NEURAL LOOP COMBINER: NEURAL NETWORK MODELS FOR ASSESSING THE COMPATIBILITY OF LOOPS

Bo-Yu Chen[1], **Jordan B. L. Smith**[2], **and Yi-Hsuan Yang**[1]
[1] Academia Sinica, Taiwan, [2] TikTok, London, UK
bernie40916@citi.sinica.edu.tw, jordan.smith@tiktok.com, yang@citi.sinica.edu.tw

ABSTRACT

Music producers who use loops may have access to thousands in loop libraries, but finding ones that are compatible is a time-consuming process; we hope to reduce this burden with automation. State-of-the-art systems for estimating compatibility, such as AutoMashUpper, are mostly rule-based and could be improved on with machine learning. To train a model, we need a large set of loops with ground truth compatibility values. No such dataset exists, so we extract loops from existing music to obtain positive examples of compatible loops, and propose and compare various strategies for choosing negative examples. For reproducibility, we curate data from the Free Music Archive. Using this data, we investigate two types of model architectures for estimating the compatibility of loops: one based on a Siamese network, and the other a pure convolutional neural network (CNN). We conducted a user study in which participants rated the quality of the combinations suggested by each model, and found the CNN to outperform the Siamese network. Both model-based approaches outperformed the rule-based one. We have opened source the code for building the models and the dataset.

1. INTRODUCTION

The emergence of digital audio editing techniques and software such as digital audio workstations (DAWs) has changed the way people make music. In a DAW, producers can easily produce music by making use of pre-existing audio as loops. Loops are used in many styles, especially Electronic and Hip-Hop music. Perhaps noticing the business value of such loop-based music, many companies such as Splice and LANDR have built up databases of loops. Community efforts have flourished too, including Looperman, an audio community that provides free loops for music producers to use. But having so many resources to choose from leaves a separate problem: how to navigate the library efficiently and how to choose which loops to combine. These tasks require human practise, expertise, and patience: to recognize the characteristics of the loop and to determine whether it is suitable for their song. However, thanks to recent advances in music information retrieval (MIR), we believe it is possible to make loop selection from large-scale loops database easier.

A few existing systems could potentially solve this problem. Kitahara *et al.* [1] presented an intelligent loop sequencer that chooses loops to fit a user-defined 'excitement' curve, but, excitement only accounts for part of what makes two loops compatible. The AutoMashUpper system [2] could also be applied to the loop selection process: it involves estimating the 'mashability' of two songs (i.e., how compatible they would be if played at the same time). It is a rule-based system that computes the harmonic and rhythmic similarity and spectral balance, and it has proven useful in other applications [3,4]. However, AutoMashUpper has two limitations that could be improved by current machine learning methods. First, it regards the harmonic and rhythmic similarity as part of mashability, while it is actually possible that two music segments match perfectly despite having different harmonies and rhythms. Second, hand-crafted representations cannot fully describe all features in the music segment.

To capture the more complicated compatible relationship between two music segments, we propose to employ modern machine learning models to learn to predict the compatibility of loops. A major obstacle preventing the development of such a model is the lack of a dataset with sufficient labelled data. While there are many datasets of loops, none provide ground truth compatibility values.

We make two main contributions. First, we propose a data generation pipeline that supplies labelled data for model-based compatibility estimation (see Section 3). This is done by using an existing loop extraction algorithm [5] to yield positive examples of loops that have been used together in real loop-based music. We also propose procedures to ensure the quality of the positive data, and investigate different strategies to mine negative data from the result of loop separation.

Second, we develop and evaluate two neural network based models for loop compatibility estimation (see Section 4), one based on convolutional neural network (CNN) and the other Siamese neural network (SNN) [6]. The two approaches perform loop compatibility estimation using different approaches: the CNN directly evaluates the combination of loops in a time-frequency representation, whereas SNN processes the two loops to be evaluated for compatibility separately. We report both objective and sub-

jective evaluations (see Section 5) to study the performance of these approaches, and to compare the model-based approaches with AutoMashUpper.

The audio data we employ to build the neural network is from the Free Music Archive (FMA) [7] (see Section 3.1), which make data re-distribution easier. Moreover, we have open-sourced the code implementing the proposed CNN and SNN models at `https://github.com/mir-aidj/neural-loop-combiner`.

