Improving Automatic Jazz Melody Generation by Transfer Learning Techniques

Hsiao-Tzu Hung*[†], Chung-Yang Wang[†] Yi-Hsuan Yang[†‡], Hsin-Min Wang*
* Institute of Information Science, Academia Sinica, Taipei, Taiwan
[†] Taiwan AI Labs, Taipei, Taiwan
[‡] Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan
E-mail: {fbiannahung,wangyygogo}@gmail.com, yhyang@ailabs.tw, whm@iis.sinica.edu.tw

Abstract—In this paper, we tackle the problem of transfer learning for Jazz automatic generation. Jazz is one of representative types of music, but the lack of Jazz data in the MIDI format hinders the construction of a generative model for Jazz. Transfer learning is an approach aiming to solve the problem of data insufficiency, so as to transfer the common feature from one domain to another. In view of its success in other machine learning problems, we investigate whether, and how much, it can help improve automatic music generation for under-resourced musical genres. Specifically, we use a recurrent variational autoencoder as the generative model, and use a genre-unspecified dataset as the source dataset and a Jazz-only dataset as the target dataset. Two transfer learning methods are evaluated using six levels of source-to-target data ratios. The first method is to train the model on the source dataset, and then fine-tune the resulting model parameters on the target dataset. The second method is to train the model on both the source and target datasets at the same time, but add genre labels to the latent vectors and use a genre classifier to improve Jazz generation. Our subjective evaluation shows that both methods outperform the baseline method that uses Jazz data only for training by a large margin. Among the two methods, the first method seems to perform better. Our evaluation also shows the limits of existing objective metrics in evaluating the performance of music generation models.

I. INTRODUCTION

Deep learning-based machine learning algorithms have been increasingly employed for automatic music composition in recent years [1]–[3]. Typically, this is done by collecting a large dataset of machine-readable musical scores of existing music, in formats such as MIDI files,[1] and then using neural network models, such as the generative adversarial model (GAN) [4] and variational autoencoder (VAE) [5], to learn to compose new music via learning from the provided dataset. For example, the Lakh Pianoroll Dataset (LPD) is a public-domain dataset compiled by Dong *et al.* for building a GAN model for multitrack music composition [6]; it encompasses 50,266 four-bar MIDI phrases of Rock/Pop music in 4/4 time signature. As another example, Roberts *et al.* [7] attempted to collect a large number of MIDI files from the Web to train a VAE model for generating melodies (monophonic note sequences) for unspecified musical genres; it is said that around 1.5 million unique MIDI files were found and downloaded.

TABLE I
THE PERCENTAGE OF MELODY LABELED WITH DIFFERENT GENRE TAGS IN THE THEORYTAB (TT) DATASET. IT CONTAINS 11,329 MELODIES, AND IS USED AS THE "SOURCE DOMAIN" DATASET IN THIS WORK.

Jazz	Folk	Dance	Electronic	Rock	Pop	Unlabeled
2.12%	2.12%	6.23%	10.70%	9.04%	11.25%	58.54%

The main advantage of such deep learning-based models for automatic music composition, compared to the rule-based or genetic algorithm-based algorithms studied by researchers decades ago [8], appears to be their unprecedented ability to find their own ways in learning from big data. We have already seen from the literature promising examples that use deep learning to learn to compose music for musical genres such as Rock [6], Pop [9], and Classical music [11]. Common to these genres is the availability of MIDI files from the Web, providing sufficient data to train deep learning models.

However, this is not the case with many main musical genres in the world. An obvious example is Jazz, which often features live improvisations (i.e., with spontaneously invented melodic solo lines or accompaniment parts). In other words, a complete Jazz music piece is rarely composed *offline* with a MIDI editor; rather, Jazz is usually created *online* with spontaneous interaction among musicians. Extra effort is required to listen to the audio recording of a Jazz performance and carefully transcribe it by hand into a MIDI file. Consequently, MIDI files for Jazz music are relatively scarce on the Web.

To illustrate this, we wrote a crawler to download a total of 11,329 melody phrases from an online music theory forum called TheoryTab.[2] As shown in Table I, only 2.12% of the melodies were labeled as Jazz by the contributing forum users. Pop and Rock, for example, have around five times more data.

To our best knowledge, little work has been done recently on building automatic Jazz composition. One prominent prior work is the work by Trieu and Keller [3], who employed GAN to build a model called JazzGAN for chord-conditioned melody composition. However, likely due to the reasons outlined above, the dataset they used to train JazzGAN contained only 44 leadsheets, approximately 1,700 bars.

In the machine learning community, many "transfer learn-

---

[1][Online] https://www.midi.org/

[2]TheoryTab is hosted by Hooktheory, a company that produces educational music software and books ([Online] https://www.hooktheory.com/theorytab).

ing" techniques have been proposed to address the data scarcity of target tasks [12]–[15]. The idea is to find a related *source task* where the training data is easier to collect, and then adapt the model of the source task to the model of the *target task* with a small dataset in the target domain. Given the relative richness of non-Jazz MIDI data, a natural research question is whether, and how much, we can leverage a large genre-unspecified *source domain* MIDI dataset to improve the model for Jazz with a small genre-specific *target domain* MIDI dataset.

