MelodySim: Measuring Melody-aware Music Similarity for Plagiarism Detection

Tongyu Lu *[1], **Charlotta-Marlena Geist***[2], **Jan Melechovsky**[1], **Abhinaba Roy**[1], **Dorien Herremans**[1]
[1]Singapore University of Technology and Design
[2]Otto von Guericke University Magdeburg
tongyu_lu@mymail.sutd.edu.sg, charlotta-marlena.geist@ovgu.de,
jan_melechovsky@mymail.sutd.edu.sg, abhinaba_roy@sutd.edu.sg, dorien_herremans@sutd.edu.sg

ABSTRACT

We propose **MelodySim**, a melody-aware music similarity model and dataset for plagiarism detection. First, we introduce a novel method to construct a dataset with focus on melodic similarity. By augmenting Slakh2100; an existing MIDI dataset, we generate variations of each piece while preserving the melody through modifications such as note splitting, arpeggiation, minor track dropout (excluding bass), and re-instrumentation. A user study confirms that positive pairs indeed contain similar melodies, with other musical tracks significantly changed. Second, we develop a segment-wise melodic-similarity detection model that uses a MERT encoder and applies a triplet neural network to capture melodic similarity. The resultant decision matrix highlights where plagiarism might occur. Our model achieves high accuracy on the MelodySim test set.

1. INTRODUCTION

In recent years, the popularity of generative music models has rapidly increased. With the rise of commercial models such as Suno[1] and Udio[2], as well as open source models like Mustango [1] and MusicGen [2], the question of artist-protection question arises. There currently is an ongoing discussion as well as legal battles on how artists should be compensated for the use of their music as training data [3], e.g. Recording Industry Association of America (RIAA) vs. Udio and Suno (June 2024)[3]. In addition, the music generated by these models might plagiarize the original training data. In this work, we develop tools that may help with melody-related plagiarism detection.

When generative models are trained on (often improperly licensed) copyrighted data, it becomes a strong possibility that the generated music plagiarizes the original training data. In particular, diffusion models have shown to be prone to replicate their training data, as shown by [4, 5] on the image generation task. Artists have made public outcries showcasing examples of their work or style replicated by generative models[4]. In literature, we noticed that generative AI models are typically evaluated in terms of their ability to predict similarly to the input data (accuracy) rather in terms of the originality of the generated output [6]. At the moment there is no clear legal precedent or ruling to tackle the copyright issues on the input data, however, we can examine resulting plagiarism by the output of the generative models.

Finding and confirming music plagiarism in general is a complex task. When deciding on plagiarism cases, [7] highlight the necessity of individually considering each case. An automatic plagiarism detection tool could help speed up the process of both flagging new plagiarism cases, as well as confirming expert opinions in existing lawsuits. Such tools might even be integrated into the music generation models themselves to avoid plagiarized output in the first case. This task, however, is not trivial, as there is no generally accepted, objective definition of what plagiarized music is. In an analysis of 17 lawsuits, [7] observed that the melody was prioritized when deciding on plagiarism, followed by the 'overall impression' of the music. This leads us to believe that there is a need for a melody-aware music similarity tool. The existing work on melody similarity metrics, however, is limited to the field of symbolic music (MIDI) [8–10]. To be able to deal with real-life court case data and generated music, we develop an audio based melody-aware similarity model in this work. This task is arguably more challenging than using symbolic music, due to the overlay of multiple audio signals, as well as the lack of data to train the model.

The contributions of this work include the creation of a novel dataset, MelodySim, which contains 1568 full length instrumental songs, with three additional variations per song, resulting in a total of 6,272 files. These variations contain slight melodic changes in terms of altered pitches, note durations, speed and instrumentation. These changes are subtle such that the resulting matching tracks can still be construed as 'plagiarized' in terms of melody. This data is then used to train a Triplet Neural Network with a MERT encoder [11], that minimizes the distance in representation of matching segments and maximizes the distance between different segments. This results in a melody-aware similarity embedding that is then used in a classification model to directly predict matching segments.

In the following sections, we first provide an overview of related work. Section 3 then describes how we have

*These authors contributed equally to this work.
[1]`https://suno.com`
[2]`https://udio.com`
[3]`https://shorturl.at/YlQ0P`
[4]`https://shorturl.at/j5tJb`

created the MelodySim dataset. This is followed by a description of the melody-aware triplet neural network that we developed to predict similar music fragments as well as the full-song plagiarism detection method. Finally, the results of our model on the training set as well as an in-the-wild plagiarism dataset are presented, followed by a conclusion.

