t[y_i]\big)}{\sum_{y=1}^{K}\exp\big(o_t[y]\big)}. \end{aligned} \tag{11}$$

Finally, VaryNote trains with the presented L_{total}

$$\min_{\theta,\phi} L_{\text{total}}\Big(F_{vc}(X \mid r),\ X\Big) \ \text{ s.t. } \frac{||F_{vc}(X \mid r)||_0}{||X||_0} = r. \tag{12}$$

The constraints are automatically met by the mask M. During training we can fix r. Alternatively, VaryNote can train without a mask by using the autoencoder output with no threshold, and then apply a mask during inference.

3.3 Music Theory Baseline

VaryNote enables varying the number of notes along a continuous spectrum from very sparse to very dense orchestration. There are no existing rule-based methods that can similarly control the number of notes in the same way. There are relevant examples of music algorithms based on theory rules such as voice leading applied to automatic harmonization. However, none of these methods provide a comparison as we increase or decrease notes. So we designed a method that can automatically generate harmonic intervals and automatically remove notes. To add notes, the algorithm requires two steps. First we sample harmonic intervals from a probability distribution computed from aggregating music theory rules used in prior work [10]. Table 3 in the Appendix summarizes the weighted probabilities of harmonic intervals. To remove notes, we randomly find a note with probability proportional to the density of notes at each time step.

4 Experiment

We conduct experiments to compare the original music with the output of VaryNote variants and the baseline method, using three different criteria. Section 4.3 examines the impact on chord structure when notes are added or removed, Section 4.4 compares various musical features using a probabilistic framework, and Section 4.5 evaluates the perception of VaryNote's output through a human survey. To verify that the surrogate loss in Eq. (11) is superior to a standard MSE loss, we compare VaryNote variants with a standard autoencoder. We also test k-WTA and Lifetime sparsity constraints on the autoencoders with the expectation they will achieve better generalization on pitch reconstructions. In more detail:

- **Lifetime:** VaryNote with Lifetime ($k = 3$) sparsity constraints, described in Section 3.1.
- **k-WTA:** VaryNote with k-WTA ($k = 3$) sparsity constraints, described in Section 3.1
- **Ordinary:** VaryNote with no sparsity constraints, using a standard ReLU activation, described in Section 3.1
- **AE:** VaryNote with no sparsity constraints, trained only with L_{MSE}. That is $c = 0$ in Eq. (11).
- **Rules:** This is the Music Theory baseline: a simple algorithm that can generate harmonic intervals sampled from weighted probabilities in Table 3 in the Appendix, see Section 3.3 for more details.

4.1 Dataset

We use the BPS-FH dataset with 32 movements of Beethoven Piano Sonatas [1]. The musical pieces in the repertoire are represented as binary piano rolls with the time resolution of one 16th note. A sliding window of length 128 time-steps (equal to 32 quarter notes) with a hop size of 16 is applied to the piano rolls to generate the instances for recognition. For chord recognition, we use the maj-min chord vocabulary (including 24 major and minor chords plus an additional 'others' class which is excluded from evaluation). We only consider 64 pitches, excluding the lowest and highest octave of the standard 88 key piano notes.

4.2 Model Training

We train VaryNote, Eq. (12), without a threshold mask $\mathbf{M}$ and apply the mask during inference. All VaryNote models are trained with the same train-validation BPS-FH dataset. We train each method for 20 epochs using Adam optimizer, and use $c = \frac{1}{3}$ for our loss Eq. (11) (i.e: MSE + $\frac{1}{3}$ CE). In the interest of reproducibility, all experimental parameters are stored in the code repository.[6]

[6] See the README.md file in the code repository

4.3 Recovering Chord Information

To verify that the added or reduced notes do not significantly affect the harmonic structure of music we test if we can recover ground truth chords from the original piano roll (Fig. 3). To accomplish this, first we train each method. Then we transform the validation data using note multiples: $r \in [0.3, 0.5, 0.7, 1, 1.3, 1.5, 1.9]$. Finally, using a separate and isolated Bi-LSTM model trained on the original data, we predict symbolic chords for each note multiple.