2. RELATED WORK

Along with the growing interest in loop-based music, academic studies focusing on assisting loop-based music production have become popular. An early study [8] proposed two ways to help create loop-based music: automatic loop extraction and assisted loop selection. This laid the foundation for this field.

For **loop extraction**, Shi and Mysore [9] proposed an interface with automatic and semi-automatic modes for the producer to find the most suitable segment in a piece of music to excerpt and use as a loop, by cropping directly. These algorithms estimate similarity with handcrafted features: harmony, timbre, and energy [8, 9]. However, the segments are excerpted without any attempt to isolate one part of a potentially multi-part piece of music. One solution to this used a common quality of loop-based music—that loops are often introduced one at a time—to recover the source tracks [10], but not all pieces have this form. To tackle both problems, Smith *et al.* [5, 11] proposed to extract loops by taking into account how they repeat. The algorithm they proposed can directly extract the loops from songs using nonnegative tensor factorization (NTF).

For **loop selection**, Kitahara *et al.* [1] proposed to select the loops according to the level of excitement entered by the user. However, we see three limitations in this work. First, the study focuses on Techno music only—while excitement is certainly highly relevant to this genre, the approach may not generalize well to other genres. Second, the level of excitement has to be manually entered by a user, which limits its usability. Third, the study does not take compatibility into consideration. As a result, even though a user can find loops with the desired excitement level, the loops may not be compatible with one another.

To the best of our knowledge, the work of Davies *et al.* [2, 12] represents the first study to investigate mashability estimation. Their AutoMashUpper system represents each music segment (not necessarily a loop) with a chromagram, a rhythmic representation, and a spectral band representation, each made to be beat-synchronous. Given two songs, it computes the similarity between the chromagrams and the rhythmic representations, and the spectral balance between the songs, to obtain the final mashability estimate. While AutoMashUpper is developed for general mashability estimation of longer music pieces, it can also be applied to loop selection.

Lee *et al.* [13] extended AutoMashUpper, proposing that two parts should be more compatible if they have complimentary amounts of harmonic instability. They generated mashups that were preferred by listeners to the output of AutoMashUpper, but they focused on a particular type of mash-up (combinations of vocal and backing tracks) whereas we focus on general loop compatibility. Bernardes *et al.* [14, 15] proposed several harmonic mixing approaches based on psychoacoustic principles and "tonal interval space indicators." Xing *et al.* [16] used paired audio, MIDI, and lyrics data for mashability estimation, taking into account similarity in melody, rhythm, and rhyming lyrics. Among these works, AutoMashUpper stands out as a general-purpose system requiring only audio data, so we consider it as our baseline.

Data type	# loops	# loop pairs	# songs
Training set	9,048	12,774	2,702
Validation set	2,355	3,195	7,06
Test set	200	100	100
$\sum$	11,603	16,069	3,508

Table 1: Statistics of the dataset, which is derived from Hip-Hop songs in the Free Music Archive (FMA) [7] using the data generation pipeline shown in Figure 1.

3. DATA GENERATION PIPELINE

To create the labeled data needed for training our models, we obtained a dataset of loop-based music and used loop extraction [5] to obtain audio loops. We use a new loop refinement procedure to reduce the number of duplicate loops per song, and a loop pairing procedure to get pairs of loops that co-occur in songs.

3.1 Dataset

To extract a collection of loops, we first need a large set of songs that use loops. We chose to use music from the Free Music Archive (FMA) [7] so that we could redistribute the audio data. We restricted ourselves to the genre of Hip-Hop for three reasons: First, we could not manually verify whether each song used loops, so we needed to use a genre known for using loops. Second, for the loop extraction step to work, we needed the music to have a steady tempo. Third, we expected that a Hip-Hop subset would provide a useful variety of loops since the genre is known for incorporating loops from many other genres

We collected 6,868 Hip-Hop songs in total from FMA by searching for the tag "Hip-Hop." We passed these songs through the proposed data generation pipeline, and kept the 3,508 songs from which we could find at least one valid loop pair. Specifically, from these 3,508 songs, we obtained 11,603 valid loops and 16,069 valid loop pairs. From the full set of loops, we reserved 200 loops (1 pair each from 100 different songs) for the evaluations (see Section 5), and then split the rest into train and validation sets in a 4-to-1 ratio (see Table 1). [1]