2. RELATED WORK

In this section, we provide a brief overview of how music plagiarism has been defined in existing literature. We then discuss related work on music similarity detection models.

2.1 What is plagiarism?

When developing a model for plagiarism, we have to ask ourselves: *which elements of music count towards plagiarism?* A lot of popular rock, pop, and folk music shares the same 3-chord progression: I-IV-V, and has a similar drum track, making these elements non-eligible copyright infringement. This leaves other musical features such as melody and timbre as potential sources for plagiarism.

Currently, there does not exist a fixed rule set that defines plagiarism in music. In a study by [7], 17 music plagiarism lawsuits were analyzed. The authors observed that the melody was clearly prioritized when deciding on plagiarism, but always paired with another parameter which in most cases appeared to be the rhythm. Huber also stated that melody is the most discussed aspect in legal disputes, second to ‘overall impression’, which can be considered as the composition of various musical characteristics.

Based on this, we decided to build a melody-aware similarity metric, that not yet looks at the melody, but also encodes the music in general through MERT-features [11]. To achieve this, we carefully constructed a new dataset by thoroughly altering musical features in different levels of detail while maintaining the main melody, as explained in Section 3.

2.2 Automatic Music Similarity Detection

Most existing work on *melody* similarity detection is in the symbolic domain. Much of this work is not necessarily developed towards plagiarism detection, but could have other goals such as melody retrieval [8], repeated (exact) sample detection [12]. For a more comprehensive historic overview of music similarity models, the reader is referred to [13].

For instance, [8] developed a music similarity model for that was trained on the Meertens Tune Collections dataset [14]. Their recurrent neural network models allowed them to consider melody recommendation as a ranking problem of similarity. More recently, [15] present a way to generate an originality report, which includes an originality score (based on cardinality) to evaluate how much a generative symbolic music model copies from its training set. These metrics are then used to inform an early stopping mechanism that cuts of training when the optimal level or ‘originality’ is reached on the validation set, thus preventing the transformer from generating music that is too similar to its training data.

In [9], an image-based approach for solving the task of plagiarism detection based on musical features such as rhythm and melody similarity. The authors used the Lakh MIDI dataset [16] and represented the MIDI into 8-bar units and grayscale images. Generated simulated plagiarism cases were then generated by reversing and removing operations on note and rhythm vectors as well as note sequences. This work only considered monophonic instrumental songs.

An interesting in-the-wild dataset for plagiarism in the Music Copyright Infringement Cases (MCIC) dataset [10]. The dataset contains music pairs from 116 court cases (denied: 66, infringed: 32, settled: 18) in both MIDI as well as score form.

In the domain of *audio* similarity research, [17] and [18] developed similarity techniques based on spectrograms and fingerprinting to tackle plagiarism detection. These methods require high computational power with a large fingerprint database and tend to result in low accuracy with decreasing audio quality and higher noise level [9, 12]. The resulting similarity relies on general acoustic features extracted from spectrograms and does not directly distinguish between specific musical characteristics like melody, rhythm or timbre.

Another audio-based similarity approach is the Music Replication Assessment (MiRA) tool [19], which includes several similarity metrics for raw audio. In their experiments, embedding-based metrics showed the most promising results in terms of robustness and sensitivity. The scope of their work, however, is limited to exact replications in music audio. Their dataset was generated by putting a fraction of a reference track into a random point of a target track.

The problem of finding reused samples in other songs was tackled in [12]. Their deep learning approach uses a siamese-based convolutional neural network (CNN) with mel-spectrograms and a triplet loss. Their similarity score based on the resulting embeddings consisted of a combination of Euclidean distance, cosine similarity and the Pearson correlation. The model was trained on the WhoSampled[5] dataset. The task of finding replicated samples is also limited to finding exact repetitions. In this work, we aim to improve upon such an approach by including note-level variations to make the algorithm more robust.

Our work build upon the gaps in literature by providing the first open, large-scale synthetic audio dataset for audio plagiarism. Each song contains three variations with slight music theory-informed melody changes, that contains many small melody variations (altered pitches, note durations, speed and instrumentation) while significantly altering the other tracks and timbre. The subtleness of the melody changes ensures that the paired tracks may be construed as plagiarized. This new dataset then allows us to train a triplet neural network-based melody-aware similarity model for plagiarism detection directly on audio.

[5] www.whosampled.com

3. MELODYSIM DATASET

In order to be able to create a strong melody-aware embedding for audio music, we need a suitable dataset to train on. We used various MIDI and audio augmentations to create **MelodySim**, a new audio dataset which contains three variations for each song. These variations aim to keep the melody constant (except for tiny changes for robustness), and change other aspects such as removing tracks, changing instruments, inverting chords, changing the tempo and transposing the composition as shown in Figure 1. We thus aim to capture melodic similarity between otherwise different songs, as melody is one of the main plagiarism criteria [7].

