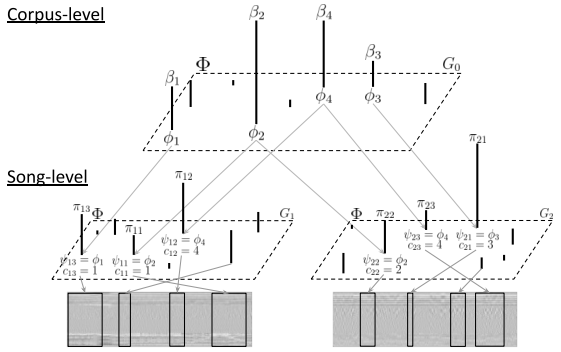



**Figure 2**: Illustration of 2-level HDPMM within corpus-song context. The top row depicts the corpus-level DP. The second row describes the draw of second-level (song-level) DPs per song from the corpus-level DP. Frame-level features are drawn from each song-level DPs, as depicted in the third row of the image.

minimizing the Kullback-Leibler (KL) divergence between the approximation $q(Z)$ and the true posterior $p(Z|X)$, where $Z$ denotes the set of latent variables and parameters that we want to find, and $X$ refers a set of observations [23]. One of the popular simplifications is the full factorization of the distribution $q(Z) = \prod_{i=1}^{|Z|} q_i(Z_i)$. In the context of HDPGMM, we have the following:

$$q(\beta', \pi', c, z, \phi) = q(\beta')q(\pi')q(c)q(z)q(\phi) \tag{5}$$

where $\beta', \pi', c, z$ denote the corpus-level and group-level stick proportions, group-level component selection variable, and finally the observation-level component selection variable, respectively. $\phi$ refers to the parameter(s) for the $F$ which draws the atomic observation, set as (multivariate) Gaussian in our context. Thus, $\phi$ includes the means $\mu$ and precision matrices $\Lambda$ of each Gaussian component. We adopt the result from [23], where the Gaussian-Wishart distribution is used for the prior.

Each variational distribution further factorizes into:

$$\begin{aligned} q(\beta') &= \textstyle\prod_k^{K-1} \text{Beta}(\beta'_k|u_k, v_k) \\ q(\pi') &= \textstyle\prod_t^{T-1} \text{Beta}(\pi'_t|a_t, b_t) \\ q(c) &= \textstyle\prod_j \prod_t \text{Mult}(c_{jt}|\varphi_{jt}) \\ q(z) &= \textstyle\prod_j \prod_n \text{Mult}(z_{jn}|\zeta_{jn}) \end{aligned} \tag{6}$$

where $u_k, v_k, a_t, b_t$ denote the variational parameters for the Beta distributions for corpus-level and group-level stick proportions, respectively. $\varphi_{jt} \in \mathbb{R}^K, \zeta_{jn} \in \mathbb{R}^T$ are the variational parameters for the Multinomial distribution to draw the selector $c$ and $z$. Also, we truncate the infinite Beta distributions by $K$ and $T$, which is a common way to avoid actually computing the infinite dimension [18]. With a sufficiently large number for the truncation, the model will still not be limited to the truncation and will only use the number of components optimal for the dataset. The final variational distribution is the Gaussian-Wishart distribution which draws the Gaussian parameters $\phi$ for $F$:

$$q(\phi) = \textstyle\prod_k^K \mathcal{N}(\mu_k|m_k, (\lambda_k\Lambda_k)^{-1})\mathcal{W}(\Lambda_k|W_k, \nu_k) \tag{7}$$

where we draw the precision $\Lambda_k \in \mathbb{R}^{d\times d}$ from the Wishart distribution with the variational parameter $W_k \in \mathbb{R}^{d\times d}$ and $\nu_k \in \mathbb{R}$, and the mean $\mu_k \in \mathbb{R}^d$ is drawn by the precision weighted by $\lambda_k \in \mathbb{R}$ and mean $m_k \in \mathbb{R}^d$.

We then can obtain the optimal model by maximizing the lower bound of the marginal log likelihood $\log p(X|Z)$ [23, 26, 27]:

$$\begin{aligned} &\log p(X|Z) \geq \mathbb{E}_q[\log p(X, \beta', \pi', c, z, \phi)] + H(q) \\ &= \textstyle\sum_j \big\{\mathbb{E}_q[\log\left(p(X_j|c_j, z_j, \phi)p(c_j|\beta')p(z_j|\pi'_j)p(\pi'_j|\alpha_0)\right)] \\ &\qquad + H(q(c_j)) + H(q(z_j)) + H(q(\pi'_j))\big\} \\ &\qquad + \mathbb{E}_q[\log p(\beta')p(\phi)] + H(q(\beta')) + H(q(\phi)) \end{aligned} \tag{8}$$

where $H(\cdot)$ denotes the entropy of given distribution, and $X_j = \{x_{j1}, x_{j2}, \cdots, x_{jN_j}\}$ is a set of observations within the $j$th group. [4]

