Modeling the Compatibility of Stem Tracks to Generate Music Mashups

Jiawen Huang[*,2], Ju-Chiang Wang[1], Jordan B. L. Smith[1], Xuchen Song[1], and Yuxuan Wang[1]
[1] ByteDance
[2] Queen Mary University of London
jiawen.huang@qmul.ac.uk, {ju-chiang.wang, jordan.smith, xuchen.song, wangyuxuan.11}@bytedance.com

Abstract

A music mashup combines audio elements from two or more songs to create a new work. To reduce the time and effort required to make them, researchers have developed algorithms that predict the compatibility of audio elements. Prior work has focused on mixing unaltered excerpts, but advances in source separation enable the creation of mashups from isolated stems (e.g., vocals, drums, bass, etc.). In this work, we take advantage of separated stems not just for creating mashups, but for training a model that predicts the mutual compatibility of groups of excerpts, using self-supervised and semi-supervised methods. Specifically, we first produce a random mashup creation pipeline that combines stem tracks obtained via source separation, with key and tempo automatically adjusted to match, since these are prerequisites for high-quality mashups. To train a model to predict compatibility, we use stem tracks obtained from the same song as positive examples, and random combinations of stems with key and/or tempo unadjusted as negative examples. To improve the model and use more data, we also train on "average" examples: random combinations with matching key and tempo, where we treat them as unlabeled data as their true compatibility is unknown. To determine whether the combined signal or the set of stem signals is more indicative of the quality of the result, we experiment on two model architectures and train them using semi-supervised learning technique. Finally, we conduct objective and subjective evaluations of the system, comparing them to a standard rule-based system.

Introduction

In Margaret Boden's account of how creativity works, "combinational" creativity—the juxtaposition of unrelated ideas—and "exploratory" creativity—the searching within the rules of a style for exciting possibilities—are two essential modes of creative thinking (Boden 2007). Modeling these processes computationally is an important step for developing artificial creativity in the field of AI (Jordanous 2014). The combinatory possibilities and joy of exploration are perhaps two causes for the continued popularity of creating music mashups.

Mashups are a popular genre of music where new songs are created by combining audio excerpts (called *samples*) of other songs. Typically, the vocal part from one song is juxtaposed with the instrumental part of another, although it is also common for basic mashups to include samples of 3 songs (Boone 2013). However, creating mashups is challenging: it requires an expert's ear to decide whether two samples would make a good mashup, and it is time-consuming to search for pairs of songs that would work well together. Both issues are exacerbated when aiming to combine elements of three or more songs.

Accordingly, efforts to assist users in creating mashups or to automate the process have continued for over a decade. At minimum, two samples being combined should have the same tempo and time signature, and they should not clash harmonically; these criteria informed early systems for assisting mashup creation (Tokui 2008; Griffin, Kim, and Turnbull 2010), but they are also easy to meet using beat-tracking, key estimation, and audio-stretching algorithms. To predict the *mashability* (i.e., the compatibility) of a candidate group of samples is more challenging, but allows one to automate the creation of mashups.

Previous methods for estimating compatibility have relied on rule-based systems with hand-crafted features (Davies et al. 2014; Lee et al. 2015), even though the criteria used by listeners to judge mashup quality are unknown and undoubtedly complex. We thus propose to use a neural network to learn the criteria. The central challenge in training such a model is that there is no training data: i.e., no datasets of audio samples with annotations defining which combinations of samples are better than others. To address this, we propose to use self-supervised learning (by leveraging existing music as training data) and a semi-supervised approach that maximizes the utility of the other data we create.

We create training data by applying a supervised music source separation (MSS) algorithm (e.g., Jansson et al. 2017; Stöter, Liutkus, and Ito 2018) to extract independent stem tracks from existing music (i.e., separated vocal, bass, drum and other parts). We can then recombine the stems to generate many new mashups, with the original combinations serving as ground truth examples of 'good' mashups; in this way our model is **self-supervised**.[1] It is straightforward to

[*]The author performed this work as an intern at ByteDance.

[1]This may be different from conventional settings such as learning a representation of signals or their temporal coherence (Misra, Zitnick, and Hebert 2016; Huang, Chou, and Yang 2018).

use separated signals to help Music Information Retrieval (MIR) tasks such as music transcription (Pedersoli, Tzanetakis, and Yi 2020), singing voice detection (Stoller, Ewert, and Dixon 2018), and modeling vocal features (Lee 2019). However, to the best of our knowledge, no prior work has leveraged *supervised* MSS for automatic generation of new music pieces.

The above explains how to acquire positive examples for training the model. To obtain negative examples, we can use random combinations of stems with different keys and tempo that are almost guaranteed to sound awful. However, the extreme difference in compatibility between these two cases may lead to a highly polarized model that only regards stems as compatible if they were extracted from the same song. To avoid this, we use **semi-supervised learning**: we create random mashups that meet the minimum requirements for viability—combinations where the tempo and key are automatically matched—and treat them as "unlabeled" instances. This step aims to improve the reliability of the model, and it also means that our model sees more mashups, including many potentially "creative" ones, because the stems are sourced from different genres. This has been described as a key aspect of successful mashups: "the combination of musical congruity and contextual incongruity" (Brøvig-Hanssen and Harkins 2012).