In this paper, we aim to address the following research questions.

- Can we use a genre-unspecified music dataset to improve a Jazz melody generation model?
- Which transfer learning technique is more useful for this task?
- Does a transfer learning method benefit from increasing the size of the source domain data?

For the second research question, we evaluate two canonical transfer learning methods in this work: model fine-tuning and multitask learning (see Section IV). For the third research question, we consider six levels of source-to-target data ratios (see Section V-A). For performance evaluation, we follow the recent work of Yang and Lerch [17] and adopt seven different criteria for quantitative evaluation (see Section V-B).

While a piece of Jazz music can be composed of multiple tracks/instruments, we only focus on the melody in this work. In addition, we consider the task of "unconditioned" Jazz melody generation, i.e., generating melodies without any pre-determined conditions or information. This scenario is more challenging, yet practically more flexible and potentially more useful, than the "chord-conditioned" scenario addressed by JazzGAN [3], where a sequence of accompanying chord labels is given to inform the melody generation model.

Certainly, Jazz is not the only "under-resourced" [16] genre in music. Since our problem formulation is generic, it is hoped that the lessons learned here can also be applied to other musical genres. In addition, the problem may be interesting for general machine learning researchers as well, as transfer learning is more commonly employed for discriminative tasks such as classification and regression, rather than generative tasks such as automatic generation.

The paper is organized as follows. Section II provides background knowledge on transfer learning. Sections III and IV present the datasets and models used in this work. Section V details on the evaluation setup. Section VI discusses the evaluation results. Finally, Section VII concludes the paper.

II. BACKGROUND

A. Transfer Learning

The general idea of transfer learning is to learn knowledge from one task and apply it to another task. Generally, there will be two tasks A and B. Both tasks have the same input type, such as image and audio. Our main task is B, but the dataset for task B is much more smaller than the dataset for

TABLE II
THE TWO DATASETS USED IN THIS WORK. A PHRASE IS DEFINED AS A
FOUR-BAR SEGMENT SAMPLED FROM A SONG.

	TT (source)	CY+R (target)
Genre	diverse	Jazz only
Song length	segment	segment
Track	melody, chord	melody
Musical key	C major, C minor	C major
Time signature	4/4	4/4
Number of phrases	9,640	1,608
Number of bars	38,560	6,432

A, which may not be enough for training. Assuming that tasks A and B share some low-level features, we can improve task B by learning task A first. There are two training steps for transfer learning. The first step is to train the model on the dataset of task A, which is often referred to as "pre-training." The second step is to further train the model obtained in the first step on the dataset of task B, which is called "fine-tuning." There are many transfer learning methods based on such a pre-training/fine-tuning strategy. An overview of recent techniques can be found in [20].

In recent years, transfer learning based on the above strategy has been widely used in many machine learning problems in computer vision (CV) and natural language processing (NLP). Well-known examples include the use of the first few layers of deep models trained on the ImageNet object recognition task as visual feature extractors for other CV tasks [14] and the use of Google's pre-trained BERT model to get word and sentence embedding features for downstream NLP tasks [15].

B. Transfer Learning in Music-related Tasks

Transfer learning techniques have also been applied to several discrminative tasks in the field of music information retrieval (MIR). For example, in [19], a convnet was trained for music tagging, and then transferred to other music-related classification and regression tasks. The tags of the source task include genres, instruments, moods, and eras. The target tasks include vocal/non-vocal classification and general audio event classification. There are many other examples, all of which are concerned with classification or regression tasks [21]–[24].

To our best knowledge, transfer learning techniques have not been used for automatic music composition. Researchers either work on genres with easy-to-access MIDI data (e.g., Rock and Pop) [6] or a general model using genre-unspecified MIDI data [7].

III. DATASETS

For this study, we have collected a clean Jazz-only dataset as the target dataset, and a genre-unspecified dataset as the source dataset. The two datasets are summarized in Table II.