Using the standard result of VI [18, 23, 27], the update rules for group-level parameters are derived as follows:

$$a_{jt} = 1 + \textstyle\sum_n \zeta_{jnt} \tag{9}$$

$$b_{jt} = \alpha_0 + \textstyle\sum_n \sum_{s=t+1}^T \zeta_{jns} \tag{10}$$

$$\varphi_{jtk} \propto \exp(\textstyle\sum_n \zeta_{jnt}\mathbb{E}_q[\log p(x_{jn}|\phi_k)] + \mathbb{E}_q[\log \beta_k]) \tag{11}$$

$$\zeta_{jnt} \propto \exp(\textstyle\sum_{k=1}^K \varphi_{jtk}\mathbb{E}_q[\log p(x_{jn}|\phi_k)] + \mathbb{E}_q[\log \pi_{jt}]) \tag{12}$$

Similarly, the update rules for the corpus-level parameters are as follows [23]:

$$u_k = 1 + \textstyle\sum_j \sum_{t=1}^T \varphi_{jtk} \tag{13}$$

$$v_k = \gamma + \textstyle\sum_j \sum_t^T \sum_{l=k+1}^K \varphi_{jtl} \tag{14}$$

$$\lambda_k = \lambda_0 + N_k \tag{15}$$

$$m_k = \lambda_k^{-1}(\lambda_0 m_0 + N_k \bar{x}_k) \tag{16}$$

$$W_k^{-1} = W_0^{-1} + N_k S_k + \frac{\lambda_0 N_k}{\lambda_0 + N_k}(\bar{x}_k - m_0)(\bar{x}_k - m_0)^\intercal \tag{17}$$

$$\nu_k = \nu_0 + N_k \tag{18}$$

where $\lambda_0 \in \mathbb{R}, m_0 \in \mathbb{R}^d, \nu_0 \in \mathbb{R}, W_0 \in \mathbb{R}^{d\times d}$ are the hyperparameters corresponding to the weight, location, degrees of freedom, and scale of Gaussian-Wishart distribution. The "sufficient statistics" and expectations in the update rules are defined as follows:

$$\begin{aligned} &\mathbb{E}_q[\log \beta_k] = \mathbb{E}_q[\log \beta'_k] + \textstyle\sum_{l=1}^{k-1} \mathbb{E}_q[\log(1 - \beta'_l)] \\ &\mathbb{E}_q[\log \beta'_k] = \Psi(u_k) - \Psi(u_k + v_k) \\ &\mathbb{E}_q[\log(1 - \beta'_k)] = \Psi(v_k) - \Psi(u_k + v_k) \\ &\mathbb{E}_q[\log \pi_{jt}] = \mathbb{E}_q[\log \pi'_{jt}] + \textstyle\sum_{s=1}^{t-1} \mathbb{E}_q[\log(1 - \pi'_s)] \\ &\mathbb{E}_q[\log \pi'_{jt}] = \Psi(a_{jt}) - \Psi(a_{jt} + b_{jt}) \\ &\mathbb{E}_q[\log(1 - \pi'_{jt})] = \Psi(b_{jt}) - \Psi(a_{jt} + b_{jt}) \\ &N_k = \textstyle\sum_j \sum_n r_{jnk} \\ &\bar{x}_k = \frac{1}{N_k} \textstyle\sum_j \sum_n r_{jnk} x_{jn} \\ &S_k = \frac{1}{N_k} \textstyle\sum_j \sum_n r_{jnk}(x_{jn} - \bar{x}_k)(x_{jn} - \bar{x}_k)^\intercal \end{aligned}$$

[4] Thus, the terms inside the sum over the group would further factorize into sums of per-group observations. We do not write these out for legibility and space considerations.

where $\Psi(\cdot)$ refers to the digamma function, and $r_{jnk} = \sum_{t=1}^{T} \zeta_{jnt}\varphi_{jtk}$ is the inferred "responsibility" of the $n$th observation of the $j$th group on the $k$th component. A standard batch update would compute statistics across the entire corpus, and update the corpus-level parameters. However, it may be slow or may suffer from too large or small numbers accumulated from a large-scale corpus.

We therefore employ the online VI, where corpus-level parameters are updated per mini-batch of groups passed. In this way, the early phase of model inference can be accelerated substantially compared to the full-batch update [18, 28]. The corpus-level update is then controlled by the learning rate $\rho_t = (\tau_0 + t)^{-\kappa}$, which is decayed over iterations parameterized by the offset $\tau_0 > 0$ and the rate $\kappa \in (0.5, 1]$:

$$Z_t = (1 - \rho_t)Z_{t-1} + \rho_t \tilde{Z}_t \tag{19}$$

where $\tilde{Z}_t$ denotes one of the corpus-level parameters from Eq. (13) to (18), which is updated by the given mini-batch at $t$th iteration, while $Z_{t-1}$ is the current parameter. To scale the update with respect to the mini-batch size, we weight $\tilde{Z}$ by the factor of $w = \frac{|\tilde{X}|}{|X|}$, where $|X|, |\tilde{X}|$ denote the number of groups within the entire observation set $X$ and the mini-batch of groups $\tilde{X}$. The overall inference algorithm is described in Algorithm 1.