[1] To facilitate follow-up research, we plan to share publicly the loops we extracted from FMA songs. While FMA has looser copyright re-

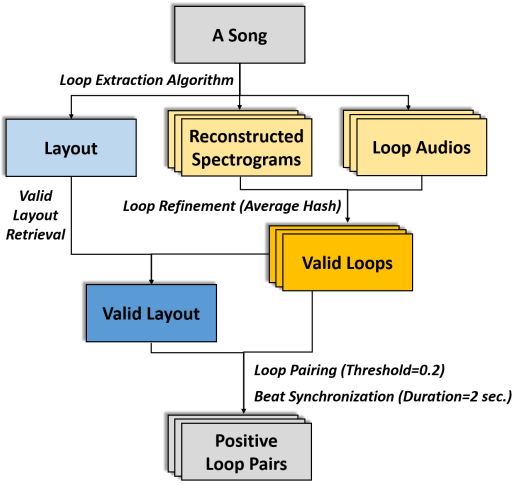

Figure 1: The proposed data generation pipeline for creating positive loop pairs.

3.2 Data Generation

Figure 1 depicts the overall data generation pipeline. We firstly use the algorithm proposed by Smith and Goto [5] for loop extraction. The algorithm uses NTF to model a song as a set of sound templates and rhythm templates, a set of "recipes" for combining these templates to recreate the loop spectrograms, and a loop layout that defines when each loop occurs in the song. In later work, the authors described how to choose the best instance of a loop to extract from a piece, and a second factorization step that can be applied to reduce the redundancy of the loops [11]. As a result of this process, we get out of a song the loop layout, the reconstructed spectrograms, and the audio files of loops, as depicted in the first half of Fig. 1.

Despite this extra step, we still found many redundant, similar-sounding loops from the Hip-Hop dataset. To further deduplicate the loops, we introduce a new loop refinement step. The main idea is to consider the reconstructed spectrograms obtained from the loop extraction algorithms as an image, and apply the average hash algorithm [17] used in the identification or integrity verification of images to detect the duplicate loops. We first construct the hash from each spectrogram extracted from the same song, then count the number of bit positions that are different in every pair of spectrograms. If a pair of spectrograms has fewer than five bit positions that are different, we regard them as duplicates and remove one of them.

After this, we refine the loop layout by two steps. First, we combine all activation values from the duplicate loop in the loop layout into a single activation value. Second, we normalize all the activation values in each bar. This leads a valid loop layout that is ready for loop pairs creation.

Finally, we have to process the real-valued loop layout to obtain pairs of loops that do co-occur. A straightforward approach is to threshold the loop layout; we found a threshold of 0.2 to work reasonably well. Please note that, to make all the loops in our dataset comparable, we time-stretch each to be 2 seconds long.

strictions, the songs are associated with 7 different Creative Commons licenses with various degree of freedom in data manipulation and redistribution. We will therefore build up our dataset into 7 groups, one for each unique license, before distributing the loops.

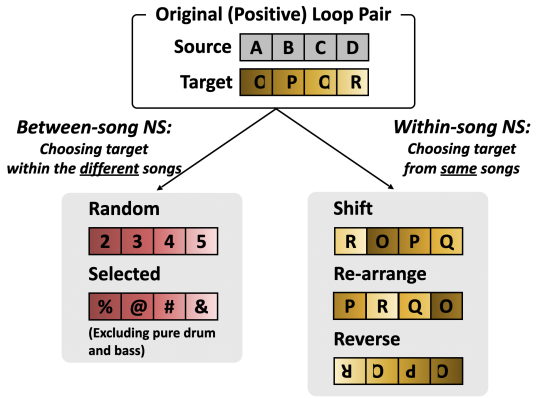

Figure 2: Illustration of various loop-pair 'negative sampling' (NS) strategies explored in the paper.