Our contributions can be summarized as follows. First, we propose a novel framework that leverages the power of MSS and machine learning to generate music mashups. Second, we propose techniques to generate data without human labels and develop two deep neural network architectures that are trained in a self- and semi-supervised way. Third, we conduct objective and subjective evaluations, where we propose to use an unstudied dataset, Ayumix[2], to evaluate the task. The result demonstrates our AI mashups can achieve a good overall quality according to our participants.

Related Work

Early approaches to estimating mashability relied on fixed notions of what makes two sound clips mesh well. AutoMashUpper (AMU) modeled the mashability of two clips as a weighted sum of harmonic compatibility, rhythmic compatibility and spectral balance, each of which is computed as a correlation between two beat-synchronous representations (Davies et al. 2014). A system based on AMU was tailored to model good combinations of vocals and accompaniments, and included a constraint that the two parts should have contrasting amounts harmonic complexity, to avoid cases where both parts are complex or both are simple (Lee et al. 2015). Rule-based models of harmonic compatibility have been improved on (Bernardes, Davies, and Guedes 2017) and deployed in mashup creation tools (Maçãs et al. 2018), but all of these approaches share the potential weakness that a hand-crafted model of compatibility may fail to generalize, or to completely capture all of the subtle factors that contribute to balanced, high-quality mixes. Nonetheless, AMU is a well-described, well-motivated baseline model that has been re-implemented for open-source use.

In contrast to AMU, our system uses a supervised model where the training data were obtained by running MSS on existing songs. These steps were also taken to train Neural Loop Combiner (NLC), a neural network model that estimates audio compatibility of one-bar loops (Chen, Smith, and Yang 2020). However, NLC uses an unsupervised MSS algorithm designed to isolate looped content (Smith, Kawasaki, and Goto 2019), resulting in a very different system to ours, which uses supervised MSS to isolate vocal, bass, drum, and other parts.

First, since vocals are looped less often, NLC likely had far fewer instances of vocals in its extracted training set. This is a drawback since vocals are an essential part of mashups. Second, the data acquisition pipeline for NLC involves several heuristics to improve source separation quality, and the outputs are not guaranteed to contain distinct instruments (e.g., a positive training pair could include two drum loops). In contrast, the supervised separation we use leads to highly distinct stems, which is more appropriate for creating mashups. It also enables us to train a model where the input audio clips have a fixed role—namely, vocal, harmonic and drum parts—which is important, since the features that determine the compatibility likely depend strongly on the role. This design is also novel since it allows us to directly estimate the mashability of groups of stems instead of learning a representation space for embedding the stems, since mashability can be non-transitive.

A separate difference is that the authors of NLC chose to focus strictly on hip-hop, whereas our training dataset spans a wide variety of genres. We do this to obtain a more generalizable model (the semi-supervised technique explained later assists here too), and because much of the joy of mashups comes from the surprise of hearing two disparate samples from different genres work well together (Brøvig-Hanssen and Harkins 2012; Boone 2013).

Data Generation Pipeline

The pipeline aims to generate mashup candidates by mixing stems with different conditions. Then the candidates are sent to a machine learning model (described in the next section) to predict their compatibility. The pipeline includes three modules, *Music Source Separation* (MSS), *Mashup Database* (MashupDB), and *Mashup Generation*.

Music Source Separation

We built our in-house MSS system based on a U-net encoder/decoder Convolutional Neural Network (CNN) architecture with skip connections (Jansson et al. 2017; Prétet et al. 2019). The network consists of 12 layers of encoders and decoders. Following the standard evaluation procedures on the MUSDB18 testset, our model achieved competitive Signal to Distortion Ratio (SDR) compared to the state-of-the-art (Défossez et al. 2019). Specifically, the mean SDR's for the four output stems, *vocals*, *drums*, *bass*, and *other*, are 7.21, 5.68, 5.51, and 3.74 dB, respectively. For reproducibility, we suggest one can use Spleeter (Hennequin et al.

[2]About Ayu Creator Challenge 2020: https://randomjpop.blogspot.com/2020/05/ayumi-hamasaki-launches-the-creator-challenge.html accessed on March 10, 2021.

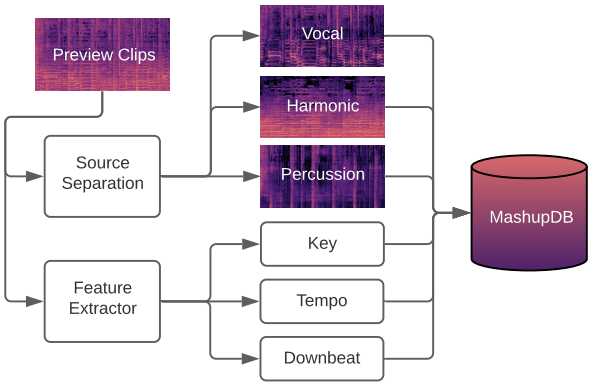


Figure 1: Diagram for MashupDB.