**Algorithm 1:** Online VI for HDPGMM

```
Initialize φ = (φ_k)_{k=1}^K, u = (u_k)_{k=1}^K, v = (v_k)_{k=1}^K
  randomly and set t = 1.
while Stopping criterion is not met do
    Fetch a random mini-batch of groups X̃
    repeat
        Update a_j, b_j, ζ_j and φ_j using Eq. (9) to (12)
    until mini-batch likelihood stops improving;
    Compute u_k, v_k, λ_k, m_k, W_k, and ν_k using Eq. (13)
      to (18)
    Set ρ_t = (τ_0 + t)^{-κ}, t ← t + 1
    Update u_k, v_k, λ_k, m_k, W_k, and ν_k using Eq. (19)
end
```

Inspired by the implementation of [18, 27], we apply further regularization to the model. Specifically, we "splash" the uniform noise to the inferred responsibility $r_{jn}$ as it can be biased if the groups are corrupted or incomplete, such as the preview audio of the entire song:

$$\tilde{r}_{jn} = (1 - \eta_t)r_{jn} + \eta_t e \tag{20}$$

where $\eta_t$ is the regularization coefficient that determines the extent uniform noise $e = (e_k)_{k=1}^K$ is mixed into $r_{jn}$. $\eta_t = \frac{\eta_0 * 10^3}{(t+10^3)}$ also is defined as decaying function similar to the learning rate $\rho_t$.

## 3. EXPERIMENTAL SETUP

The overall experimental design is adopted from the recent music representation learning studies [20, 21, 29], where multiple downstream MIR tasks are tested with the feature set learned from the representation models. [5]

[5] The source code can be found at `https://github.com/eldrin/hdpgmm-music-experiments`.

### 3.1 Datasets, Features & Evaluation

We use a subset of Million Song Dataset (**MSD**) [30] as the dataset for the representation learning; more specifically, the audio preview excerpts from the MSD "train" subset introduced by [31, 32] [6]. For the evaluation of the representation, we employ three downstream tasks encompassing *music genre classification*, *music auto-tagging*, and *music recommendation*. For each target task, we chose the **GTZAN** dataset [33] with the fault-filtered split [34], the MagnaTagATune (**MTAT**) dataset [35] with the split from [36], and finally the Echo Nest taste profile subset as part of MSD (**Echonest**) [30] filtered by user and item frequency [37].

We evaluate the modeling using the cross-validation of task-specific prediction models. We fit the logistic regression models for GTZAN and MTAT datasets, taking the learned music representation as input and predicting the genre or music tags. We find the hyper-parameters of the logistic regression model by a randomized parameter search [38] for each run. As for the recommendation, we apply the item *K* Nearest Neighbor (*item-kNN*) method [39] where the song-song similarity is measured by the cosine similarity of the learned music representation. We adopt the data split introduced in [37]. For every trial, disjoint validation and test users are uniformly sampled, which includes $1142$ users for each. Then $30\%$ of listening records of these users are held out as testing records. We find the best *K* by conducting a grid search on the range $K \in \{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$ by measuring the accuracy on the validation users. Using the best *K*, we compute the final score on the held out test records of testing users. We take the average score over $5$ repetitions for each run to incorporate the random split effect.

We repeat $5$ evaluation runs for each learning trial to consider the various random effects (i.e., initialization, randomized search). We employ the commonly used accuracy measures in each task to assess them, which are further elaborated in Table 1.

| Id | # Samples | # Classes/# Users | Acc. Measure |
|---|---|---|---|
| MSD | $213,354$ | N/A | N/A |
| Echonest | $40,980$ | $571,355$ [7] | nDCG [40] |
| GTZAN | $1,000$ | 10 | F1 [41] |
| MTAT | $25,863$ | 50 | AUROC [42] |

**Table 1**: Details on the datasets. We use the *macro* average strategy for F1 and Area Under Receiver Operating Characteristic curve (AUROC) measure, and we consider the top $500$ recommended songs for computing the normalized Discounted Cumulative Gain (nDCG).

We select a set of audio features to infer the HDPGMM and non-DL baseline models inspired by [17]. The further details of the features can be found in Table 2. We use the implementation of `librosa` [43] with the default parameters for all the features.

[6] Length of preview audio differs per clip, where about 70% of samples are approximately 30 seconds and the rest are 1 minute, with a small subset longer than 1 minute.

[7] This refers the number of users in the dataset.

| Feature | Aspect | Dim. | Transform |
|---|---|---|---|
| MFCC | Timbre | 13 | - |
| ΔMFCC | Timbre | 13 | - |
| ΔΔMFCC | Timbre | 13 | - |
| Onset Strength [44] | Rhythm | 1 | log |
| Chroma [45] | Tonal | 12 | log |

**Table 2**: Details on the base audio features we employed for the experiment.

### 3.2 Comparisons

We evaluate 4 comparisons against HDPGMM, including DL-based and non-DL-based models.