4. PROPOSED LEARNING METHODS

4.1 Negative Sampling Strategies

We get abundant positive data after the data generation pipeline. However, we also need negative examples (pairs of clips known to not match well) to train our mashability predictor. While there are straightforward ways to collect such examples, proper and domain-specific negative sampling has been shown to be important [18–22]. We experiment with the two types of methods of negative sampling. See Figure 2 for an overview of such methods.

4.1.1 Between-Song Negative Sampling

A naive approach to negative sampling, dubbed the **random** method, is to take any two loops from two different songs, and call that a negative pair. But, it is hard to ensure the loops collected in this way clash with one another.

We therefore also study a slightly more sophisticated method that takes *instrumentation* into account, dubbed the **selected** method. Specifically, we noted that many loops are pure drum or pure bass loops, and they tend to be compatible with any loops. Therefore, we use [23] to detect pure drum or bass loops and avoid selecting them in the process of random negative sampling. This way may help emphasize the harmonic compatibility of the loops. With this strategy, we can experiment whether putting more emphasis on *harmonic compatibility* can indeed improve the result, or other feature's compatibility is still crucial.

4.1.2 Within-Song Negative Sampling

Negative data can also be created by editing a loop in a positive loop pair. We come up with three methods for making conflicting loop pairs: 'reverse,' 'shift,' and 'rearrange.' Given a positive loop pair, we view one of them as the *source loop*, and the other as the *target loop*. With the **reverse** method, the target loop is played backward to create the rhythmically and harmonically conflict. The **shift**

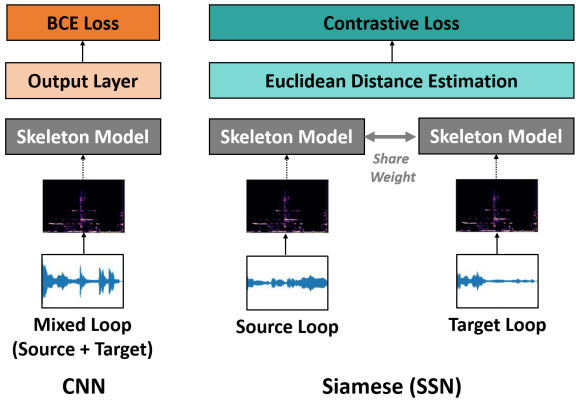

Figure 3: Architectures of the CNN and SNN models studied in this paper. The CNN takes a pair of loops (in the time domain) as the input, whereas the SNN takes the two loops as separate inputs and concatenate the intermediate abstractions only in the middle of its network.

method cycle-shifts a target loop by 1 to 3 beats at random to make the source loop and manipulated target loop unbalanced. The **rearrange** method cuts the target loop into beats and reorders them randomly. The combination of the source loop and the manipulated target loop is considered as a negative loop pair. See Figure 2 for illustrations.

4.2 Model and Training

Figure 3 shows the two models (CNN and SNN) proposed to learn the compatibility of two loops. To make a fair performance comparison, we train them with the same skeleton in a different way. Therefore, we first introduce the skeleton model and then explain how it is incorporated into the specific model architectures for the training process.

4.2.1 Skeleton model

A two-layer 2-D convolutional neural network (CNN) and 3-layer fully connected neural network is the skeleton of networks used in this paper. There are 16 filters (with kernel size 3) and 4 filters (also with kernel size 3) in the first and second convolutional layers. The 3-Layer fully-connected neural network is constructed by 256, 128, 16 output features, respectively. Furthermore, batch normalization [24], 10% dropout [25], and PReLU [26] are applied to all convolutional and fully-connected layers. All the models are trained using stochastic gradient descent with batch size 128. The input for the skeleton model is a 2-second loop audio. We compute the spectrograms by sampling the songs at 44k Hz and using a 2,048-point Hamming window and 512-point hop size. We then transform the spectrograms into 128-bin log mel-spectrograms. The resulting input has shape 173 frames by 128 bins.

4.2.2 CNN Model

A convolutional neural network is well-known for its ability as feature extraction. In our system, in order to learn the compatibility of two loops, we propose to use a CNN as a classifier by combining two loops into a single input, and then train the CNN model to learn to distinguish whether loop combinations would sound harmonious (high compatibility) or incompatible (low compatibility). For the classification purpose, we stack the skeleton model with a fully-connected output layer to get a single value as the output. Then, we compute the binary cross entropy loss (BCELoss) to update the parameters of the whole model. We note that its output is a value between 0 and 1, with values closer to 1 indicating a higher probability that the pair of loops are compatible, and closer to 0 when they are not. Therefore, we can later use its output to estimate the compatibility of any two loops.