We used 1568 MIDI files from the Slakh2100 dataset [20] as a base dataset to start the augmentations. In the following subsections the augmentation procedure is described in detail. The final dataset consists of 6,272 full-length audio music files, consisting of original pieces with three additional versions for each piece. The dataset and augmentation code are available online[6].

3.1 Step 1 - Melody track identification

For each of the multi-track MIDI files, we first identify the melody track by training a machine learning model. Our best performing model is a gradient boosting classifier model following the approach presented in [21]. A refined CMU Computer Music Analysis Dataset[7] was used for training the model, where we manually relabeled a portion of this dataset after noticing a number of incorrect labels. The refined dataset is available online[8].

Taking [21] as a reference, a number of adjustments were made to the input features, that lead to improved results. First, additional track features including polyphony rate and note activation density were added. Secondly, apart from the features from the current track under inspection, average features of other tracks in the same MIDI file were also computed and added to the classification inputs as reference-features.Through cross-validation, finally a histogram-based gradient boosting model was selected as our model, which reached an accuracy of 97% on the validation split of CMU. Through manual inspections, we found that the model generalized well on Slakh2100. Our melody track identification model is available as open source[9].

3.2 Step 2 - MIDI-level augmentations

Now that we have identified the melody track in Step 1, we are able to perform a number of MIDI augmentations on both the instrument- and note-level.

Instrument replacement: For each of the MIDI tracks a new instrument are considered. We first group the MIDI instrument indices (from 1 to 128) into ensembles (pianos, guitars, high-register strings, low-register strings, etc.), and then reassign the track instruments with the following rules: 1. with probability 0.2, retain the instrument as it is; 2. if not, then with probability 0.7, change the instrument to another one in its ensemble (e.g., replacing piano with e-piano); 3. otherwise replace the instrument with another one in a different ensemble with similar pitch register; 4. ensure coupled tracks (e.g., piano tracks) to be applied with the same replacement policy; 5. avoid different instrument tracks being replaced into the same instrument.

Track removal: 1. with a probability drawn from a uniform distribution of [0.1, 0.5], for each track, mute the track; 2. with a probability of 0.5, mute the percussion track; 3. never mute the melody tracks (identified), bass tracks and other important tracks (vocals, piano or guitar companies).

Note splitting: With a probability P_n, split the current note of typical duration (whole notes, half notes, quarter notes) into two of half the original duration. P_n is drawn from a uniform distribution of [0.3, 0.85] for each track n.

Chord inversion: For each track, detect block chords (concurrently played notes) consisting of 3 or 4 notes. For each such chord, with a probability P_n, shift the top notes an octave down or the bottom notes an octave up. P_n is drawn from a uniform distribution of [0.3, 0.85] for each track n.

Chord argpeggiation: For each track, detect block chords that are in regular durations (1x/2x/3x/4x of quarter note). With a probability P_n, split the chord into an arpeggio (consisting of equally-placed chord notes) with the same total duration as the original chord. P_n is drawn from a uniform distribution of [0.3, 0.85] for each track n.

3.3 Step 3 - Audio-level augmentations

After augmenting the MIDI files, the resulting audio files are obtained by synthesizing with the Musyng soundfont. Then, a set of audio augmentations (as depicted in Figure 1) is applied to further diversify the different versions, in particular:

- **Pitch shift:** The audio track is pitch-shifted by a random integer of semitones in the range of [-4, 4].
- **Time shift:** The whole track is shifted by a random time from a range of [-3, 3] seconds. This time shift is used when matching the positive pairs later on.
- **Tempo change:** The audio track's tempo is altered by a random factor in the range of [0.9, 1.1].

The resulting audio files are then cut into 10 sec long segments each being saved with representative track name, version index, and segment index.

4. MUSIC SIMILARITY MODEL

Using the newly created MelodySim, we train a triplet neural network model [22] that that enables the creation of melody-sensitive embeddings of music audio, and the computation of the distance or similarity between these embeddings.