- **G1** is the simplest model among the comparisons. The song-wise feature is represented by the concatenated parameters $\phi = \{\mu, \Sigma\}$ of the single multivariate Gaussian fitted on the frame-level base audio feature vectors within a song. We chose the square root of the diagonal in covariance, which means the feature is the concatenation of mean $\mu \in \mathcal{R}^{52}$ and standard deviation $\sigma \in \mathcal{R}^{52}$ of features.
- **VQ Codebook** can be deemed as the approximation of HDPGMM models [12]. First, a corpus-wise $K$-means clustering is fitted on the frame level features. Then, song-level feature is represented by the normalized frequency of cluster assignment of each frame-level feature within the song. [8] We set $K = 256$.
- **KIM** is a DL-based music representation trained in a simple self-supervised learning framework with a *VGG-like* [46] architecture [20]. We use the original implementation of the work, which takes the stereo mel-spectrogram of a 2.5 second-length clip as input. The representation is extracted from the last hidden layer before the prediction and summarized with global average and standard deviation pooling through the entire song.
- **CLMR** [29] is a recent DL-based music representation employing self-supervised learning through the contrastive learning method [47]. We employ the original implementation of [29] that uses the 2.67 second-length raw audio samples as input feature. [9] The representation is drawn from the last hidden layer and pooled with the global average over the entire song excerpt.
- **HDPGMM** uses the base audio feature with the whitening following [48]. As for song-level representation $y_j$, we employ the exponentiation of $\tilde{y}_{jk} = \exp(\mathbb{E}_q[\log p(X_j|c_j, z_j, \phi_k)])$ and normalize it over components and the length of the clip $y_{jk} = N_j^{-1} \frac{\tilde{y}_{jk}}{\sum_{k'=1}^{K} \tilde{y}_{jk'}}$. We adopt many hyper-parameter setups from the result of [18]; we set the truncation of components on the corpus level $K$ and the song level $T$ as 256 and 32 respectively, mini-batch size to 1024, learning rate parameters $\kappa = 0.6$ and $\tau_0 = 64$. We, however, explore the regularization rate $\eta_0 \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and total corpus size $|X| \in \{2 \times 10^3, 2 \times 10^4, 213354\}$ to examine their effect on the representation. [10]

For all the comparisons except G1, we repeat 3 learning trials in consideration of random effects. Within a trial, we iterated the update for 100 epochs for KIM, CLMR, and HDPGMM. Further, we set the representation's dimensionality to 256, homogeneous across all comparisons except $G1$. Finally, we take the logarithm of the representation $\hat{y}_j = \log(\max(y_j, 10^{-8}))$ if it is given as the probability simplex (i.e., VQCodebook, HDPGMM).

## 4. RESULT AND DISCUSSION

### 4.1 Effect of Learning Factors

Overall, we observe that the learning factors of HDPGMM can make a substantial difference. As Figure 3 shows, the result suggests that the additional regularization in Eq. (20) generally affects positively the performance of all the downstream tasks we tested. It implies that the entire song inputs would likely improve the representation further, as the regularization crudely masks the effect of the missing data due to the preview clipping to some extent.

Further, we find that the number of training samples positively affects the performances in general. We measured the effect by repeatedly sub-sampling the training data 5 times in two different sizes mentioned in Section 3.2, and conducted the same evaluation applied for the full training set. The result shows that the positive effect is logarithmic, suggesting that the model should be learned with an exponentially larger dataset to get a linear improvement in the performance. The recommendation task, on the other hand, indicates the effect may not be linear, as we observe that the learning with $2 \times 10^3$ samples outperforms the $2 \times 10^4$ samples. However, the largest dataset leads to a better representation than the smallest datasets.

### 4.2 Model Comparison

| Dataset | Measure | Model | Mean (±SD) |
|---|---|---|---|
| Echonest | nDCG | G1 | 0.0237 (± 0.0011) |
| | | VQCodebook | 0.0155 (± 0.0003) |
| | | KIM | 0.0257 (± 0.0015) |
| | | CLMR | **0.0362 (± 0.0012)** |
| | | HDPGMM | 0.0221 (± 0.0006) |
| GTZAN | F1 | G1 | 0.5396 (± 0.0049) |
| | | VQCodebook | 0.5777 (± 0.0022) |
| | | KIM | 0.5420 (± 0.0290) |
| | | CLMR | **0.6375 (± 0.0368)** |
| | | HDPGMM | **0.6467 (± 0.0069)** |
| MTAT | AUROC | G1 | 0.8441 (± 0.0006) |
| | | VQCodebook | 0.8386 (± 0.0013) |
| | | KIM | 0.8014 (± 0.0045) |
| | | CLMR | 0.8262 (± 0.0026) |
| | | HDPGMM | **0.8482 (± 0.0020)** |

**Table 3**: Evaluation results on downstream tasks.

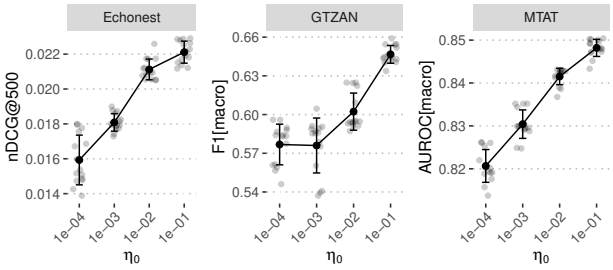
[8] It can be interpreted similarly to the inferred cluster assignment variable $\pi_j$ in Eq. (4).