4.2.3 Siamese Model

A Siamese neural network (SNN) [6] consists of twin networks that share weights and configurations. SNN has been shown effective in image retrieval [27] and various MIR tasks alike [4, 28–32]. Aiming to test its applicability to loop compatiblity estimation, we train an SNN using the labeled data we created through the data generation pipeline as follows. The outcome of an SNN is a mapping function from the input features to the output vectors in an embedding space. During the training process, our SNN first transforms the input Mel-spectrograms into a vector by a skeleton model and then optimize in contrastive loss [33] directly. After training, we have a mapping function that can map any loop to the embedding space. To select the compatible loops, we compute the Euclidean distance between two loops in that embedding space. If the distance for two loops is close, then we assume they may be compatible, and vice versa.

5. EXPERIMENTS AND EVALUATION

5.1 Objective Evaluation

In the objective evaluation, we aim to evaluate the performance of different combinations of model architectures (i.e., CNN and SNN) and negative sampling methods. In doing so, we consider two types of objective metrics.

The first type of evaluation entails a *classification* task. It assesses a model's ability to distinguish compatible loops from incompatible ones. To create a test set for this evaluation, we used the positive data from the validation data and collected the negative data by using all the negative sampling methods equally, in order not to favor any of them in this evaluation. To make the test set balanced, we set the ratio of negative to positive data to 1:1.

The second type of evaluation, on the other hand, involves a *ranking* task. Given a query loop and a certain number of candidate loops, a model has to rank the candidate loops in descending order of compatibility with the query. We created the set of candidate loops for a query loop such that we knew one and only one of the candidate loops formed a positive loop pair with the query loop. We could then evaluate the performance of a model by checking the position of this "target loop" in the ranked list. The closer the rank is to 1, the better. This evaluation task

Model	Negative sampling	Classification-based metric		Ranking-based metric			
		Accuracy	F1 score	Avg. rank	Top 10	Top 30	Top 50
CNN	Random	0.60	0.59	43.0	0.13	0.35	0.59
	Selected	0.59	0.59	43.1	0.13	0.29	0.62
	Reverse	**0.63**	**0.62**	41.2	0.19	0.42	0.62
	Shift	0.57	0.56	49.0	0.11	0.34	0.54
	Rearrange	0.57	0.57	47.7	0.10	0.31	0.57
Siamese NN	Random	0.51	0.47	**34.2**	**0.27**	**0.52**	**0.74**
	Selected	0.52	0.47	42.8	0.18	0.39	0.59
	Reverse	0.53	0.48	42.7	0.16	0.37	0.62
	Shift	0.53	0.52	43.0	0.16	0.41	0.65
	Rearrange	0.53	0.53	44.2	0.16	0.40	0.60

Table 2: Objective evaluation result of different combinations of models (CNN or SNN; see Section 4.2) and negative sampling strategies (see Section 4.1). We highlight the best result for each metric in bold.

aligns well with the real-world use case of the proposed model: to find loops compatible with a query loop from a pool of loop libraries. However, we note that it may not be always possible to rank the target loop high in this evaluation, because the other loops in the candidate pool may also be compatible with the query.

In our experiment, we set the number of candidate loops to 100. We computed four metrics for the ranking task: top-10 accuracy, top-30 accuracy, and top-50 accuracy—which evaluate whether the target loop was placed in the top-10, top-30, and top-50 of the ranked list, respectively—as well as the average rank. We report the average of these values for the 100 different query loops.

Classification Result—Table 2 shows that, regardless of the negative sampling method, the CNN models outperform the SNN models for the classification-based metrics. While the CNN learns to rate the compatibility of the loop combinations directly, the SNN learns to place the loops in a space, with nearby loops being more compatible. The training data were the same, so the difference in performance suggests that the space learned was not effective for the classification task.