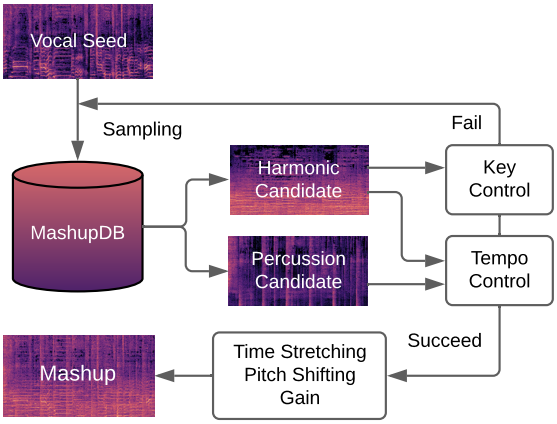


Figure 2: Mashup Generation pipeline.

2020), an open-source tool with pre-trained models, as a replacement. It is expected to have similar effectiveness for generating the stem samples.

Mashup Database

To construct a MashupDB, we need a music database which ideally spans many genres. To simplify, we use preview versions (clips around 30 seconds long) instead of the full audio tracks. For each clip, we extract the stems using MSS (see Figure 1), and we obtain the key, tempo, and downbeat information using madmom (Böck et al. 2016). In this work, we consider three major stem types: vocal (*vocals*), harmonic (*bass* + *other*), and percussion (*drums*)—i.e., we mix the original *bass* and *other* stems for their similar purpose of representing the harmonic component of the accompaniment.

Mashup Generation

Figure 2 depicts the pipeline for generating mashups of different types. First, a vocal stem is randomly selected as the seed. The system then searches in the MashupDB for harmonic and percussion stem candidates. Three conditions are allowed for the pipeline to generate mashups, namely *original*, *matched*, and *unmatched* conditions.

For *original* condition, it selects the harmonic and percussion stems from the same clip of the vocal seed, and mixes them without adjusting the key and tempo.

For *matched* condition, the generated mashup shall satisfy basic harmonic and rhythmic requirements. With the vocal seed as the reference, this can be achieved by finding the harmonic stem candidates within ± 3 semitones in key, and the percussion stem candidates having tempo within the ratio range of $[0.8, 1.2]$. If no such harmonic and percussion stems exist, the pipeline will draw another vocal seed and start the process again. If multiple harmonic or percussion stems are found, it selects one at random and then adjusts the key and tempo respectively to match the vocal seed using Rubberband (Cannam 2012). Like in (Davies et al. 2014), limiting the search space this way serves to avoid artefacts that can be caused by large amounts of pitch-shifting and time-stretching. The resulting mashups have duration between 25 and 60 seconds.

The purpose of the *unmatched* condition is to simulate cases where the key and tempo detection are wrong or the stems are incompatible. We propose three strategies to this end. First, *unmatched key*, by disabling the key control and pitch shifting, the selected harmonic stem only satisfies the tempo constraint. Second, *unmatched tempo*, by disabling the tempo control and time stretching, it makes the three stems very likely to have clashing tempos. To increase the chance that this is immediately perceptible, the first downbeat of one stem is offset randomly by $\pm$ 1 second at most. Third, *unmatched key & tempo*, by jointly applying the above two strategies, neither key nor tempo of the three stems meet the basic requirement.

For practical usage, we adopt the *matched* condition to generate the mashup candidates for testing. To generate a good mashup, however, there are still numerous factors that the pipeline does not account for, such as chord, instrumentation, and groove. We believe those factors are too complicated to enumerate and justify manually. In addition, any errors in key, beat and downbeat detection and any artefacts from MSS and Rubberband can be propagated. All these factors can result in only part of the mashup candidates being good. To solve this problem, we make use of the pipeline to generate data to train machine learning models that could identify good pieces.

Modeling the Mashability

Data Preparation

To train the model to predict mashability, we build a training MashupDB based on a large music collection. Then, we employ the Mashup Generation pipeline to generate the positive, negative, and unlabeled data with the *original*, *unmatched*, and *matched* conditions, respectively.

Our system is fully audio-based, so for testing or practical use, one can build a new MashupDB different from the training one, depending on what musical sources the user wants

to explore to generate mashups.

The role of positive data is to guide the model to learn what a good combination of stems should sound like. Since the test samples are generated with the *matched* condition, we also include the unlabeled data to ensure the model can see similar samples of the same condition, which we treat as the intermediate space between positive and negative data.

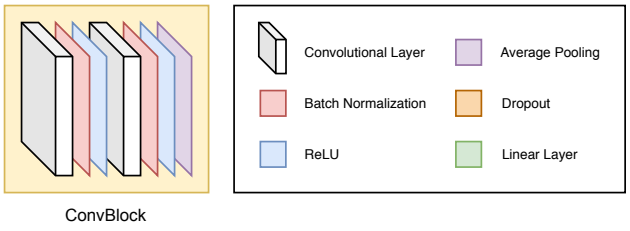
On the other hand, the role of negative data is to provide examples that do not meet the basic harmonic and rhythmic compatibility to be a musical piece. It has been shown that incorporating domain knowledge in designing the negative sampling strategies could help learning a robust model (Schroff, Kalenichenko, and Philbin 2015; Wu et al. 2017; Riad et al. 2018). For Neural Loop Combiner, Chen et al. investigated 5 ways of negative sampling, including random combinations of different loops, shifting or rearranging the order of beats (Chen, Smith, and Yang 2020). Those techniques were devised in order to make 1-bar loops that already had the same tempo and duration incompatible. However, none of them were guaranteed to lead to incompatible pairs. The negative sampling strategies we have used are more clearly extreme—juxtaposing incompatible keys, tempos, and beat phases, in clips of 16 bars or longer—so most of their proposed strategies are not needed for our scenario. Instead, manipulating the key and tempo for negative sampling is more likely reflecting the typical mistakes that an existing mashup system could make.