[9] Unlike the original work, we reduce the dimensionality of the hidden layer, from which we extract the representation, from 512 to 256 to control the representation dimensionality with other comparisons. Also, we do not apply the data augmentation for a fair comparison and also as it is beyond the main scope of the work. However, as it is reported that it can meaningfully improve the representation, we assume that applying the method would benefit other models similarly.

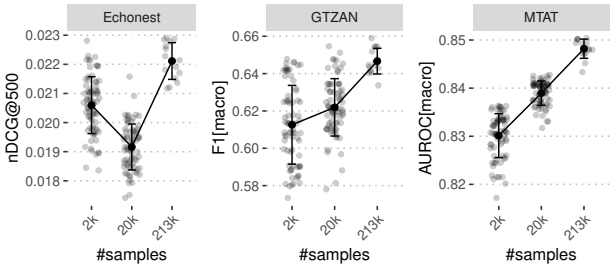
[10] The implementation can be found at `https://github.com/eldrin/pytorch-hdpgmm`.

| | | | | | | |
|---|---|---|---|---|---|---|
| Hip-Hop | country | female vocalists | pop | electronic | pop | oldies |
| pop | rock | singer-songwriter | female vocalists | dance | soul | blues |
| rnb | pop | pop | female vocalist | electronica | female vocalists | country |
| soul | oldies | acoustic | rock | funk | rnb | 60s |
| male vocalists | indie | Mellow | Love | electro | dance | soul |
| rock | singer-songwriter | folk | dance | Hip-Hop | Hip-Hop | rock |
| 0.0394 | 0.0308 | 0.0299 | 0.0269 | 0.0268 | 0.0248 | 0.0227 |

**Table 4**: Top tags for a few mostly "loaded" components (column-wise). The last row is the normalized total responsibility $\tilde{N}_k = \frac{N_k}{\sum_{k'}^{K} N_{k'}}$, meaning the proportional amount of songs having the component.




(a) Effect of regularization factor $\eta_0$. ($|X| = 213,354$)




(b) Effect of the number of training samples. ($\eta_0 = 10^{-1}$)

**Figure 3**: Effect of the learning factors on HDPGMM. Error bars indicate the standard deviations, and the grey dots are the raw data points.

Model comparison suggests that the HDPGMM model is comparable to the DL representation on average and can outperform it in a few specific scenarios. For the comparison, we chose the best HDPGMM model according to the result in Section 4.1 by setting $\eta_0 = 10^{-1}$ and using the entire dataset. As reported in Table 3, KIM achieves worse performance than HDPGMM on GTZAN and MTAT while performing better at recommendation (Echonest). On the other hand, CLMR, which adopts a more sophisticated learning strategy, achieves the best performance in Echonest, but performs worse than HDPGMM on MTAT and comparable on GTZAN. It is notable that the mixture model variants (VQCodebook, HDPGMM) perform particularly worse on the recommendation task. We hypothesize that the cosine similarity might not be an optimal choice for the probability simplex representation, or the feature set we consider may lack aspects that are crucial for the recommendation; this requires further study.

Except for this case, HDPGMM outperforms the other non-DL comparisons in general. It especially achieves significant improvement over its simplified version (VQCodebook) for all tasks, and mostly outperforms G1.

### 4.3 Component Interpretability

Finally, we explore the interpretability aspect of the HDPGMM model. To achieve it, we employ the *Lastfm-MSD* music tag dataset [30] where we find the mapping between the MSD songs and the social music tags. All our MSD training samples have mappings to the social tags, whose vocabulary size reaches roughly half a million. We compute the most relevant music tags for each corpus-level component found by the HDPGMM model. We estimate the relevance by a proxy measure $\alpha_{tk} = \sum_{j \in X: t \in j} N_j^{-1} \sum_n r_{jnk}$, where $t$ denotes the tag index and $r_{jnk}$ refers the responsibility computed in Eq. ( 20). To improve the clarity, we filter the most popular tags by weighting the measure by the Inverse Document Frequency (IDF) for each tag.

Table 4 shows an example from one of the HDPGMM with high regularization ($\eta_0 = 10^{-1}$). It suggests that the most frequent component (the first column) can be interpreted as the corpus-wide, universal topic. From second to the rest it represents a few other topics of music, such as *country-rock (column 1)*, *pop-female vocalist (column 2, 3, 5)*, and *electronic-dance (column 4)*. Notably, several compatible topics (i.e., columns 2, 3, and 5) exist, which can be interpreted as the collection of slightly different sub-clusters of an umbrella topic. It suggests that the advanced BN methods such as the hierarchical latent component models using the nested HDP [49] would further improve the representation.