[6]https://huggingface.co/datasets/amaai-lab/melodySim
[7]https://www.cs.cmu.edu/music/data/melody-identification/
[8]https://huggingface.co/datasets/amaai-lab/melodySim
[9]https://huggingface.co/amaai-lab/MelodySim

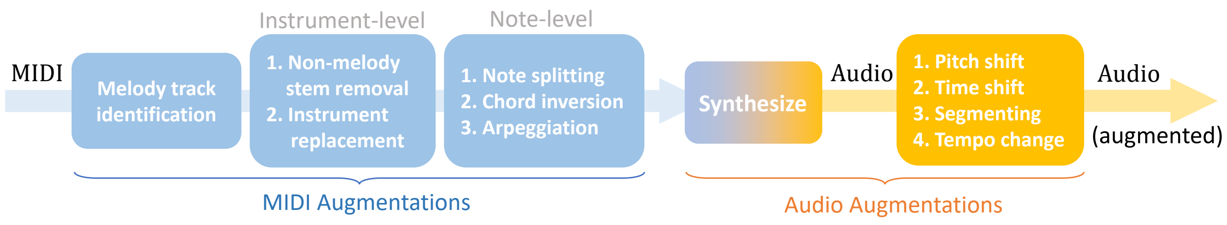

Figure 1. The proposed melody-aware augmentation pipeline used for constructing MelodySim dataset by augmenting Slakh MIDI.

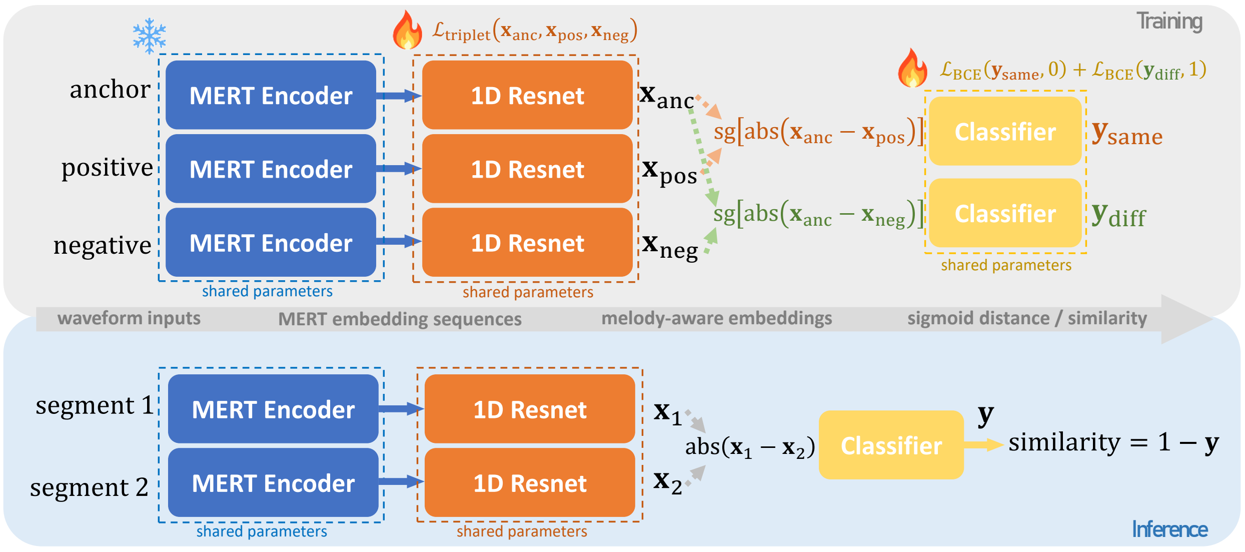
Training
anchor
positive
negative
MERT Encoder
MERT Encoder
MERT Encoder
shared parameters
$\mathcal{L}_{\text{triplet}}(\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{pos}}, \mathbf{x}_{\text{neg}})$
1D Resnet
1D Resnet
1D Resnet
shared parameters
$\mathbf{x}_{\text{anc}}$
$\mathbf{x}_{\text{pos}}$
$\mathbf{x}_{\text{neg}}$
$\text{sg}[\text{abs}(\mathbf{x}_{\text{anc}} - \mathbf{x}_{\text{pos}})]$
$\text{sg}[\text{abs}(\mathbf{x}_{\text{anc}} - \mathbf{x}_{\text{neg}})]$
$\mathcal{L}_{\text{BCE}}(\mathbf{y}_{\text{same}}, 0) + \mathcal{L}_{\text{BCE}}(\mathbf{y}_{\text{diff}}, 1)$
Classifier
Classifier
$\mathbf{y}_{\text{same}}$
$\mathbf{y}_{\text{diff}}$
shared parameters
waveform inputs
MERT embedding sequences
melody-aware embeddings
sigmoid distance / similarity
segment 1
segment 2
MERT Encoder
MERT Encoder
shared parameters
1D Resnet
1D Resnet
shared parameters
$\mathbf{x}_1$
$\mathbf{x}_2$
$\text{abs}(\mathbf{x}_1 - \mathbf{x}_2)$
Classifier
$\mathbf{y}$
$\text{similarity} = 1 - \mathbf{y}$
Inference

Figure 2. The proposed architecture for training and inference. $\text{sg}[\cdot]$ means "stop gradient" and $\text{abs}(\cdot)$ notates element-wise absolute function.