Semi-Supervised Learning

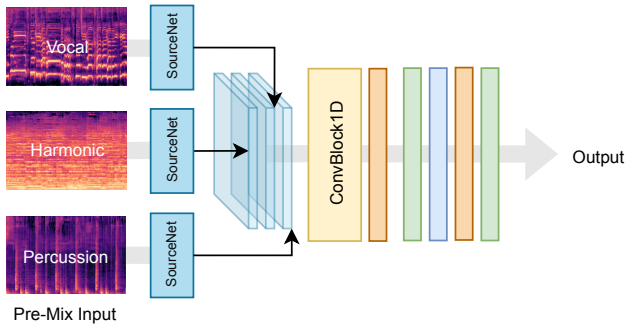
Based on the nature of the positive and negative data, perceptually distinguishing between them is very easy for humans. We also observe similar trends that the training can achieve almost 100% accuracy using only positive and negative labels on the validation set. To mitigate this concern, we explore semi-supervised learning methods that enable the model training to take the unlabeled data into consideration (Kingma et al. 2014; Kipf and Welling 2017). Generally speaking, the quality of these unlabeled data is expected to vary widely, and thus we treat them as "average" examples with the hope that the training could help clarify them.

We adopt the method proposed by (Zheng, Zheng, and Yang 2017), called *label smoothing regularization for outliers* (LSRO). The original use case was on person re-identification in images, where the unlabeled images are generated by a generative adversarial network (GAN). The main idea of LSRO is to assign a uniform label distribution to the unlabeled data. In our case, supposing that the model output is two-dimensional classes (positive, negative), we assign a virtual label of $(0.5, 0.5)$ to each of the unlabeled data for loss computation. In this way, the learning process is regularized when an unlabeled example is close to either positive examples or negative examples in the feature space.

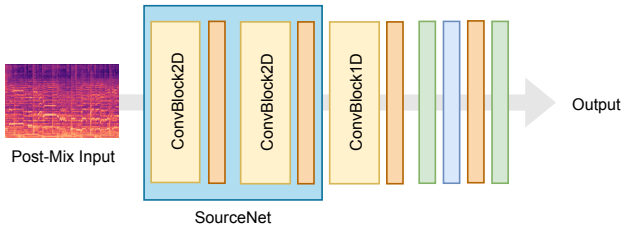
As pointed out by (Zheng, Zheng, and Yang 2017), LSRO exhibited superior performance to its semi-supervised learning counterparts, such as Pseudo Labeling (Lee 2013) and All-in-One (Salimans et al. 2016), in their application. In fact, we experimented with Pseudo Labeling, and it turned out not to be as good as LSRO in our pilot study. Therefore, we opt to present only LSRO in this paper.



(a) ConvBlock



(b) PreMixNet



(c) PostMixNet

Figure 3: The proposed model architectures.

Network Architectures

We formulate the task as a classification problem. The model can be described by $p(y \mid V, H, P)$, where V, H, and P are respectively the input signals of the vocal, harmonic, and percussion stems, and $y \in \{0, 1\}$ is the binary label (good or bad). We take the output posterior probability as the mashability of the combination.

We propose two types of model architecture, *PreMixNet* and *PostMixNet*, where the former takes the three individual stems as input (denoted as *Pre-Mix*), and the latter takes the single downmixed track as input (denoted as *Post-Mix*). The two models are illustrated in Figure 3(b) and Figure 3(c). When mixing audio tracks, it is common practice to judge the quality by listening alternately to the downmixed track and to each individual stem, or by comparing side-by-side the compatibility of a pair of individual stems (Senior 2011). Because this process is complicated and different from person to person, we want to investigate whether the difference between the Pre-Mix and Post-Mix representations would affect the final performance.

Figure 3(a) defines the model components and the basic convolutional block (denoted as ConvBlock), which consists of two convolutional layers, each followed by a Batch-

Norm and a ReLU, and finally an average pooling layer. The convolutional layers in ConvBlock can be either 1D or 2D. Both PreMixNet and PostMixNet take the 128-bin mel-spectrogram as input and have several convolutional blocks (Figures 3(b) and 3(c)). A unit we call *SourceNet*, which contains two ConvBlock2D's, takes the input of either *Pre-Mix* or *Post-Mix*. The purpose of SourceNet is to enhance the saliency of patterns on the mel-spectrograms (McFee and Bello 2017). We use a kernel size of (3,3) and a temporal pooling size of (2,1) for each ConvBlock2D. Each SourceNet contains 4 convolutional layers (see 3(c)) with 64, 64, 128, and 128 filters, respectively. A ConvBlock1D follows to summarize the content for each frame. For a ConvBlock1D, we use kernel sizes of (3, 128) and (3, 1) for the two convolutional layers, respectively (see 3(a)), with 256 filters and a temporal pooling size of (2, 1) for both. The time dimension is then reduced by average pooling, and the filter dimension is reduced by adding the maximum and average. Finally, two fully-connected layers (with 128 features) make the output. In PreMixNet, we combine the three stem SourceNet outputs by concatenating along the filter dimension. The three SourceNets do not share the weights.