## 5. CONCLUSION AND FUTURE WORK

In this work, we explore the potential of a BN model in the MIR task space. The results suggest that the HDPGMM representation achieves comparable effectiveness over the DL equivalents and outperforms a few scenarios. It implies that the method can be as effective as DL models when used in the supervised learning setup as well, by, for instance, jointly maximizing the variational lower bound both for the data generation and the prediction tasks. It has already been shown that such a joint objective approach can outperform the separated feature learning and transfer strategy that we demonstrate in this work [17].

Notably, HDPGMM is one of the more simple models in the BN approach. As mentioned above, there are models that are "deeper" such as nested DP models, which allow for a hierarchical structure of latent components [49]. The model also can be improved by introducing a sequential model such as Hidden Markov Models (HMM) as a base distribution [11], as the HDPGMM model assumes the exchangeability of tokens (feature frames), which is not ideal for a music signal. We left these possibilities to future work.

## 6. ACKNOWLEDGEMENT

A part of this work was carried out on the Dutch national e-infrastructure with the support of the SURF Coorperative.

## 7. REFERENCES

[1] M. Won, J. Spijkervet, and K. Choi, *Music Classification: Beyond Supervised Learning, Towards Real-world Applications*. https://music-classification.github.io/tutorial, 2021. [Online]. Available: https://music-classification.github.io/tutorial

[2] J. Briot, G. Hadjeres, and F. Pachet, *Deep Learning Techniques for Music Generation*, ser. Computational Synthesis and Creative Systems. Springer International Publishing, 2019.

[3] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019.

[4] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds. FEUP Edições, 2012, pp. 403–408.

[5] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *5th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2018, Turin, Italy, October 1-3, 2018*, F. Bonchi, F. J. Provost, T. Eliassi-Rad, W. Wang, C. Cattuto, and R. Ghani, Eds. IEEE, 2018, pp. 80–89.

[6] B. L. Sturm, "A simple method to determine if a music information retrieval system is a "horse"," *IEEE Trans. Multim.*, vol. 16, no. 6, pp. 1636–1644, 2014.

[7] ——, "The "horse" inside: Seeking causes behind the behaviors of music content analysis systems," *Comput. Entertain.*, vol. 14, no. 2, pp. 3:1–3:32, 2016.

[8] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[9] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, 2019.

[10] Y. W. Teh, "Dirichlet process," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Springer, 2017, pp. 361–370.

[11] Y. Qi, J. W. Paisley, and L. Carin, "Dirichlet process HMM mixture models with application to music analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, Honolulu, Hawaii, USA, April 15-20, 2007*. IEEE, 2007, pp. 465–468.

[12] M. D. Hoffman, D. M. Blei, and P. R. Cook, "Content-based musical similarity computation using the hierarchical dirichlet process," in *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 349–354.

[13] ——, "Finding latent sources in recorded music with a shift-invariant hdp," in *in International Conference on Digital Audio Effects (DAFx) (under review)*, 2009.

[14] M. Nakano, J. L. Roux, H. Kameoka, N. Ono, and S. Sagayama, "Infinite-state spectrum model for music signal analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. IEEE, 2011, pp. 1972–1975.

[15] M. Nakano, Y. Ohishi, H. Kameoka, R. Mukai, and K. Kashino, "Bayesian nonparametric music parser," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*. IEEE, 2012, pp. 461–464.

[16] K. Yoshii and M. Goto, "Continuous plsi and smoothing techniques for hybrid music recommendation," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 339–344.

[17] J. Wang, Y. Lee, Y. Chin, Y. Chen, and W. Hsieh, "Hierarchical dirichlet process mixture model for music emotion recognition," *IEEE Trans. Affect. Comput.*, vol. 6, no. 3, pp. 261–271, 2015.

[18] C. Wang, J. W. Paisley, and D. M. Blei, "Online variational inference for the hierarchical dirichlet process," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, ser. JMLR Proceedings, G. J. Gordon, D. B. Dunson, and M. Dudík, Eds., vol. 15. JMLR.org, 2011, pp. 752–760.

[19] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[20] J. Kim, J. Urbano, C. C. S. Liem, and A. Hanjalic, "One deep music representation to rule them all? A comparative analysis of different representation learning strategies," *Neural Comput. Appl.*, vol. 32, no. 4, pp. 1067–1093, 2020.

[21] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 141–149.

[22] S. Dieleman, P. Brakel, and B. Schrauwen, "Audio-based music classification with a pretrained convolutional network," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 669–674.

[23] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, January 2006.

[24] J. Sethuraman, "A constructive definition of Dirichlet priors," *Statistica Sinica*, vol. 4, pp. 639–650, 1994.

[25] J. Pitman, "Poisson-dirichlet and gem invariant distributions for split-and-merge transformations of an interval partition," *Comb. Probab. Comput.*, vol. 11, no. 5, pp. 501–514, 2002.

[26] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, 1999.

[27] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Analysis*, vol. 1, no. 1, pp. 121 – 143, 2006.

[28] M. D. Hoffman, D. M. Blei, and F. R. Bach, "Online learning for latent dirichlet allocation," in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 856–864.