The target dataset, referred to as the CY+R dataset here-after, consists of two small Jazz music collections. The first collection consists of 575 four-bar melody phrases composed by one of the authors, who is a well-trained musician. All
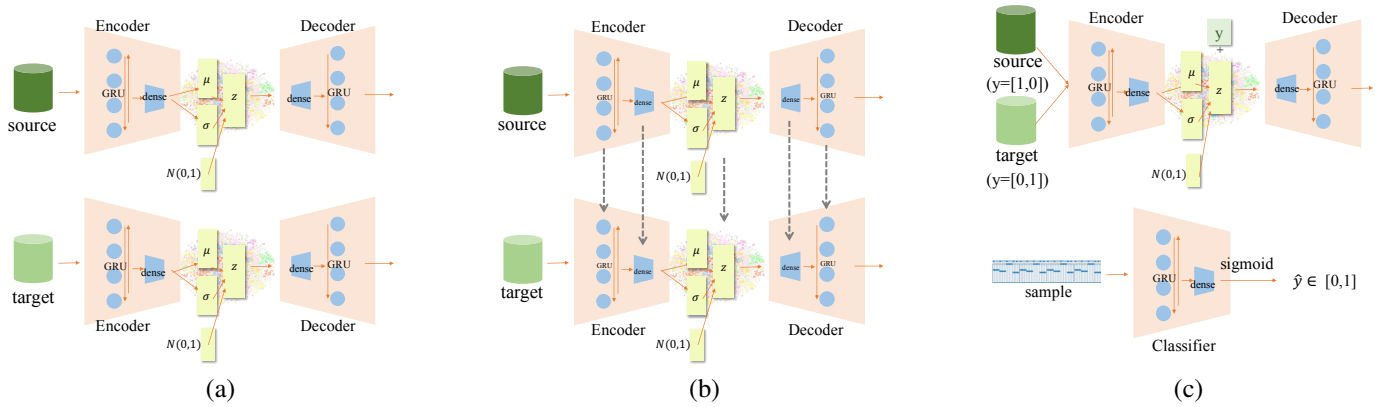
Fig. 1. Model architectures and training flows of the methods evaluated in this paper. (a) The baseline methods trained only on the source dataset (upper) or the target dataset (lower). (b) The "fine-tuning" method that pre-trains a model using the source dataset (upper) and then fine-tunes the model using the target dataset (lower). (c) The "multi-task" method that introduces an additional genre label y as a conditional variable so as to train the model together on the source and target datasets (upper), with a separately trained genre classifier for further improving Jazz generation (lower).

phrases are soft Jazz music. The second collection comes from the Jazz Realbook,[3] which contains 240 unique songs.

The musician manually split out a few four-bar phrases from each song, ensuring that each phrase has a musically plausible ending. Finally, there are 1,608 four-bar phrases in CY+R, 1,446 phrases for training, and 162 phrases for testing.

The source dataset contains 11,329 melody phrases downloaded from TheoryTab.[4] As shown in Table I, more than 50% of the data are genre-unspecified. Although a small portion of them were labeled as Jazz, we checked the songs and found that some of the labels were unreliable. For example, "Hard To Say I'm Sorry" by Chicago was wrongly labeled as Jazz. In addition, most Jazz songs were style-wise quite different from our target dataset. Therefore, we ignored the genre labels and treated all the melodies as genre-unspecified. We use 'TT' as the abbreviation for the TheoryTab dataset. Each song in TT is saved separately, including intro, chorus, verse, and outro. The intros of most songs in TT are composed of broken chords, which means repeating some notes from the chords, and they may not be considered as melodies. Therefore, the intros are excluded from the TT dataset, leaving 9,640 four-bar phrases. The ratio of training data to testing data is 9:1, i.e., 90% for training, and 10% for testing.

Table II summarizes the two datatsets. Both datasets are in 4/4 time signature. The CY+R dataset is written in C-major scale, and each melody phrase is transposed to the C major or C minor key in TT dataset. In order to broaden the diversity of generated melodies, we keep both C major and C minor songs in the TT dataset.

A. Representation of a melody phrase

There are multiple ways to computationally represent a melody phrase. For example, Trieu and Keller investigated and compared the so-called "event-based" and "time step-based"

representations of melody [3]. In this work, we adopt the time step-based method and represent each four-bar melody phrase as a fixed-size matrix.[5] This matrix-like representation has also been referred to as the *pianoroll* representation [25], where the horizontal axis denotes time (time step), and the vertical axis denotes frequency (MIDI note). For each bar, we set the height of the matrix to 48 (considering MIDI notes from C3 to B6) and the width (time resolution) to 16 (i.e., 16 time steps per bar, or equivalently 4 time steps per beat in 4/4 time signature). As a result, the size of the target output tensor for melody generation is 4 (bars) × 16 (time steps) × 48 (MIDI notes) × 1 (track).

IV. METHODOLOGY

A. Model Architecture

Following [26], we adopt a recurrent VAE model here. In the encoder part, four-bar melody sequences are fed into bidirectional gated recurrent units (BGRU) to learn the correlation between bars. The outputs of all GRU time steps are then concatenated and passed through several dense (i.e., fully-connected) layers to get the embedding vector. In other words, given an observed input melody $\mathbf{x}$, the encoder $\mathbb{E}_\theta$ with parameter set θ encodes $\mathbf{x}$ into a latent vector $\mathbf{z} = \mathbb{E}_\theta(\mathbf{x})$.

In the decoder part, a latent vector $\mathbf{z}$ is sampled from a normal distribution characterized by μ and σ, and then passed through several fully-connected layers parameterized by ϕ to separately form the initial states of melody. The outputs are processed by a unidirectional GRU with a sigmoid activation layer to finally output an four-bar pianoroll. This model is illustrated in Fig. 1(a), either the upper or lower panel.