4.1 Triplet dataset

To train a triplet neural network model, we reformulate MelodySim into triplets, consisting of an anchor sample, a positive sample similar to the anchor, and a negative sample dissimilar to the anchor. We construct the positive pairs by combining time-aligned segments from the original and augmented tracks. The negative pairs are formed using inter-song segments.

The example below illustrates a triplet $(\text{anchor}, \text{positive}, \text{negative})$ structure:

$$\begin{aligned} \text{anchor} &= \text{Track}_{00125}/\text{version}_0/\text{segment}_{02}, \\ \text{positive} &= \text{Track}_{00125}/\text{original}/\text{segment}_{02}, \\ \text{negative} &= \text{Track}_{00007}/\text{version}_2/\text{segment}_{12}. \end{aligned}$$

Each triplet consists of an anchor data sample, a positive data sample that shares the same melody but varies in other characteristics (such as texture, tempo, or instrumentation), and a negative data sample that differs in both melody and other features. This triplet construction ensures the model can learn to differentiate between similar and dissimilar musical excerpts based on melody.

4.2 Triplet Neural Network

As shown in Figure 2, the music similarity model is a triplet neural network (TNN) consisting of a MERT encoder, a ResNet backbone and a classifier head.

The similarity model starts with a MERT encoder, a pretrained state of the art model open source on huggingface by [11]. For capacity limitation reasons the audio files were fed into the more compact `MERT-v1-95M` version of the feature extractor and stored as encodings, before using them as input for the adaption network in the training process. In order to reduce memory load the output features of MERT were postprocessed with a moving average with `size=10, stride=10` over the time token axis and the selected hidden states were limited to h_3, h_6, h_9, h_{12}.

After MERT encoding, a sequence of trainable 1D convolutional residual blocks is applied as an adaption network. An average pooling layer is applied at the end of 1D Resnet to aggregate the information over time dimension, getting a fixed dimension embedding for its corresponding MERT embedding sequence. In a training step, all three components in a triplet $(\text{anchor}, \text{positive}, \text{negative})$ run through the MERT encoder (parameters frozen) and the 1D Resnet (parameters trainable), getting the corresponding embeddings, i.e., $\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{pos}}, \mathbf{x}_{\text{neg}}$. To integrate melody-aware information, we update the 1D Resnet parameters through backward propagation with triplet loss, which is defined as follows:

$$\begin{aligned}&\mathcal{L}_{\text{triplet}}(\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{pos}}, \mathbf{x}_{\text{neg}})\\&= \max\left(d(\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{pos}}) - d(\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{neg}}) + \alpha, 0\right)\end{aligned}$$

where $\alpha = 1.0$ is the margin, $d(x_i, y_i) = \|x_i - y_i\|_2$ is the Euclidean distance.

Finally, a fully-connected classifier is maintained at the end in measuring the sigmoid distance between embeddings, with output scaling in range $[0, 1]$. In each triplet, we inspect a "same case", namely $(\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{pos}})$ and a "different case" $(\mathbf{x}_{\text{anc}}, \mathbf{x}_{\text{neg}})$. The classifier takes $\text{abs}(\mathbf{x}_{\text{anc}} - \mathbf{x}_{\text{pos}})$ and $\text{abs}(\mathbf{x}_{\text{anc}} - \mathbf{x}_{\text{neg}})$ as inputs, giving $\mathbf{y}_{\text{same}}, \mathbf{y}_{\text{diff}}$ as outputs respectively. To train the classifier, we backward propagate the Binary Cross Entropy (BCE) loss

$$\begin{aligned}&\mathcal{L}_{\text{BCE}}(\mathbf{y}_{\text{same}}, 0) + \mathcal{L}_{\text{BCE}}(\mathbf{y}_{\text{diff}}, 1)\\&= \text{mean}(\log(1 - \mathbf{y}_{\text{same}}) + \log \mathbf{y}_{\text{diff}})\end{aligned}$$

In this way, we train the classifier with balanced labels. In addition, we remove the gradient of classifier inputs (i.e., $\text{abs}(\mathbf{x}_{\text{anc}} - \mathbf{x}_{\text{pos}})$ and $\text{abs}(\mathbf{x}_{\text{anc}} - \mathbf{x}_{\text{neg}})$) during training to avoid BCE loss confusing the 1D Resnet. In Figure 2 we use $\text{sg}[\cdot]$ to show the "stop gradient" operation.