[29] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 673–681.

[30] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[31] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018, pp. 637–644.

[32] J. Lee, J. Park, K. L. Kim, and J. Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, 2018.

[33] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, 2002.

[34] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning and music adversaries," *IEEE Trans. Multim.*, vol. 17, no. 11, pp. 2059–2071, 2015.

[35] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 387–392.

[36] J. Lee and J. Nam, "Multi-Level and Multi-Scale Feature Aggregation Using Pretrained Convolutional Neural Networks for Music Auto-Tagging," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1208–1212, Aug. 2017.

[37] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds. ACM, 2018, pp. 689–698.

[38] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[39] M. Deshpande and G. Karypis, "Item-based top-$N$ recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004.

[40] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[41] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979.

[42] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[43] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, and et al., "librosa/librosa: 0.9.1," Feb 2022.

[44] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, September 2013.

[45] D. P. Ellis, Apr 2007, accessed: 2022-05-12. [Online]. Available: https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[47] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 1597–1607.

[48] J. Nam, J. Herrera, M. Slaney, and J. O. S. III, "Learning sparse feature representations for music annotation and retrieval," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds. FEUP Edições, 2012, pp. 565–570.

[49] J. W. Paisley, C. Wang, D. M. Blei, and M. I. Jordan, "Nested hierarchical dirichlet processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 256–270, 2015.

# POP MUSIC GENERATION WITH CONTROLLABLE PHRASE LENGTHS

**Daiki Naruse**[1] **Tomoyuki Takahata**[1] **Yusuke Mukuta**[1,2] **Tatsuya Harada**[1,2]
[1] The University of Tokyo, Japan
[2] RIKEN, Japan
{naruse, takahata, mukuta, harada}@mi.t.u-tokyo.ac.jp

## ABSTRACT

Research on music generation using deep learning has attracted more attention; in particular, Transformer-based models have succeeded in generating coherent musical pieces. Recently, an increasing number of studies have focused on phrases that are smaller musical units, and several studies have addressed phrase-level control. In this study, we propose a method for sequentially generating a piece that enables the control of each phrase length and, consequently, the length of the entire piece. We added PHRASE and a new event, BAR COUNTDOWN, which indicates the number of bars remaining in the phrase, to the existing event-based music representations. To reflect user input indicating the phrase lengths of the piece being generated, we used an autoregressive generation model that adds these two events to the generated event-token sequence based on the user input and uses it as input for the next time step. Subjective listening tests revealed that the pieces generated by our methods possessed designated phrase lengths and ended naturally at the determined length. [1]

## 1. INTRODUCTION

Music generation has been studied for more than half a century [1] and has advanced significantly in recent years with the development of deep learning. In many studies, deep neural sequence models such as recurrent neural networks (RNNs) and Transformers [2] have been used to model music. Transformer-based methods [3–7] have succeeded in generating coherent music throughout a piece. To apply these sequence models to music generation, it is necessary to represent a piece as a sequence of tokens. Event-based representations such as MIDI-like [8] and its advanced versions, REMI [6] and CP [7], have been used.

More recently, an increasing number of studies have focused on phrases and sections [9–14], which are smaller musical segments. These studies aimed to generate a structured piece that was divided into several segments and developed through repetition and transformation. Several studies addressed phrase-level control [11–13] and allowed the control of phrase attributes such as melody, rhythm, and harmonic fullness. Length is another important phrase attribute and is controllable with a phrase-by-phrase generation policy [13], which means that each phrase is generated independently and joined. However, this phrase-by-phrase generation policy has a limitation in that natural transitions between phrases are not guaranteed.

Therefore, we worked on controlling the phrase lengths with a sequential generation policy. The sequential generation policy, unlike the phrase-by-phrase or section-by-section generation policy [9, 13], is a method of sequentially generating an entire piece at once. We aim to create a model that outputs a piece according to user input regarding the configuration of the phrases and the length of each phrase, as shown in Figure 1 (the detailed generation process is described in Sections 3.3 and 3.4). The controllability of each phrase length implies that the length of the entire piece can be controlled. To control the length of each phrase and the entire piece, we extended two recently used event-based music representations, REMI [6] and CP [7]. The random timing of the switching of phrases and the end of the generation in the existing representations is likely due to the model not knowing which phrase and where it is generating. Therefore, we added PHRASE and a new event, BAR COUNTDOWN, which indicates the number of bars remaining in the phrase, to REMI and CP. To reflect the user input, we used an autoregressive generation method in which these two events were added based on the user input to the generated event-token sequence, and the sequence was entered into the model again. To evaluate this approach, two subjective listening tests were conducted for the length of each phrase and the entire piece. By comparing our methods with the dataset and existing methods, we demonstrate our methods are effective in length control.

Our contributions are summarized as follows:

- We extended the existing music representations (REMI [6] and CP [7]) by adding PHRASE and BAR COUNTDOWN events and showed that both events are necessary for length control.
- We enabled the reflection of the user input by an autoregressive generative model that adds the PHRASE and BAR COUNTDOWN events to the generated event-token sequence based on the user input and uses it as input for the next time step.