B. Method 1: Fine-tuning

As with the basic process in transfer learning, we can train the model in two stages, as illustrated in Fig. 1(b). First, we pre-train the recurrent VAE model with TT. In this stage, the

---

[3][Online]https://www.profesordepiano.com/Real%20Book/Realbook.htm

[4]Note that the data scraped from TheoryTab is in the so-called lead sheet format, containing both the melody track and the chord track. We ignore the chord track in this work, since we consider unconditioned melody generation.

[5]One advantage of the time step-based representation over the event-based representation is that we can more easily emphasize the beat position of notes.

goal of training is to let the model learn *what melody is*. Here, the learning rate is set to $1e^{-3}$, and the ADAM optimization algorithm is used to accelerate stochastic gradient descent. As for the objective function, we use the classic binary cross-entropy and KullbackLeibler divergence (KLD) losses. The model parameters are obtained by maximizing the following variational lower bound:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathcal{L}_{\text{recon}}(\mathbf{x}) + \mathcal{L}_{\text{lat}}(\mathbf{x}), \quad (1)$$

where $\mathcal{L}_{\text{recon}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$ is the reconstruction term, and $\mathcal{L}_{\text{lat}} = -D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))$ regularizes the encoder to align the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ with the prior distribution $p(\mathbf{z})$. $p_\theta(\mathbf{x} \mid \mathbf{z})$ is the data likelihood.

In the second stage, we fine-tune the model using the Jazz dataset CY+R. The goal is to let the model learn *what Jazz is*. The learning rate in this stage is set to $1e^{-5}$ for $t < 40$, $1e^{-7}$ for $40 \leq t < 80$, and $1e^{-9}$ for $t \geq 80$, where t denotes the epoch. Other parameters remain the same.

C. Method 2: Multitask Learning

Here, we additionally concatenate a one-hot genre label y to the latent vector $\mathbf{z}$, as shown in Fig. 1(c). We train the model on both TT and CY+R at the same time and regard them as two different tasks for the model to work on. For the Jazz dataset CY+R, we set the label $y = [0, 1]$; for TT, we set the label $y = [1, 0]$.

To evaluate whether our model learns to generate two different types of melodies, we pre-train a genre classifier $C(\cdot)$ to see if the output melody has the Jazz elements. The classifier basically has the same structure of the VAE encoder, but changes the output size of the final fully-connected layer to 1. When a generated melody passes through the genre classifier, the classifier outputs the probability of Jazz. We apply sigmoid activation to the output neuron of the last layer, and optimize the classifier using a cross-entropy loss. The training goal is to output 1 for Jazz and 0 for non-Jazz. As a result, after the output melody is generated by our VAE model, it will be passed through the classifier and a probability $\hat{y} = C(\mathbf{x})$ will be obtained.

We define the genre prediction loss as $L_{\text{genre}}(\hat{y}, y)$ and add it to the objective function of VAE. Accordingly, the recurrent VAE model trained under the multitask learning based transfer learning method is optimized with the following objective function:

$$\mathcal{L}(\theta, \phi; \mathbf{x}, y) = \mathcal{L}_{\text{recon}}(\mathbf{x}, y) + \mathcal{L}_{\text{lat}}(\mathbf{x}) + L_{\text{genre}}(\hat{y}, y) \quad (2)$$

Please note that, unlike the case in Eq. (1), here the reconstruction loss L_{recon} additionally consider the provided genre label y.

V. EVALUATION SETUP

A. Evaluated Models

As our goal is to compare the effectiveness of the fine-tuning method and the multi-task learning method, we implement both of them in the evaluation. In order to examine how the transfer learning methods benefit from increasing the size of the source domain training data, we consider six levels of source-to-target data ratios (R):

$$R = \frac{\text{number of non-Jazz training phrases}}{\text{number of Jazz training phrases}}, \quad (3)$$

where $R \in \{1, 2, ..., 6\}$. Moreover, as depicted in Fig. 1(a), we implement two baseline models. The first model is trained on the small, Jazz-only CY+R dataset; this can be considered as the case when $R = 0$. This model may not even learn *what melody is* because the training set is really small. The second baseline model is trained only on the large, genre-unspecified TT dataset; this can be considered as the case when $R = \infty$. This model may not learn *what Jazz is* because the training set contains music of arbitrary genres.

B. Feature Metrics

In order to evaluate the quality of generated melody, some features are extracted based on the work of Yang and Lerch [17]. The features describe two aspects of music, including pitch- and rhythm-related ones.

The pitch-related features, including the following four, describe the preferences for arranging pitch:

- **Pitch count (PC)**: The pitch count is the number of unique pitches within a phrase. The output is a scalar for each phrase.
- **Pitch class histogram (PCH)**: The pitch class histogram is a 12-dimensional, octave-independent representation of the pitch content for achromatic scale [27].
- **Pitch class transition matrix (PCTM)**: The transition of pitch classes contains useful information for tasks such as key detection, chord recognition, and genre recognition [17]. The two-dimensional pitch class transition matrix is a histogram-like representation computed by counting the pitch transitions for each (ordered) pair of notes. The resulting matrix size is 12×12.
- **Pitch range (PR)**: The pitch range is calculated as the difference between the highest and lowest MIDI pitches in semitones within a phrase. The output is a scalar for each phrase.