4.4 Music Similarity with Kullback-Leibler Divergence

To get a sense of the music similarity without using a human analyst, we apply Lerch e.t. multi-criteria evaluation metrics based on probabilistic measures of musical features [8]. We compare the original MIDI music datasets against every method with 1.5 $\times$ notes by applying kernel density estimation (Gaussian kernel) to find a Probability Density Function (PDF) for each musical feature, and plot them in Fig. 4. Related to harmony, we measure *Pitch Count (PC)*: the number of different pitches within a sample, *Pitch Range (PR)*: the difference of the highest and lowest used pitch in semitones, and *Average Pitch Interval (PI)*: the average value of the interval between two consecutive pitches in semitones. Related to rhythm, we measure *Average Inter-Onset-Interval (IOI)*: the time between two consecutive notes.

4.5 Human Evaluations

In order to evaluate the practical use of this method, we conduct a small survey designed to understand how human listeners, musically trained and untrained, judge reduced/added note transformations[7]. The goal is to understand if the transformations sound realistic, pleasant, and match our expectation of complexity. $11/30$ participants self-report knowing how to play an instrument. We test results for VaryNote Lifetime since it is the best performing method. The survey has three sections. The first is *Musical Preference*: the participants are asked to score VaryNote output from 1-5, 1 being the lowest appeal, and 5 being the highest appeal. The second is *Perceived Musical Complexity*: the participants are asked to score VaryNote output from 1-5, 1 being the lowest complexity, and 5 being the highest complexity[8]. The final is *Music Turing Tests (MTT)*: the participants are given two examples, VaryNote output, and the original music and are asked to identify the piece of music that was fully composed by a human–we do this for piano, and multi-instrument output. The piece the participant selects as being composed by a human receives a score of 1. We sum the total scores and divide by the total number of participants to get a proportion of times humans select the VaryNote output over the original music. To generate a multi-instrument output we simply isolate the notes from the VaryNote output and synthesize the MIDI with a new instrument. Results are summarized in Table 1 and Table 2, the best mean for each question is shown in bold.

[7] The survey form is available in the code repository.

[8] This question is intended to provide insight into how listeners perceive and differentiate between music with different note multiples. It is worth noting that the use of the term "complexity" was chosen to align with a previous version of the paper.

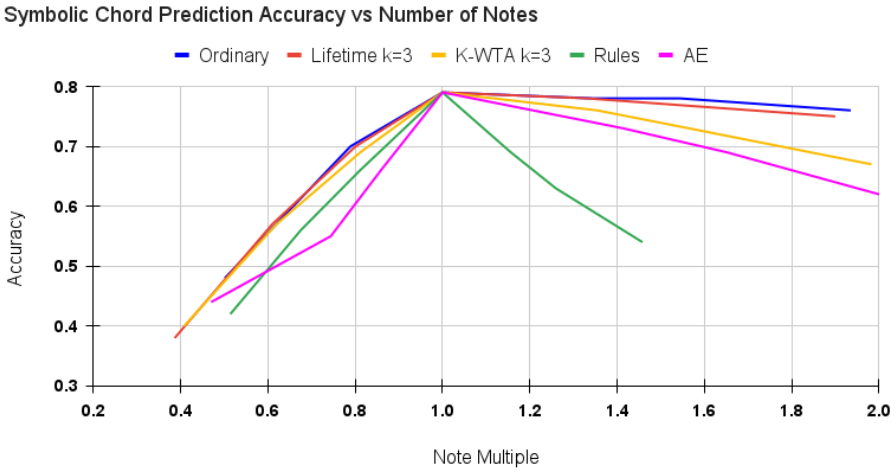

Fig. 3. Symbolic chord prediction accuracy using a Bi-LSTM model trained on the original data as we transform our validation data using VaryNote