During inference, we utilize the similarity model as a Siamese Neural Network: forwarding both input audio segments with MERT encoder and 1D Resnet respectively in the same manner, getting the absolute difference and finally get the classification result. Note that the final output of the inference pipeline is similarity (also falls in range $[0, 1]$) instead of sigmoid distance.

4.3 Plagiarism identification

Note that the TNN mentioned in the previous section computes similarity between two music segments. However, it remains a problem to decide whether or not two entire pieces are plagiarized. In view of this, we compute the similarity matrix and design a rule-based decision strategy.

Given two pieces, we segment them into 10-sec windows $[\mathbf{w}_1^{(i)}, i = 1, ..., N_1], [\mathbf{w}_2^{(j)}, j = 1, ..., N_2]$ in the same way as when constructing the MelodySim dataset. If we notate the similarity model (inference mode) with

$$s_{ij} = f(\mathbf{w}_1^{(i)}, \mathbf{w}_2^{(j)})$$

we can finally get a similarity matrix

$$\mathbf{S} = [s_{ij}] \in [0, 1]^{N_1 \times N_2}$$

Next, we threshold (default $\gamma = 0.5$) the similarity matrix, getting a decision matrix

$$\mathbf{D} = u(\mathbf{S} - \gamma)$$

where $u(\cdot)$ is the unit step function.

Summing up rows or columns of $\mathbf{D}$, we have plagiarized counts in both directions, namely

$$\mathbf{d}_{1\to2}^{(i)} = \sum_j \mathbf{D}_{ij}, \quad \mathbf{d}_{2\to1}^{(j)} = \sum_i \mathbf{D}_{ij}$$

If we further define a sensitivity (how many similar segments in piece 2 are enough to determine that the segment in piece 1 is plagiarized and vice versa), we can finally obtain the proportion of plagiarized segments in $\mathbf{d}_{1\to2}, \mathbf{d}_{2\to1}$. In our testing cases, we set the maximum proportion to be 0.2, which means that "if both pieces have number of plagiarized segments larger than 20% of the total segments, then the two pieces have plagiarism relationship".

5. EXPERIMENTS

5.1 Experimental setup

The similarity model was trained on 95% of the MelodySim dataset, reserving the remaining 5% for evaluation purposes. The training process was executed on a single Nvidia V100 GPU for a duration of 7.3 hours, with a batch size of 512. During the training phase, all MelodySim training tracks were traversed, and anchors were randomly selected along with their corresponding positive and negative samples. To enhance diversity, each track was loaded 4 times per epoch. The training regimen encompassed a total of 797 epochs.

To thoroughly test the model, we utilize the 78 pieces from the test split. Specifically, we construct $546 = 7 \times 78$ positive pairs , where the factor 7 comes from all combinations among versions along with self-comparison, i.e., $\{(\text{orig}, \text{orig}), (\text{orig}, \text{ver1}), (\text{orig}, \text{ver2}), ..., (\text{ver2}, \text{ver3})\}$. Correspondingly, we select an equal number of negative pairs to maintain a balanced test set. These negative pairs are formed by combining excerpts from different tracks and randomly sampling from all possible combinations.

5.2 Objective evaluation

We performed objective evaluation of the melody-similarity classifier that detects positive pairs in the dataset. We present the similarity matrices between selected examples and report the classification metrics on the test set. A selection of similarity matrices is depicted in Figure 3.

Table 1 shows the classification results on the test set.

Table 1. Classification metrics on test split.

	Precision	Recall	F1
Different	1.00	0.94	0.97
Similar	0.94	1.00	0.97
Average	0.97	0.97	0.97
Accuracy			0.97

The similarity matrix reveals that the model effectively captures melodic similarity, accurately reflecting the proximity between music audio segments. However, we notice that the positive pairs tend to have large-scale activations like Figure 3 shows. This shows that the model may not only be sensitive in melody, but also the music texture (if the model is only sensitive in melody, then the positive pair similarity matrix should present the repeating pattern). In addition, some of the negative pair similarity