The rhythm-related features, encompassing the following three, describe how the notes are arranged:

- **Note count (NC)**: The note count is the number of notes within a phrase. As opposed to the pitch count, the note count does not contain pitch information, but a rhythm-related feature that records only how many notes are in the phrase. The output is a scalar for each phrase.
- **Note length histogram (NLH)**: To extract the note length histogram, we define a set of allowable beat length classes [full, half, quarter, 8th, 16th, dot half, dot quarter, dot 8th, dot 16th, half note triplet, quarter note triplet, 8th note triplet]. The length of a bar is defined to contain 96 unit lengths, and each note length is quantized to the nearest number of unit lengths. The rest option, when activated, will double the vector size to represent the same length classes for rests. The output vector has a length of

TABLE III
OVERLAPPING AREA (OA) BETWEEN THE TRAINING MELODIES AND THE
MELODIES GENERATED BY METHOD 1.

	Method 1: fine-tuning					
	R = 1	R = 2	R = 3	R = 4	R = 5	R = 6
NC	0.7997	0.7941	**0.8385**	0.7690	0.7713	0.7677
NC/bar	0.8006	**0.8380**	0.8280	0.8005	0.7939	0.7963
NLH	0.7286	0.7185	**0.7485**	0.7203	0.7074	0.6976
NLTM	0.9158	0.8850	**0.9179**	0.8855	0.8869	0.8856
PC	0.6387	0.5986	**0.6859**	0.6515	0.6690	0.6770
PC/bar	0.8106	0.8123	**0.8510**	0.7833	0.7919	0.7851
PR	0.6676	0.6436	**0.7134**	0.6956	0.6824	0.7063
PCH	0.3198	0.3029	**0.3733**	0.3424	0.3679	0.3545
PCTM	0.6091	0.6227	0.6113	0.6918	0.7120	**0.7355**
average	0.6990	0.6906	**0.7298**	0.7044	0.7092	0.7117

TABLE IV
OVERLAPPING AREA (OA) BETWEEN THE TRAINING MELODIES AND THE
MELODIES GENERATED BY METHOD 2.

	Method 2 : multitask learning					
	R = 1	R = 2	R = 3	R = 4	R = 5	R = 6
NC	0.8536	**0.8724**	0.8412	0.7972	0.8002	0.8098
NC/bar	0.8449	0.8474	**0.8562**	0.8073	0.8218	0.8278
NLH	0.7661	**0.7713**	0.7631	0.7485	0.7341	0.7621
NLTM	**0.9292**	0.9166	0.9132	0.9055	0.8874	0.8631
PC	0.7157	**0.7434**	0.7349	0.7344	0.7146	0.7219
PC/bar	0.8582	0.8593	**0.8637**	0.8176	0.8157	0.8184
PR	0.7544	0.7261	0.7470	0.7299	0.7320	**0.7585**
PCH	0.3938	0.3862	0.3478	**0.4062**	0.2810	0.3991
PCTM	0.6670	0.6290	0.6575	0.6982	0.7142	**0.7416**
average	**0.7536**	0.7502	0.7472	0.7383	0.7223	0.7447

TABLE V
THE PERFORMANCES OF DIFFERENT METHODS.

	Baseline 1 (source)	Baseline 2 (target)	Method 1 (R=3)	Method 2 (R=1)
NC	0.7847	0.8287	0.8385	**0.8536**
NC/bar	0.8200	0.8393	0.8280	**0.8449**
NLH	0.6914	0.7407	0.7485	**0.7661**
NLTM	0.8520	0.9081	0.9179	**0.9292**
PC	0.6530	0.7039	0.6859	**0.7157**
PC/bar	0.8020	0.8544	0.8510	**0.8582**
PR	0.7455	0.7217	0.7134	**0.7544**
PCH	0.3997	**0.4531**	0.3733	0.3938
PCTM	**0.7432**	0.6909	0.6113	0.6670
average	0.7213	0.7490	0.7298	**0.7536**

either 12 (for notes) or 24 (12 for notes and 12 for rests), respectively.

- **Note length transition matrix (NLTM)**: Similar to PCTM, the note length transition matrix provides useful information for rhythm description. The matrix size is 12×12 or 24×24.

C. Overlapping Area (OA)

Yang and Lerch [17] proposed to use Overlapping Area (OA) as an evaluation measure. The rational is given below. To compare different output sets, relative measurements may be a better choice instead of using the mean of features directly. Through relative measurements, the diversity of the dataset can be obtained.

There are three steps in calculating the OA:

1. A pairwise exhaustive cross-validation is first performed for each feature. In each cross-validation step, the Euclidean distance of one sample to each of the other samples is computed. If the cross-validation is conducted on the samples in the same set, the intra-set distances are calculated. If we compare each sample in one set with all samples in another set, we calculate the inter-set distances. The output of the cross-validation process is a histogram of distances for each feature.