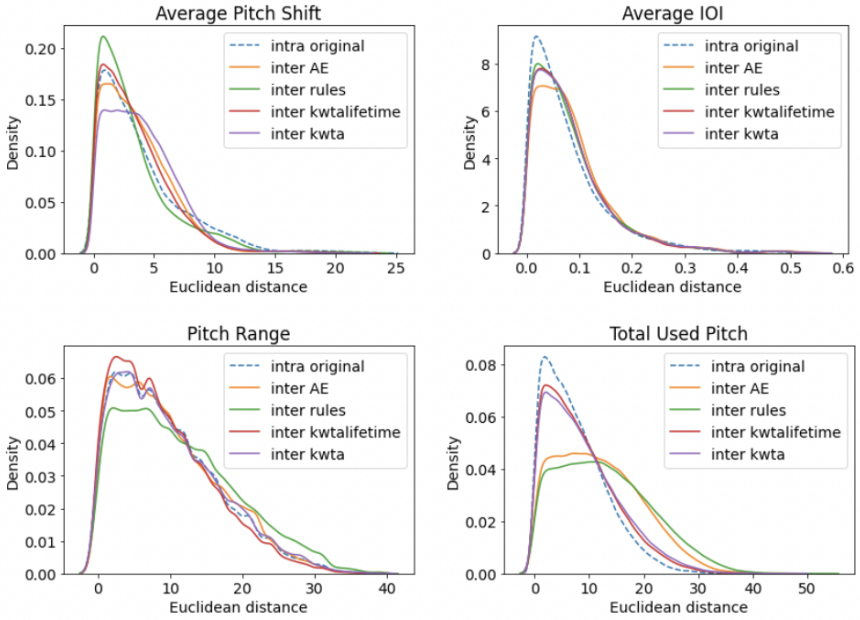

Fig. 4. We extract certain features and use kernel density estimation (Gaussian kernel) to find a probability density function for specific dataset generated by a model. "Intra" refers to comparisons made within a single group of the orignal music. "Inter," on the other hand, refers to comparisons made between two different groups or categories, in this case comparisons made between the altered music and the original music.

Table 1. Human survey results for preference and complexity. Participants are asked to rate the VaryNote output based on preference on a scale of 1-5, 1 being the lowest appeal, and 5 being the highest appeal. Participants also rate complexity from 1-5, 1 being the lowest complexity, and 5 being the highest complexity. There were 30 total participants; $11/30$ participants self-reported knowing how to play an instrument. The highest mean for each question is shown in bold.

Experiment	**Score Report**				
	Original	$\times$0.5 Notes	$\times$0.7 Notes	$\times$1.5 Notes	$\times$1.9 Notes
Preference Mean	3.09	2.15	2.73	**3.62**	3.41
Std. Deviation	1.33	1.23	1.23	1.11	1.35
Complexity Mean	3.25	1.62	2.52	**3.92**	3.85
Std. Deviation	1.61	1.21	1.46	1.24	1.32

Table 2. This table includes results for Music Turing Tests (MTT). The participants are given two examples, VaryNote output, and the original music, and are asked to identify the piece of music that is fully composed by a human. The piece the participant selects as being composed by a human receives a score of 1. We sum the total scores and divide by the total number of participants to get a proportion of times humans select the VaryNote output over the original music. The multi-instrument question uses string and woodwind MIDI instruments.

Experiment	$\times$0.5 Notes	$\times$1.5 Notes	$\times$1.9 Notes
Music Turing Test (MTT) - Piano	0.22	**0.36**	0.17
MTT - Multi-Instrument	N/A	**0.57**	N/A

5 Discussion

As we vary the note multiple r, the Ordinary and Lifetime methods achieve the highest accuracy in chord recognition according to Fig. 3. We also measure similarity between the original music and 1.5 $\times$ note outputs using KL-divergence. All methods have very similar IOI values. Other harmonic features such as PC, PR, and PI, closely match the original music for all VaryNote methods, and the Rules method is clearly inferior at matching the distribution of the original music.

The human survey results in Table 1 indicate humans prefer VaryNote output, with 1.5, 1.9 $\times$ notes, over the original music. Table 2 indicates humans perceive increased complexity with higher note multiples, except that $1.5\times$ notes seems to be perceived with higher complexity than $1.9\times$ notes. On the MTT-Piano, participants identify the original music 64% of the time. In comparison, participants only identify MTT-Multi-instrument pieces 43% of the time.