Experiments

Owing to people's different music backgrounds, tastes, and understandings of harmonic and rhythmic compatibility, judging the quality of a mashup is highly subjective. Following prior works (Davies et al. 2014; Lee et al. 2015; Xing et al. 2020), we focus more on subjective evaluation but also provide the objective result for a reality check that the models are learning something useful.

Datasets

We built a training MashupDB based on an internal music collection of 33,192 music clips (with average duration about 30 seconds). It covers many genres of popular music, including Asian pop, Western pop, rock, folk, electronic, and hip-hop. Most of them have vocals, but a few are purely instrumental. We used the pipeline to generate a balanced dataset, denoted as *in-house*, yielding a set of 51,507 examples, where 1/3 are positive, 1/3 negative, and 1/3 unlabeled. Note that any music collection could work as a training MashupDB as long as it is sufficiently large and generic.

For testing, we built two MashupDB's based on two publicly available datasets: Harmonix Set (Nieto et al. 2019) and Ayumix2020. We note that even though these two datasets may have few common songs to the training set, it is still valid because we did not use any human labels in training.

Harmonix Set contains 912 full-tracks covering genres in a wide range of popular western genres, such as pop, electronic, hip-hop, rock, country, and metal. Human annotations of beats, downbeats, and segments are available. To build the MashupDB, we extracted all the verse and chorus parts based on the segment labels, and used the beat and downbeat labels to obtain the tempo. There are in total 5,310 vocal seeds obtained from the verse and chorus segments of the 912 songs. In order to have a balanced test set, we applied the pipeline to generate, for each vocal seed, 5 clips with *matched* condition, leading to 26,550 unlabeled mashup candidates.

The Ayumix2020 dataset originates from the Ayu Creator Challenge event, in which the record label Avex released 100 studio acapella tracks[3] from J-pop star Ayumi Hamasaki. The aim was to encourage creators to create and share remixes using these tracks during the COVID-19 pandemic. We selected 30 Ayumix songs, segmented a chorus clip from each song, and extracted its corresponding key and tempo information from the original (non-acapella) version using madmom. We used Harmonix MashupDB to generate 180 unlabeled candidates for each song, meaning that the accompaniment stems (harmonic + percussion) to be combined with an Ayumix vocal stem are all from Harmonix Set.

	Accuracy	Average Rank
PreMixNet	99.8%	1.0000
PostMixNet	98.5%	1.0019
PreMixNet + LSRO	98.9%	1.0024
PostMixNet + LSRO	99.3%	1.0338

Table 1: Results for objective evaluation.

Model Training

To compare the effectiveness of different model settings in the evaluation, we trained PreMixNet and PostMixNet with and without unlabeled data. We used a ratio of $4:1$ for splitting the in-house dataset into training and validation sets. All models were trained with Adam Optimization with a $1\mathrm{e}^{-4}$ learning rate. All models were trained using an NVIDIA Tesla-V100 GPU for 3 days.

Objective Evaluation

For objective evaluation, we are interested in knowing (i) how well a trained model classifies the generated labeled data from the Harmonix Set (cross-dataset evaluation), and (ii) how the positive data rank against the unlabeled data on the test set (accompaniment retrieval). To this end, we used the pipeline to further generate 5,310 negative examples with *unmatched* condition and 5,310 positive examples with *original* condition based on the vocal seeds.

The first objective can be evaluated in terms of accuracy on a binary prediction task using 0.5 as the threshold. For the second objective, we used the test set of 26,550 unlabeled examples as the retrieval database. Following prior work (Chen, Smith, and Yang 2020), we calculate the ranking position (≥ 1) based on the probability scores for each positive example, and report the average rank.

As shown in Table 1, all proposed models achieve very high accuracy on labeled data and successfully rank the positive (original) one at the top. This demonstrates the model effectively discriminates between positive and negative examples, and between positive and unlabeled examples. However, this result does not show that the model is able to se-

[3]Studio acapella is the clean vocal stem track of a commercially released song. The data are available at https://youtube.com/playlist?list=PL57sdSoJE6THHJyhWfFU7z1RdmWkdmL43 accessed on Mar 10, 2021.