Ranking Result—Table 2 also shows CNN and SNN models seem to perform comparably with respect to the ranking-based metrics. Yet, the best scores in the four ranking-based metrics are all obtained by SNN with 'random' negative sampling. When an SNN is used, there appears to be a large performance gap between 'random' and all the other negative sampling methods. This suggests that focusing on harmonic compatibility alone (e.g., as done by using the 'selected' negative sampling method) is not optimal; compatibility in other dimensions of music is also important. On the other hand, when a CNN is used, 'reverse' appears to perform the best among the five negative sampling methods.

5.2 Subjective Listening Test

We note that even if a model obtains the highest classification or ranking result, we still cannot guarantee that the model also works well in real-world applications. Accordingly, we deployed a user study by releasing an online questionnaire to get user feedback. As the users' time is precious, it is not possible to test the result of all combinations of the models and negative sampling methods. Therefore, we considered the result of the objective evaluation and picked five methods for subjective evaluation:

- 'CNN + reverse', 'SNN + random,' and 'SNN + reverse,' three instances of the proposed methods that performed well in the objective evaluation;
- 'AutoMashUpper' [2], as it represents the state-of-the-art for the task. Our implementation is based on the open source code from `https://github.com/migperfer/AutoMashupper`. As the rhythmic compatibility part of [2] is missing in this repo, we implement it ourselves following [2].
- 'Original,' which stands for the real loop combinations observed in FMA and extracted by the procedures described in Section 3.2. Specifically, an 'original' loop combination is one of the 100 positive loop pairs from the 'test set' listed in Table 1.

The loop combinations, or *test clips*, presented to users for evaluation are created as follows. As in Section 4.1.2, for each positive loop pair, we view one as the source loop, and the other as the target loop. Accordingly, we have 100 source loops and 100 target loops in the test set. A test clip is a combination of two loops. For the 'original' method, we pair the source loop with its true target. For the other methods, the 99 target loops (excluding the true target) are ranked, and the one predicted to match the source best is combined with it to form a test clip. The 5 resulting test clips for each source loop were considered as a *group*. This way, we have 100 groups in total. Among them, we picked 6 groups for evaluation: they have 2 vocal loops, 2 drum loops, and 2 loops of instrumental melody as the source loop, respectively.

We designed a subjective listening test that could be administered to users through an anonymous online questionnaire, and advertised it on social media websites related to EDM and Hip-Hop. A participant was randomly directed to a questionnaire containing one group of test clips, and was then asked to listen to the 5 test clips and rate each clip, on a 5-point Likert scale, in terms of: i) whether they sounded *matched*, ii) whether they were an *innova-*

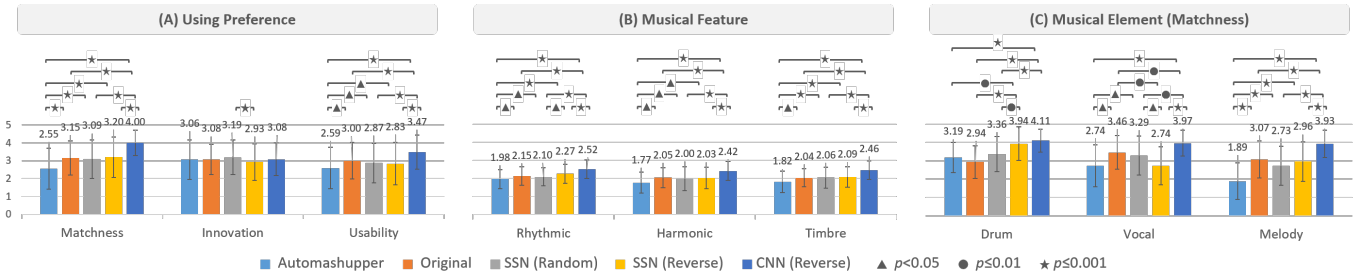

Figure 4: The subjective evaluation results of comparing the preference, musical feature and musical element among 5 methods. A method is considered to show low abilities to manipulate loops if its MOS is under 3 in (A) 'using preference'-related and (C) 'musical elements'-related metrics, and under 2 in (B) 'musical feature'-related metrics.

tive combination, iii) and whether they were *usable* in future compositions (see Figure 4A). They were also asked to indicate whether the loops matched according to 3 musical features: rhythm, harmony, and timbre (see Figure 4B). Finally, to see how consistent the models are at recommending loops given different query types, Figure 4C breaks down the results for Matchness (Figure 4A(i)) according to source loop type: drum, vocal or melody loop.