2. In order to smooth the histogram results for a more general representation, kernel density estimation [18] is applied to convert the histogram into a Probability Distribution Function (PDF).

3. After getting the PDFs of the target dataset and the generated melodies, OA is used to compare them. Since the melodies are generated under random sampling conditions of a Gaussian distribution, there is no overfitting problem.

The Kullback-Leibler Divergence (KLD) is commonly used to compare two distributions. However, since in discrete probability distributions, KLD is calculated in an element-wise manner, PDFs with an identical shape (as indicated by similar Kurtosis and Skewness) but shifted on the x-axis (distinct in the mean value) yield insignificant differences in KLD. In this case, OA is able to indicate the differences.

VI. EXPERIMENTAL RESULTS

In this section, we discuss the experimental results.The source codes for the experiments and the demo page are

available on GitHub.[6]

A. Overlapping Area

Tables III and IV show the OAs of two transfer learning methods evaluated on different features under six levels of source-to-target data ratios (R). The bold number indicates the highest OA for each feature under different Rs. For example, the highest OA of NC for Method 1 (fine-tuning) is 0.8385 when $R = 3$.

From Table III, we observe that $R = 3$ gives the best performance in most features and the best average performance for Method 1. In contrast, in Table IV, although the best performances for different features are quite divergent between different Rs, Method 2 (multitask learning) seems to perform better when R is smaller. The reason may be due to the imbalance of source and target training data. Method 2 is more affected by the imbalance of source and target training data because its model is trained on both the source and target training datasets at the same time. Overall, the results in Tables III and IV indicate that Method 2 (multitask learning) outperforms Method 1 (fine-tuning).

In Table V, we compare the performances of different methods, including Baseline 1 (the model is trained on the source training dataset), Baseline 2 (the model is trained on the small target training dataset), Method 1 (fine-tuning with $R = 3$), and Method 2 (multitask learning with $R = 1$). Several observations can be drawn from the table. First, Baseline 2, in which the model is trained on the small target

---

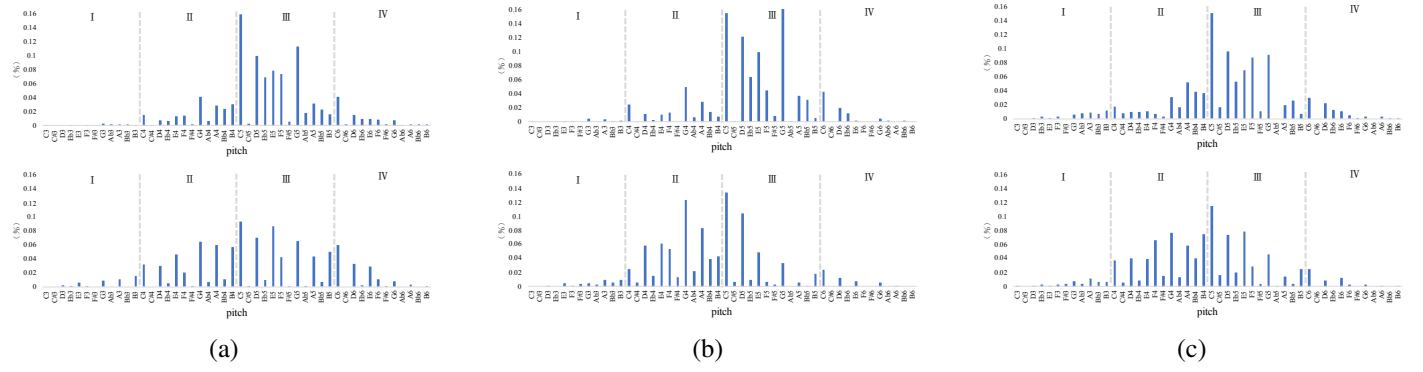[6] https://github.com/annahung31/jazz_melody_generation

Fig. 2. Pitch histograms of (a) the source (TT) dataset (upper) and the target (CY+R) dataset (lower), (b) the melodies generated by Baseline 1 trained on TT (upper) and Baseline 2 trained on CY+R (lower), and (c) the melodies generated by Method 1 with $R = 3$ (upper) and Method 2 with $R = 1$ (lower).
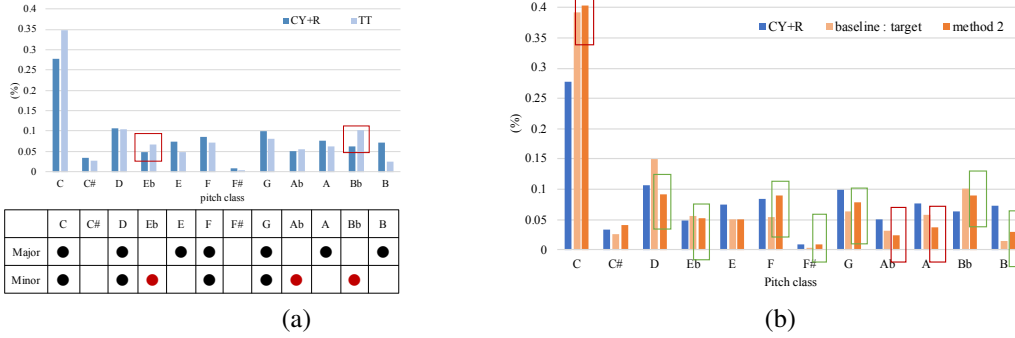