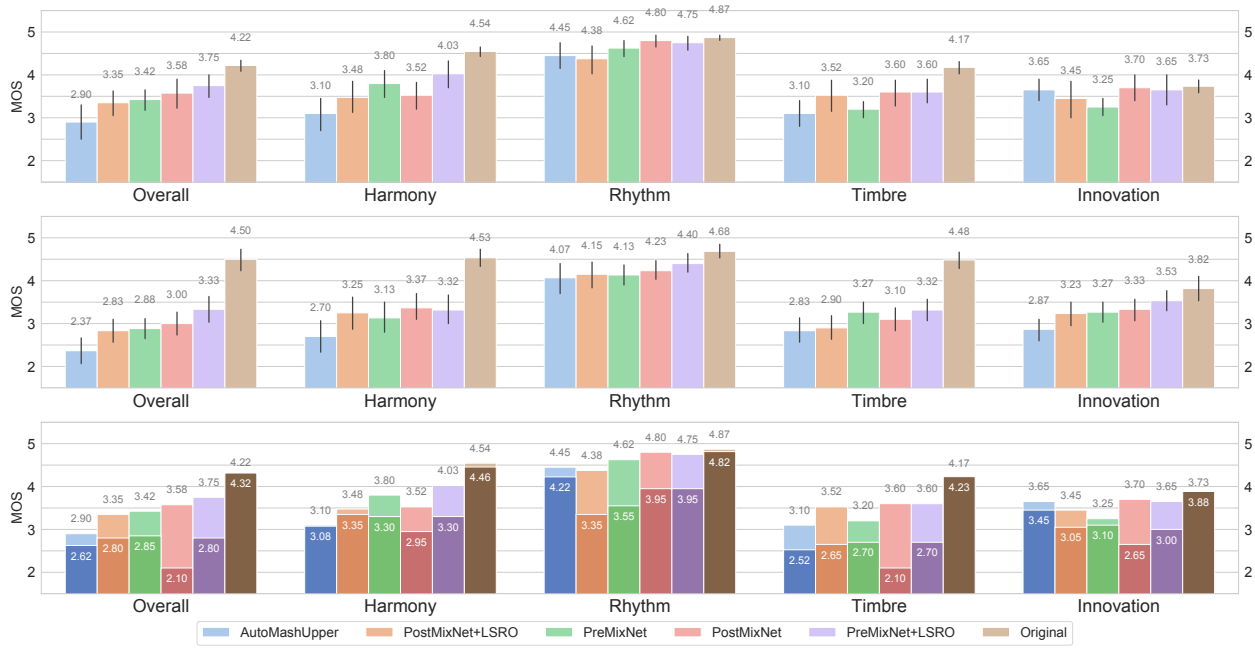


Figure 4: Comparison of different systems in subjective evaluation. All values are Mean Opinion Score (MOS). The upper and center sub-figures display the generation and retrieval tasks, respectively. The bottom sub-figure shows the MOS of top-20 mashups (in lighter colors) and bottom-20 mashups (in darker colors) for the retrieval task.

lect unlabeled mashups that human audiences would prefer. Therefore, we conduct the following subjective evaluation.

Subjective Evaluation

Baseline We compare our models with AutoMashUpper (AMU), which represents a rule-based automatic mashup creation system. We implemented the system based on an open-source tool.[4] AMU takes two stems as input. To get the mashability of a group of three stems, we take the average of the three pairwise mashabilities.

Configuration Five systems were compared in the subjective evaluation (AMU and our four model variants), and we considered two evaluation tasks: *generation* and *retrieval*.

For the generation task, no vocal seed constraint is applied. Given a test MashupDB, we used the pipeline to generate as many *matched* candidates as possible. Then, a model is employed to recommend the top candidates so that musicians may benefit from the ideas by listening to them. We used Harmonix Set alone for this task and picked the top 20 and bottom 20 mashups for each system in the listening test. This yielded $(20 + 20) \times 5 = 200$ samples to be evaluated.

For the retrieval task, we used the 30 selected songs from Ayumix as the test vocal seeds. We generated 180 mashup candidates for each vocal seed and selected the top 1 by each corresponding system, leading to 30 (queries) $\times 5$ (systems) $= 150$ samples to be evaluated.

Questionnaire Subjects were required to rate the quality of each test sample on a scale of 1 to 5 (1: awful, 2: poor, 3: fair, 4: good, and 5: excellent) according to five aspects: rhythm, harmony, timbre, innovation, and an overall score. The questions were as follows: *Harmony*: How good is the harmonic compatibility between the vocals and music? *Rhythm*: How good is the rhythmic compatibility between the vocals and music? *Timbre*: Is each instrumental part clear and recognizable? How spectrally balanced and professional is the combination? *Innovation*: Does the combination sound unexpected without clashing? *Overall*: Does it sound like it was professionally made, not a randomly remixed piece?

Methodology We developed a web-based platform for the listening test. Each page presented two audio samples which use the same vocal stem but different accompaniments. Subjects were not told about the relationship between the two samples, so they do not necessarily compare between them. Subjects were also asked to answer whether or not they have heard the vocal melody of the singer before. Pages were shown in a random order, and each page was rated by three subjects. To assign the samples to web pages, separate strategies were used for the generation and retrieval tasks. For generation, each page presented a mashup (selected by a system) and its original version (generated by the pipeline with the *original* condition), in a random order. This means that

[4] https://github.com/migperfer/AutoMashupper accessed on Mar 10, 2021.

	Generation	Retrieval
Number of subjects	28	12
Time spent per page	95 sec	108 sec
Report vocal 'never heard'	68%	75%
Cronbach's Alpha	0.79	0.73

Table 2: Statistics of subjects.

subjects were also required to rate the original version for a fair evaluation. For retrieval, because rating 6 samples (5 systems + 1 original) for a query vocal in one page is too much, we divided them into three pairs using a random permutation, so that each page still displayed two samples.

Statistics of Subjects We recruited 40 subjects. Each subject rated at least 10 pages. According to self-report, 85% said they listened to music on a daily basis, and 40% had experience in music production. Table 2 summarizes the statistics of the two tasks.

Subjective Result Figure 4 shows the Mean Opinion Score (MOS) results of all cases. For each test clip, we take the median score from the three subjects, and average them over all clips for a system. In Tables 3 and 4, we also report the significance levels using Mann-Whitney U test for generation, and Wilcoxon rank-sum test for retrieval. In these tables, 'O', 'H', 'R', 'T', and 'I' stand for overall, harmony, rhythm, timbre, and innovation, respectively. Several points can be made as follows.