6 Conclusion

In summary, we have introduced the task of automatic note variation in music and proposed a novel method, VaryNote, that outperforms a music theory baseline. The proposed method offers significant advantages over traditional approaches by generating a coherent range of outputs for any given note multiple. Notably, our method requires only a corpus of chord labels for training, and it can be easily extended to other divergence metrics beyond chord predictions. Moreover, our results indicate that VaryNote's output is preferred over the original music, suggesting that VaryNote can be a useful tool in music generation applications.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, FAIN-2019844), ONR (N00014-18-2243), ARO (W911NF-19-2-0333), DARPA, GM, Bosch, and UT Austin's Good Systems grand challenge. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

1. Tsung-Ping Chen and Li Su. Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models. *Transactions of the International Society for Music Information Retrieval*, 4(1):1–13, 2021.
2. W. Jay Dowling, James C. Bartlett, Andrea R. Halpern, and Melinda W. Andrews. Melody recognition at fast and slow tempos: Effects of age, experience, and familiarity. *Perception & Psychophysics*, 70(3):496–502, 2008.
3. Kemal Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.
4. Benjamin Evans, Satoru Fukayama, Masataka Goto, Nagisa Munekata, Tetsuo Ono, et al. Autochoruscreator: Four-part chorus generator with musical feature control, using search spaces constructed from rules of music theory. In *ICMC*, 2014.
5. Andrea R. Halpern, James C. Bartlett, and W. Jay Dowling. Perception of Mode, Rhythm, and Contour in Unfamiliar Melodies: Effects of Age and Experience. *Music Perception*, 15(4):335–355, 07 1998.
6. Ivan Jimenez and Tuire Kuusi. Connecting chord progressions with specific pieces of music. *Psychology of Music*, 46:716–733, 9 2018.
7. Phillip B. Kirlin and Jason Yust. Analysis of analysis: Using machine learning to evaluate the importance of music parameters for schenkerian analysis. *Journal of Mathematics and Music*, 10(2):127–148, 2016.
8. Li-chia Lerch and Alexander Yang. On the evaluation of generative models in music. *Neural Computing and Applications*, 32:4773–4784, 2020.
9. Elad Liebman, Peter Stone, Supervisor Kristen, Grauman Scott, Niekum Maytal, Saar-Tsechansky Roger, and B Dannenberg. Sequential decision making in artificial musical intelligence.

10. Chien-Hung Liu and Chuan-Kang Ting. Polyphonic accompaniment using genetic algorithm with music theory. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2012.
11. Alireza Makhzani and Brendan Frey. Winner-take-all autoencoders. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2791–2799, 2015.
12. Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.
13. Alan Marsden. Software for schenkerian analysis. In *ICMC*, 2011.
14. Montserrat Puiggròs, Emilia Gómez Gutiérrez, Rafael Ramírez, Xavier Serra, and Roberto Bresin. Automatic characterization of ornamentation from bassoon recordings for expressive synthesis. In *Baroni M, Addessi AR, Caterina R, Costa M, editors. 9th International Conference on Music Perception and Cognition; 2006 Aug 22-26; Bologna, Italy. Bologna: Bononia University Press; 2006.* Bononia University Press, 2006.
15. Rafael Ramirez and Amaury Hazan. A tool for generating and explaining expressive music performances of monophonic jazz melodies. *International Journal on Artificial Intelligence Tools*, 15(04):673–691, 2006.
16. Miguel Sarabia, Kyuhwa Lee, and Yiannis Demiris. Towards a synchronised grammars framework for adaptive musical human-robot collaboration. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 715–721. IEEE, 2015.
17. Ziyu Wang, Ke Chen, Junyan Jiang, Yiyi Zhang, Maoran Xu, Shuqi Dai, Xianbin Gu, and Gus Xia. Pop909: A pop-song dataset for music arrangement generation. *arXiv preprint arXiv:2008.07142*, 2020.
18. Richard M. Warren, Daniel A. Gardner, Bradley S. Brubaker, and James A. Jr. Bashford. Melodic and Nonmelodic Sequences of Tones: Effects of Duration on Perception. *Music Perception*, 8(3):277–289, 04 1991.

Appendix

A Weighted Probabilities of Harmonic Intervals

To add notes using the Rules approach, we sample harmonic intervals from a probability distribution computed from aggregating music theory rules used in prior work. The harmonic intervals are summarised in Table 3 below.

Table 3. Assigned probabilities p for intervals according to music theory rules from prior work [10]. To add a new note, a random note from the original music is selected uniformly and harmonized with a random interval drawn with probability p.

Assigned Prob. (p)		
$p = 0.19$	$p = 0.10$	$p = 0.003$
Perfect fourth	Minor third	Minor second
Perfect fifth	Major third	Major second
	Minor sixth	Minor seventh
	Major sixth	Major seventh
	Perfect octave	Augmented interval
	Perfect unison	Diminished interval