Fig. 3. (a) Upper: Pitch class histograms of the target (CY+R) and source (TT) datasets. Lower: the scales of C-Major and C-Minor; (b) Pitch class histograms of the target dataset and the melodies generated by Baseline 2 and Method 2 ($R = 1$).

dataset, outperforms Baseline 1 whose model is trained on the large source dataset. Second, Method 1 can improve the model trained on the source dataset by fine-tuning it with a small target dataset, but the improvement is not much. Third, surprisingly, Method 1 is worse than Baseline 2, indicating that instead of fine-tuning a model trained on the source dataset with the target dataset, it is better to directly train the model on the target dataset. This result is clearly not in line with expectations and deserves in-depth study. Fourth, Method 2 is superior to the other three methods in most features except PCH and PCTM.

B. Pitch-related Analysis

In order to take a closer look to how these models manage the pitches, basic pitch histogram and pitch class histogram are drawn as Figs. 2 and 3, respectively.

Basic pitch histogram indicates the probability of presence of every pitch from C3 to B6. The pitch range covers four octaves, denoted as I, II, III, and IV, in Fig. 2. In Fig. 2(a), we can see that most of the pitches in the target dataset fall in octaves II and III, while most of the pitches in the source dataset fall in octave III. In other words, Jazz music has a wider range of pitches than generic music. In Fig. 2(b), the melodies generated by baseline methods seem to lose the diversity of pitch, in particular for Baseline 1, the pitches of the generated melodies tend to accumulate in octave III.

For Baseline 2, although there are quite a few pitches of the generated melodies in octave III, there are much more pitches in octave II. In Fig. 2(a), it is obvious that the basic pitch histogram of Method 2 is more similar to that of the target training data than Method 1. In summary, the basic pitch histogram of Method 2 is most similar to that of the target training data.

Pitch class histogram shows how the scales are used in melodies regardless of octaves. The lower part of Fig. 3(a) shows the scales of C-Major and C-Minor. As mentioned in Sec. III, the TT dataset contains both C-Major and C-Minor scales. As a result, TT has higher probabilities in Eb and Bb, as highlighted with the red box in Fig. 3(a). Fig. 3(b) compares the pitch class histograms of the target CY+R dataset and the melodies generate by Baseline 2 and Method 2. Although Method 2 achieves a lower OA in PCH as shown in Table V, we can see in Fig. 3(b) that Method 2 actually performs better than Baseline 2 in many scales highlighted with green box. A possible reason is that the OA of PCH in Table V is calculated in a sample-wise manner, but the PCH in Fig. 3(b) presents the overall pitch class distributions of the target dataset and the melodies generated by the models. This means that Method 2 learns the probability of the presence of a scale in Jazz music, but does not mean that the combination of pitches within each sample (i.e., a four-bar melody phrase) generated by it conforms to the overall target distribution.

344

(a)

(b)

Fig. 4. Score of melodies generated by (a) Method 1, (b) Method 2.

TABLE VI
THE RESULTS OF THE SUBJECTIVE TEST.

	CY+R	Baseline 1 (source)	Baseline 2 (target)	Method 1 (R=3)	Method 2 (R=1)
Type I	3.5905	2.6286	2.7524	2.8095	**2.8857**
Type II	3.7744	2.3179	2.3282	**2.6564**	2.3538
Type III	3.8500	2.2750	2.4000	**2.8500**	2.5250

C. Subjective Test

In addition to the objective test, we also conducted a subjective listening evaluation. We let the subjects listen to two demo melodies from the CY+R dataset, and asked them to rate five groups of four-bar melody phrases. Each group contained five melody phrases, one from the CY+R dataset, and the remaining four were generated by two baselines, Method 1 with $R = 3$, and Method 2 with $R = 1$, respectively. After listening to each test melody, the subjects were asked to give a score in a five-point Likert scale according to the degree of similarity between the test melody and the demo melody. The subjects were also required to provide information about their musical expertise. They could choose the category that best fits them from

Type I: seldom listening to soft jazz,
Type II: a music lover, and listening to jazz (soft jazz) sometimes, and
Type III: professional composer.

The results of the subjective test are shown in Table VI. 69 subjects participated in the test, of which 21 belonged to Type I, 39 belonged to Type II, and 9 belonged to Type III. From the results, the following observations can be drawn.

- Both Method 1 and Method 2 scored higher than the two baselines.
- Method 2 got the highest score for Type I subjects.
- The subjects for Types II and III preferred the melodies generated by Method 1 instead of Method 2.