Our proposed models outperform AMU by a large margin on 'overall' and on harmony, and by a comfortable margin on timbre. PreMixNet+LSRO performs the best among different variants in most of the cases, indicating that modeling individual stems and using unlabeled data with LSRO can improve predictions of mashability. The significance test results of PreMixNet+LSRO versus other systems are shown in Table 4. Other than PreMixNet+LSRO, we see that PostMixNet also works well. Closer investigation reveals that PostMixNet prefers mashups with strong percussion stems, which are possibly easier to learn from the *Post-Mix* positive examples alone. However, because it learns no unlabeled data, we can still find a few bad cases in the top 20 that degrade the MOS overall.

From Table 3 we observe clearly that, for our models, the top-20 mashups outperform the bottom-20 ones significantly in most cases. By contrast, for AMU, this only happens on timbre. This demonstrates the effectiveness of a learned model in distinguishing between good and bad mashups.

All the systems obtain higher MOS on rhythm. We attribute this to the straightforward beat structure of songs in Harmonix Set (Nieto et al. 2019). We also note that, by observing the low rhythm MOS of bottom-20, there are still many mashups having very poor rhythmic compatibility in the test set. This is because tempo can vary within a clip, but the time-stretching by Rubberband is performed globally, leading to beat phase mismatches in the later measures.

The MOS for the original version can be regarded as the upper bound. But we note that these scores could be overrated, because in many cases when a subject is familiar with the song, it is very likely she/he would directly rate 5 for the original version without further judgement. On the other hand, the original versions have noticeably fewer MSS artefacts, and this may also have influenced the ratings.

Finally, we find that the retrieval task is generally more difficult than the generation one. This is as expected because Ayumix vocals represent the most iconic J-pop style, while Harmonix Set is composed mostly by Western pop songs. We also ascribe the higher MOS of Ayumix's original versions to the relatively low quality of the generated Ayumix mashups, since it may be easier to find which is the original version by the subjects.

System	O	H	R	T	I
Original	×	×	×	×	×
AutoMashUpper	×	×	×	.01	×
PreMixNet	.05	×	.001	.01	×
PostMixNet	.001	.01	.01	.001	.001
PreMixNet + LSRO	.001	.05	.01	.01	.05
PostMixNet + LSRO	.05	×	.01	.01	×

Table 3: Significance levels (p-value) between top-20 and bottom-20 mashups, where '×' means no significance.

System (Generation)	O	H	R	T	I
AutoMashUpper	.01	.001	×	.05	×
PreMixNet	.05	×	×	.05	.05
PostMixNet	×	.05	×	×	×
PostMixNet + LSRO	.05	.05	.05	×	×
System (Retrieval)	**O**	**H**	**R**	**T**	**I**
AutoMashUpper	.001	.05	×	.05	.01
PreMixNet	.05	×	×	×	×
PostMixNet	.05	×	×	×	×
PostMixNet + LSRO	.01	×	×	.01	×

Table 4: Significance levels (p-value) on PreMixNet+LSRO versus other systems, where '×' means no significance.

Conclusion and Future Work

In this paper, we have proposed a novel framework to generate music mashups from separated stems and shown the potential of a learning-based approach to modeling the mashability. No human labels are needed, and the approach could work on any large music collection. The results of a listening test demonstrated the effectiveness of our models. Among the models we tested, adding unlabeled data to train a PreMixNet gave the best performance.

Currently our system's efficiency is low because the model cannot predict the mashability before the key and tempo of each stem were adjusted. For improvement, we plan to learn an effective embedding that is invariant to key and tempo to hopefully generalize the mashability of stems using the metric learning techniques (Hu, Lu, and Tan 2014; Lin et al. 2017; Humphrey, Bello, and LeCun 2013; Movshovitz-Attias et al. 2017). In addition, we plan to use the generated mashups as means of data augmentation to improve MIR tasks such as beat tracking and chord estimation.

Acknowledgements

We would like to thank Yi Deng and Yuan Wan for helping prepare the dataset, the participants for providing their ratings in the subjective evaluation, and the anonymous reviewers for their valuable comments.

References

Bernardes, G.; Davies, M. E.; and Guedes, C. 2017. A hierarchical harmonic mixing method. In *International Symposium on Computer Music Multidisciplinary Research*, 151–170. Springer.

Böck, S.; Korzeniowski, F.; Schlüter, J.; Krebs, F.; and Widmer, G. 2016. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, 1174–1178.

Boden, M. 2007. Creativity in a nutshell. *Think* 5: 83–96.

Boone, C. 2013. Mashing: Toward a typology of recycled music. *Music Theory Online* 19(3).

Brøvig-Hanssen, R.; and Harkins, P. 2012. Contextual incongruity and musical congruity: the aesthetics and humour of mash-ups. *Popular Music* 87–104.

Cannam, C. 2012. Rubber Band Audio Time Stretcher Library. URL https://breakfastquay.com/rubberband/. Accessed: March 10, 2021.

Chen, B.-Y.; Smith, J. B. L.; and Yang, Y.-H. 2020. Neural Loop Combiner: Neural Network Models for Assessing the Compatibility of Loops. In *Proceedings of the International Society for Music Information Retrieval Conference*, 424–431.