5.2.1 Subjective Evaluation Result

Data from 116 Taiwanese participants were collected, and all of them were included for analysis. The participants self-reported the gender they identified with (36 female, 80 male) and age (19–33). 50% said they listened to loop-based music more than 5 days a week, 54% had classical musical training, and 42% had experience composing loop-based music. Overall, the responses indicated an acceptable level of reliability (Cronbach's $\alpha = 0.778$).

Figure 4 indicates the mean opinion score (MOS). A Wilcoxon signed-rank test was conducted to statistically evaluate the comparative ability of the 5 methods.

Overall, we observe that AutoMashUpper performs least well in almost all the evaluation metrics. Outside of the *Innovation* (Figure 4A) and *Drum* metrics (Figure 4C), AutoMashUpper is significantly worse than almost all the other methods. This suggests that, in loop selection, considering only loop similarity and spectral balance (as assumed by AutoMashUpper) is not enough—perceptually compatible loops are not necessarily similar in content.

On the other hand, we note that 'CNN with reverse negative sampling' performs the best in almost all of the evaluation metrics. To our surprise, the loop combinations picked by the CNN are preferred even to the original loop combinations extracted from FMA songs. The performance difference between 'CNN + reverse' and 'Original' is significant (p-value < 0.001) in terms of several metrics, including *Matchness* and *Usability*. In contrast, there is no significant performance difference between 'Original' and the two SNN models.

One finding was surprising: The MOS obtained by 'CNN + reverse' for *Matchness* is 4.0, higher even than the MOS of 3.15 obtained by 'Original.' I.e., the loop combinations proposed by the system were found to match better than the original combinations. We are not sure how to account for this success. One might conjecture that the quality of the 'Original' pair suffers from the imperfect loop extraction algorithm, but since all the loops were extracted the same way, they should be on equal footing. Alternatively, the novelty of the non-original mash-ups could be more interesting to the listeners than the originals; however, there was no clear difference among the systems in terms of 'Innovation.' We can only conjecture that 'CNN + reverse' found better loop combinations than human-made ones because the model-based method could examine all the possible loop combinations, while humans cannot.

We found that SSN methods, despite their performing better in ranking-based objective metrics than the CNN method, did not create perceptually-better loop combinations. This suggests a discrepancy between the objective and subjective metrics. And lastly, while it seems fair to say that CNNs outperform SNNs in the subjective evaluation, it is hard to say whether 'reverse' is the most effective negative sampling, because we were not able to evaluate all the possible methods here. This is left as a future work.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a data generation pipeline and several negative sampling strategies to address the need of ground-truth labels for building a machine learning model for estimating the compatibility of loops. We have also implemented and evaluated two different network architectures (CNN and SNN) to build such models. Our objective evaluation shows that a CNN does well in classifying incompatible loop pairs and an SNN is good at imitating how producers combined loops, and our subjective evaluation suggests the loop combinations created by a CNN are favored over those created by an SNN and even, in some aspects, real data from FMA. Both CNNs and SNNs outperform the rule-based system AutoMashUpper.

We have two plans in place for future work. First, as we see some inconsistency between the results of objective and subjective evaluations, we plan to investigate other objective metrics for performance evaluation. Second, we plan to exploit the loop layouts estimated by the loop extraction algorithm [5] to study further the relationship between loops and their arrangement, which may aid in the automatic creation of loop-based music.

7. ACKNOWLEDGEMENT

This research was in part funded by a grant from the Ministry of Science and Technology, Taiwan (MOST107-2221-E-001-013-MY2).

8. REFERENCES

[1] T. Kitahara, K. Iijima, M. Okada, Y. Yamashita, and A.Tsuruoka, "A loop sequencer that selects music loops based on the degree of excitement," in *Proc. Int. Sound and Music Computing Conf.*, 2015.

[2] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "AutoMashUpper: Automatic creation of multi-song music mashups," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 22, no. 12, p. 1726–17370, 2014.