[1] Samples of the generated pieces are available at https://mil-tokyo.github.io/phrase-length-designated-music-generation/.

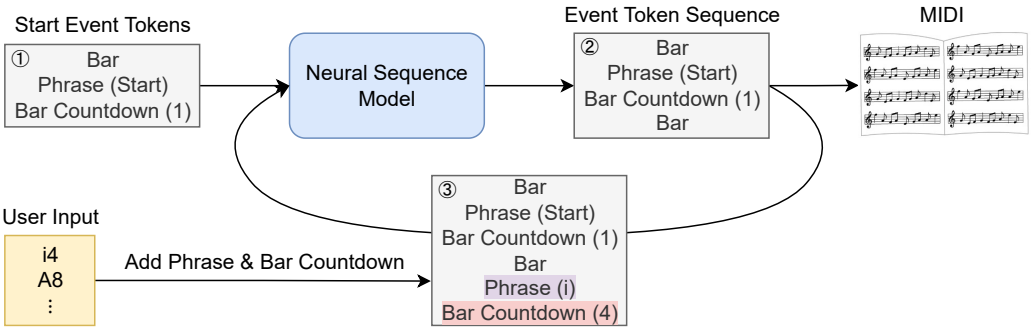



**Figure 1**: Generation process reflecting user input regarding phrase lengths.

## 2. RELATED WORK

### 2.1 Event-based Music Representations

To apply neural sequence models to music generation, music must be represented using a token sequence. Many studies [3–5, 15] adopted MIDI-like [8]. NOTE ON and NOTE OFF events indicate the start and end of a note, respectively, and a TIME SHIFT event advances the time step.

In MIDI-like, bars and beats are implicit, which are clearly indicated in the score, and it is difficult for the model to learn beat regularity and rhythmic structure. The model also has difficulty learning that the NOTE ON and NOTE OFF events must exist in pairs. To address these problems, an improved music representation called REMI (revamped MIDI-derived events) [6] was proposed. In REMI, the TIME SHIFT event in MIDI-like is replaced with BAR and BEAT events, and the NOTE OFF event is replaced with a NOTE DURATION event. TEMPO and CHORD events are added for clear harmony and expressive rhythmic freedom.

Later, a further extension of REMI called CP (compound word representation) [7] was suggested. In CP, consecutive and related events are grouped and placed in the same time step. Specifically, BAR, BEAT, CHORD, and TEMPO events are grouped into a METRICAL family and note-related events into a NOTE family. Additionally, a new event, EOS, is added to mark the end of a piece.

### 2.2 Latest Transformer-based Music Generation

Recently, Transformer-based methods have successfully generated coherent music throughout a piece and have become common in automatic composition in the symbolic domain. The Music Transformer study [3] was the first to apply the Transformer model to music generation. This study used MIDI-like to generate pieces by autoregressively predicting an event token at each time step. The Pop Music Transformer study [6] proposed REMI and generated pieces with a better rhythmic structure. The Jazz Transformer study [16] addressed the generation of Jazz. An attempt was made to introduce structure by adding the following four structure-related events to REMI: PHRASE, MLU, PART, and REPETITION. The CP Transformer study [7] proposed CP. Predicting events of the same family simultaneously at each time step significantly reduces the length of the token sequence, resulting in faster learning and inference. In addition, the EOS event allowed the model to complete the generation with the natural closure.

HAT (Harmony-Aware Hierarchical Music Transformer) [14] focused on phrases and sections. It represents music in CP and uses three Transformers hierarchically to allow event tokens to interact at different levels. The structure of the pieces was improved by learning the texture and the form jointly bridged by the harmony.

MusicFrameworks [13] is a monophonic melody generation system that enables phrase-level control. Music is described using music frameworks, a hierarchical music structure representation, and melodies are generated through a multi-level generative process. The manipulation of music frameworks allows control over phrase attributes such as structure, melody, and rhythm. Phrase length can also be controlled by modifying the structural information and length of the rhythm and chord information in each phrase. However, because of the phrase-by-phrase generation policy, there is no guarantee that transitions between phrases are natural.

## 3. METHOD

### 3.1 Phrase-Length Designated Music Generation

In this study, we worked on an automatic composition task that allowed control over the length of each phrase (and the length of the entire piece) with a sequential generation policy. The user inputs the configuration of the phrases and the length of each phrase in units of bars, and a piece with the designated phrase lengths and total length is generated. For example, if the input is "i4 A8 B8 o4," then a piece is generated with four bars for the intro phrase, eight bars for phrase A, eight bars for phrase B, and four bars for the outro phrase, with a total length of 24 bars.

### 3.2 PHRASE and BAR COUNTDOWN Events

We added the following two events to REMI [6] and CP [7]: PHRASE and BAR COUNTDOWN.

The first one, PHRASE, is an event that indicates to which phrase a bar belongs, e.g., PHRASE (i) refers to the intro phrase. This event was first proposed in the Jazz Transformer study [16]. In our study, PHRASE (Start) and PHRASE (End) were used in addition to phrase labels in the