The reason for the difference between objective and subjective results might be that Type II and Type III subjects are more aware of the existence of some jazz-related techniques. When any such pattern appears in a melody, subjects tend to consider it as more like real data. For example, Fig. 4 shows the score of melodies generated by Method 1 and Method 2 in one of the subjective test rounds. In the third bar of Fig.4(a), there seems to be a "chromatic enclosures", which is a part of jazz vocabulary. As a result, (a) gets a higher score of 3.44 than (b), which is 1.78. Maybe we should have adopted an algorithm to find such musical patterns and include them in the evaluation metrics.

VII. CONCLUSIONS

In this paper, we proposed using a recurrent VAE to randomly generate a jazz melody. We compared two methods of utilizing a big source dataset for transfer learning with a small target dataset and investigated the influence of the source-to-target data ratio. The overlapping areas computed based on the distributions of different pitch-related and rhythm-related features demonstrates that the multitask learning-based method (Method 2 in this paper) is slightly better than the fine-tuning-based method (Method 1) and the baseline methods that train the model on either the source dataset or the target dataset. On the other hand, the subjective test shows that the subjects who sometimes listen to soft jazz or are professional about composition think the melodies generated by Method 1 are better than those generated by Method 2. In the future, we will investigate more advanced methods for generating melodies. We also plan to develop better objective metrics for evaluation.

REFERENCES

[1] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.
[2] E. Waite, D. Eck, A. Roberts, and D. Abolafia, "Project Magenta: Generating long-term structure in songs and stories," [Online] https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/, 2016.
[3] N. Trieu and R. Keller, "JazzGAN: Improvising with generative adversarial networks," *Proc. Int. Workshop on Musical Metacreation*, 2018.
[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Proc. Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[5] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", arXiv preprint arXiv:1312.6114, 2013.

[6] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks," *Proc. AAAI Conf. Artificial Intelligence*, 2018.

[7] A. Roberts, J. Engel, C. Raffel, C.Hawthorne and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," *Proc. Int. Conf. Machine Learning*, 2018,

[8] J. D. Fernández and F. Vico, "AI methods in algorithmic composition: A comprehensive survey," *J. Artificial Intelligence Research*, vol. 48, no. 1, pp. 513–582, 2013.

[9] H. Chu, R. Urtasun, and S. Fidler, "Song from PI: A musically plausible network for pop music generation," *Proc. Int. Conf. Learning Representations, Workshop Track*, 2017.

[10] B. L. Sturm, J. Felipe Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning, " arXiv preprint arXiv:1604.08723, 2016.

[11] G. Hadjeres, F. Pachet and F. Nielsen, "DeepBach: A steerable model for Bach chorales generation, *Proc. Int. Conf. Machine Learning*, 2017.

[12] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[13] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 17171724, 2014.

[14] M.-Y. Huh, P. Agrawal and A. A. Efros, "What makes ImageNet good for transfer learning?," arXiv preprint arXiv:1608.08614, 2016.

[15] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[16] M. A. Hasegawa-Johnson *et al.*, "ASR for under-resourced languages from probabilistic transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 50-63, 2017.

[17] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, pp. 1–12, 2018.

[18] B.W.: Density estimation for statistics and data analysis, vol. 26. *CRC press*,1986.

[19] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," *Proc. Int. Society of Music Information Retrieval Conf.*, 2017.

[20] H.-Y. Lee, "Transfer learning," National Taiwan University, class lecture, [Online] http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/transfer.pdf, 2017.

[21] M. E. P. Davies, K. Yoshii, and M. Goto, "Transfer learning in MIR: sharing learned latent representations for music audio classification and similarity," *Proc. Int. Society of Music Information Retrieval Conf.*, 2013.

[22] J. Park, J. Lee, J. Park, J. W. Ha, J. Nam, "Representation learning of music using artist labels," *Proc. Int. Society of Music Information Retrieval Conf.*, 2018.

[23] W.-T. Lu and L. Su, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," *Proc. Int. Society of Music Information Retrieval Conf.*, 2018.

[24] Y.-J. Luo and L. Su, "Learning domain-adaptive latent representations of music signals using variational autoencoders," *Proc. Int. Society of Music Information Retrieval Conf.*, 2018.

[25] H.-W. Dong, W.-Y. Hsiao, and Yi-Hsuan Yang, "Pypianoroll: Open source Python package for handling multitrack pianoroll," *Proc. Int. Society of Music Information Retrieval Conf.*, Late-breaking and demo paper, 2018.

[26] H.-M. Liu, M.-H. Wu, and Y.-H. Yang, "Lead sheet generation and arrangement via a hybrid generative model," *Proc. Int. Society of Music Information Retrieval Conf.*, Late-breaking and demo paper, 2018.

[27] T. Fujishima, "Realtime chord recognition of musical sound: A system using common Lisp," *Proc. Int. Computer Music Conf.*, pp. 464–467, 1999.