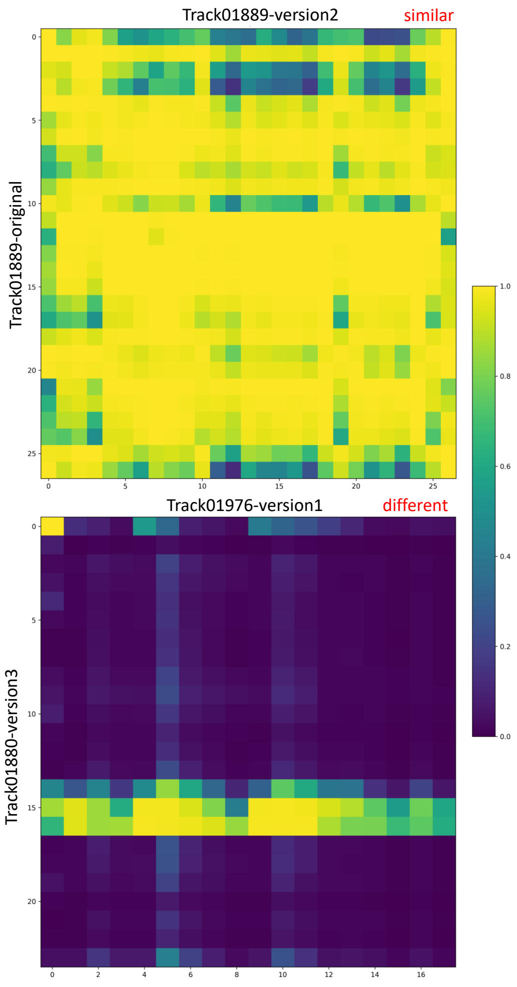

Figure 3. Examples of similarity matrices, a positive pair (top) and a negative pair (bottom) from the test split are demonstrated.

matrices shows horizontal or vertical activations, meaning that "one or several adjacent segments in piece 1 may be similar to all windows in piece 2", which is not likely in real case. This reflects to some inherent problem on the similarity model as a black box. Observed from the classification metrics, we would say that our model fits well on MelodySim, reaching 97% accuracy as well as F1 score, which indicates that the detection on positive pairs and negative pairs is balanced.

5.3 Subjective evaluation of dataset

To assess the efficiency of our MelodySim dataset, we conducted a listening study. A total of 12 participants listened to 12 audio pairs and rated the overall similarity, melodic similarity, and similarity of non-melodic aspects on a 7-point Likert scale [23]. The results, depicted in Table 2, confirm that the proposed augmentations mainly alter non-melodic aspects of the music.

Table 2. MelodySim dataset listening study results; aspects are rated on a 7-point Likert Scale; reported Mean Opinion Score with 95% Confidence interval.

Aspect	Positive pairs	Negative pairs
Overall similarity	4.23 ± 0.80	2.00 ± 0.68
Melodic similarity	4.53 ± 0.84	1.90 ± 0.90
Non-melodic similarity	3.94 ± 0.53	2.27 ± 0.22

6. DISCUSSION AND LIMITATIONS

The task of targeted augmentation to preserve melody but alter other attributes is not simple due to a few factors. First, identifying melody is not always straightforward, as some files include multiple melody tracks, or have melody being played in some parts of the song by an otherwise non-melodic track. This makes it difficult to craft a simple rule for melody identification, which could sometimes result, for instance, in a part of the melody missing, or a non-melody track being treated as a melody track, thus being always present after passing through the augmentation pipeline. Furthermore, melody identification rules can be genre-dependent. In this paper, we offer a good baseline melody identification model, which can be further improved in future work.

When constructing positive and negative pairs, we did not consider the possibility of pairing two segments from the same song at different time marks. The probability of a repeating motive in the same song is too high and would require a similarity metric to automatically identify such similar segments. However, using segments of the same song as either positive pairs (with matching melody, but slightly varied background, for instance, when the song culminates vs when it starts), or as negative pairs (when the melody played is different, e.g., verse vs chorus), would benefit the training of the similarity model further.

Future work will focus on further augmentation improvement, and include more analyses, potentially with real-life plagiarism cases.

7. CONCLUSION

We present the MelodySim dataset, an open source audio dataset and model for melody-aware music similarity and plagiarism detection. MelodySim was constructed through a set of targeted midi and audio augmentations such that it contains original tracks as well as three variations that have a comparable melody, but vary in terms of other musical aspects (arpeggiated chords, changed instruments, missing tracks, etc.). The similarity in terms of melody and other musical aspects was verified through a listening study. We also propose a melody-aware similarity model. This model consists of a MERT-encoder, combined with a ResNet backbone and classification head. We employ a triplet neural network architecture for training the model on the MelodySim dataset. In an objective evaluation, we show that the model performs admirably in detecting variations of songs in the test set.

8. ACKNOWLEDGMENT

This work has received support from SUTD's Kickstart Initiative under grant number SKI 2021 04 06 and MOE under grant number MOE-T2EP20124-0014.