[3] J. B. L. Smith, G. Percival, J. Kato, M. Goto, and S. Fukayama, "CrossSong Puzzle: Generating and unscrambling music mashups with real-time interactivity," in *Proc. Int. Sound and Music Computing Conf.*, 2015.

[4] K. Lee and J. Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2019.

[5] J. B. L. Smith and M. Goto, "Nonnegativetensor factorization for source separation of loops in audio," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing.*, 2018.

[6] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *In Proc. Annual Conf. Neural Information Processing Systems*, 1994.

[7] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.

[8] S. Streich and B. S. Ong, "A music loop explorer system," in *Proc. Int. Computer Music Conf.*, 2008.

[9] Z. Shi and G. J. Mysore, "LoopMaker: Automatic creation of music loops from pre-recorded music," in *Proc. SIGCHI Int. Conf. Human Factors in Computing Systems*, 2018.

[10] P. Seetharaman and B. Pardo, "Simultaneous separation and segmentation in layered music," in *Proc. Int. Soc. Music Information Retrieval Conf.*, New York, NY, USA, 2016, pp. 495–501.

[11] J. B. L. Smith, Y. Kawasaki, and M. Goto, "Unmixer: An interface for extracting and remixing loops," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2019.

[12] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "AutoMashUpper: An automatic multi-song mashup system," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2013.

[13] C.-L. Lee, Y.-T. Lin, Z.-R. Yao, F.-Y. Lee, and J.-L. Wu, "Automatic mashup creation by considering both vertical and horizontal mashabilities," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2015.

[14] G. Bernardes, M. E. P. Davies, and C. Guedes, "Psychoacoustic approaches for harmonic music mixing," *Applied Science*, 2016.

[15] ——, "A perceptually-motivated harmonic compatibility method for music mixing," in *Proc. Int. Symp. Computer Music Multidisciplinary Research*, 2017, pp. 105–115.

[16] B. Xing, X. Zhang, K. Zhang, X. Wu, H. Zhang, J. Zheng, L. Zhang, and S. Sun, "Popmash: an automatic musical-mashup system using computation of musical and lyrical agreement for transitions," *Multimedia Tools and Applications*, 2020.

[17] C. Zauner, "Implementation and benchmarking of perceptual image hash functions," 2010, master Thesis. University of Applied Sciences Upper Austria.

[18] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. IEEE Int. Conf. Computer Vision*, 2015.

[19] F. Schroff, D. Kalenichenko, and J. Philbi, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015.

[20] F. Schroff, D. Kalenichenko, and J. Philbin, "Sampling matters in deep embedding learning," in *Proc. IEEE Int. Conf. Computer Vision*, 2017.

[21] R. Riad, C. Dancette, J. Karadayi, T. S. N. Zeghidour, and E. Dupoux, "Sampling strategies in siamese networks for unsupervised speech representation learning," in *Proc. Interspeech.*, 2018.

[22] J. Royo-Letelier, R. Hennequin, V. AnhTran, and M. Moussallam, "Disambiguating music artists at scale with audio metric learning," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2018.

[23] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," Late-Breaking/Demo ISMIR, 2019.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Machine Learning*, 2015.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," in *J. Machine Learning Research*, 2014.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Computer Vision*, 2015.

[27] G. Koc, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. Int. Conf. Machine Learning, ICML Deep Learning Workshop*, 2015.

[28] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.

[29] Y.-S. Huang, S.-Y. Chou, and Y.-H. Yang, "Generating music medleys via playing music puzzle games," in *Proc. Int. Conf. Artificial Intelligence*, 2018.

[30] U. Sandouk and K. Chen, "Learning contextualized music semantics from tags via a Siamese neural network," *ACM Trans. Intelligent Systems and Technology*, vol. 8, no. 2, 2016.

[31] P. Manocha, R. Badlani, A. Kumar, A. Shah, B. Elizalde, and B. Raj, "Content-based representations of audio using siamese neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2018, pp. 3136–3140.

[32] L.-C. Yu, Y.-H. Yang, Y.-N. Hung, and Y.-A. Chen, "Hit song prediction for Pop music by Siamese CNN with ranking loss," 2017, arxiv preprint: 1710.10814.

[33] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.