Davies, M. E.; Hamel, P.; Yoshii, K.; and Goto, M. 2014. AutoMashUpper: Automatic creation of multi-song music mashups. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22(12): 1726–1737.

Défossez, A.; Usunier, N.; Bottou, L.; and Bach, F. 2019. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254* .

Griffin, G.; Kim, Y. E.; and Turnbull, D. 2010. Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 437–440.

Hennequin, R.; Khlif, A.; Voituret, F.; and Moussallam, M. 2020. Spleeter: A fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software* 5(50): 2154.

Hu, J.; Lu, J.; and Tan, Y.-P. 2014. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1875–1882.

Huang, Y.-S.; Chou, S.-Y.; and Yang, Y.-H. 2018. Generating music medleys via playing music puzzle games. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2281–2288.

Humphrey, E. J.; Bello, J. P.; and LeCun, Y. 2013. Feature learning and deep architectures: New directions for music informatics. *Journal of Intelligent Information Systems* 41(3): 461–481.

Jansson, A.; Humphrey, E.; Montecchio, N.; Bittner, R.; Kumar, A.; and Weyde, T. 2017. Singing voice separation with deep u-net convolutional networks. In *Proceedings of the International Society for Music Information Retrieval Conference*, 323–332.

Jordanous, A. 2014. What is Computational Creativity? URL https://www.creativitypost.com/article/what_is_computational_creativity. Accessed: March 10, 2021.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 3581–3589.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

Lee, C.-L.; Lin, Y.-T.; Yao, Z.-R.; Lee, F.-Y.; and Wu, J.-L. 2015. Automatic Mashup Creation by Considering both Vertical and Horizontal Mashabilities. In *Proceedings of the International Society for Music Information Retrieval Conference*, 399–405.

Lee, D.-H. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Proceedings of Workshop on Challenges in Representation Learning, in conjunction with International Conference on Machine Learning* .

Lee, S. 2019. Investigation of Timbre-related Music Feature Learning using Separated Vocal Signals. *Journal of Broadcast Engineering* 24(6): 1024–1034.

Lin, J.-C.; Wei, W.-L.; Yang, J.; Wang, H.-M.; and Liao, H.-Y. M. 2017. Automatic music video generation based on simultaneous soundtrack recommendation and video editing. In *Proceedings of the 25th ACM international conference on Multimedia*, 519–527.

Maçãs, C.; Rodrigues, A.; Bernardes, G.; and Machado, P. 2018. MixMash: A visualisation system for musical mashup creation. In *The 22nd International Conference on Information Visualisation*, 471–477.

McFee, B.; and Bello, J. P. 2017. Structured Training for Large-Vocabulary Chord Recognition. In *Proceedings of the International Society for Music Information Retrieval Conference*, 188–194.

Misra, I.; Zitnick, C. L.; and Hebert, M. 2016. Shuffle and learn: Unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, 527–544. Springer.

Movshovitz-Attias, Y.; Toshev, A.; Leung, T. K.; Ioffe, S.; and Singh, S. 2017. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, 360–368.

Nieto, O.; McCallum, M.; Davies, M. E.; Robertson, A.; Stark, A. M.; and Egozy, E. 2019. The Harmonix Set: Beats, Downbeats, and Functional Segment Annotations of Western Popular Music. In *Proceedings of the International Society for Music Information Retrieval Conference*, 565–572.

Pedersoli, F.; Tzanetakis, G.; and Yi, K. M. 2020. Improving Music Transcription by Pre-stacking a U-Net. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 506–510.

Prétet, L.; Hennequin, R.; Royo-Letelier, J.; and Vaglio, A. 2019. Singing voice separation: A study on training data. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 506–510.

Riad, R.; Dancette, C.; Karadayi, J.; Zeghidour, N.; Schatz, T.; and Dupoux, E. 2018. Sampling strategies in Siamese Networks for unsupervised speech representation learning. In *Proceedings of INTERSPEECH, the 19th Annual Conference of the International Speech Communication Association*.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2234–2242.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815–823.

Senior, M. 2011. *Mixing secrets for the small studio*. Taylor & Francis.

Smith, J. B. L.; Kawasaki, Y.; and Goto, M. 2019. Unmixer: An Interface for Extracting and Remixing Loops. In *Proceedings of the International Society for Music Information Retrieval Conference*, 824–831.

Stoller, D.; Ewert, S.; and Dixon, S. 2018. Jointly detecting and separating singing voice: A multi-task approach. In *International Conference on Latent Variable Analysis and Signal Separation*, 329–339. Springer.

Stöter, F.-R.; Liutkus, A.; and Ito, N. 2018. The 2018 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, 293–305. Springer.

Tokui, N. 2008. Massh! A web-based collective music mashup system. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, 526–527.

Wu, C.-Y.; Manmatha, R.; Smola, A. J.; and Krahenbuhl, P. 2017. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2840–2848.

Xing, B.; Zhang, X.; Zhang, K.; Wu, X.; Zhang, H.; Zheng, J.; Zhang, L.; and Sun, S. 2020. PopMash: An automatic musical-mashup system using computation of musical and lyrical agreement for transitions. *Multimedia Tools and Applications* 79: 21841–21871.

Zheng, Z.; Zheng, L.; and Yang, Y. 2017. Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. In *Proceedings of the IEEE International Conference on Computer Vision*, 3754–3762.