9. REFERENCES

[1] J. Melechovsky, Z. Guo, D. Ghosal, N. Majumder, D. Herremans, and S. Poria, "Mustango: Toward controllable text-to-music generation," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 8286–8309.

[2] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 704–47 720, 2023.

[3] M. Wei, M. Modrzejewski, A. Sivaraman, and D. Herremans, "Prevailing research areas for music ai in the era of foundation models," *arXiv preprint arXiv:2409.09378*, 2024.

[4] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein, "Diffusion art or digital forgery? investigating data replication in diffusion models," in *Proc. of the IEEE/CVF Conf. on computer vision and pattern recognition*, 2023, pp. 6048–6058.

[5] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," in *32nd USENIX Security Symposium*, 2023, pp. 5253–5270.

[6] B. L. Sturm, M. Iglesias, O. Ben-Tal, M. Miron, and E. Gómez, "Artificial intelligence and music: open questions of copyright law and engineering praxis," in *Arts*, vol. 8, no. 3. MDPI, 2019, p. 115.

[7] J. Huber, D. Müllensiefen, and R. Kopiez, "Von der" armseligen allerweltsfloskel" zur" schöpferischen eigenart": Eine analyse deutscher gerichtsentscheidungen zu plagiaten in der musik von 1966 bis 2020," Ph.D. dissertation, Hochschule für Musik, Theater und Medien Hannover, 2020.

[8] F. Karsdorp, P. van Kranenburg, and E. Manjavacas, "Learning similarity metrics for melody retrieval," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, 2019, pp. 478–485.

[9] K. Park, S. Baek, J. Jeon, and Y.-S. Jeong, "Music plagiarism detection based on siamese cnn," *Hum.-Cent. Comput. Inf. Sci*, vol. 12, pp. 12–38, 2022.

[10] S. Park, H. Kim, J. Pak, and J. Kim, "Quantitative analysis of melodic similarity in music copyright infringement cases," in *International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval, 2024.

[11] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Xiao, C. Lin, A. Ragni, E. Benetos *et al.*, "Mert: Acoustic music understanding model with large-scale self-supervised training," *arXiv:2306.00107*, 2023.

[12] G. Kasif and G. Thondilege, "Exploring music similarity through siamese cnns using triplet loss on music samples," in *2023 Int. Research Conf. on Smart Computing and Systems Engineering (SCSE)*, vol. 6. IEEE, 2023, pp. 1–8.

[13] P. Knees and M. Schedl, "A survey of music similarity and recommendation from music context data," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 10, no. 1, pp. 1–21, 2013.

[14] P. Van Kranenburg, M. de Bruin, L. P. Grijp, and F. Wiering, "The meertens tune collections," *Meertens Online Reports*, vol. 2014, no. 1, 2014.

[15] Z. Yin, F. Reuben, S. Stepney, and T. Collins, ""'a good algorithm does not steal–it imitates": The originality report as a means of measuring when a music generation algorithm copies too much," in *Artificial Intelligence in Music, Sound, Art and Design: 10th Int. Conf., EvoMUSART 2021, Part of EvoStar*. Springer, 2021, pp. 360–375.

[16] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset." in *Ismir*, vol. 2, no. 9, 2011, p. 10.

[17] N. Borkar, S. Patre, R. S. Khalsa, R. Kawale, and P. Chakurkar, "Music plagiarism detection using audio fingerprinting and segment matching," in *2021 Smart Technologies, Communication and Robotics (STCR)*. IEEE, 2021, pp. 1–4.

[18] A. López-García, B. Martínez-Rodríguez, and V. Liern, "A proposal to compare the similarity between musical products. one more step for automated plagiarism detection?" in *Int. Conf. on Mathematics and Computation in Music*. Springer, 2022, pp. 192–204.

[19] R. Batlle-Roca, W.-H. Liao, X. Serra, Y. Mitsufuji, and E. Gómez Gutiérrez, "Towards assessing data replication in music generation with music similarity metrics on raw audio," 2024.

[20] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 45–49.

[21] Z. Jiang and R. B. Dannenberg, "Melody track identification in music symbolic files." in *16th Sound and Music Computing Conf. (SMC), Málaga, Spain*, 05 2019.

[22] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” in *Similarity-based pattern recognition: third international workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*. Springer, 2015, pp. 84–92.

[23] A. Joshi, S. Kale, S. Chandel, and D. K. Pal, “Likert scale: Explored and explained,” *British journal of applied science & technology*, vol. 7, no. 4, p. 396, 2015.