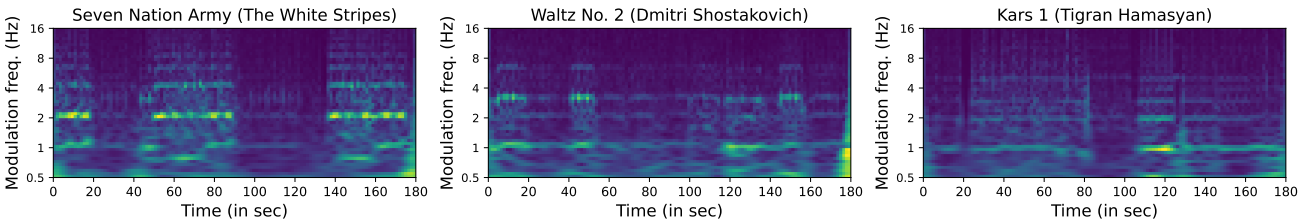



**Figure 1**: Examples of the Constant Q Fluctuation Patterns (CQFP).

## 3. PROPOSED METHOD

In this section, we describe and motivate our design choices. We first present how we extracted our rhythmic and lyrics features, and publicly release our datasets [1] . We then present our metric learning-based VI model.

### 3.1 Rhythmic features

Our assumption is that the FP representation described in Section 2.2 displays both the local rhythmic patterns along the frequency modulation axis, as well the global rhythmic evolution of the piece over time. We therefore propose to use this representation as our rhythmic feature for version identification. However, we introduce a CQT to compute the periodicities, and to achieve tempo invariance along the frequency modulation dimension. We choose as minimum frequency for the CQT 0.5 Hz (i.e. 30 bpm which we assume would be the tempo of the slowest tracks), and to cover up to 5 octaves i.e. 16Hz (960 bpm) with 10 bins per octave to get all the rhythmic subtleties. We kept the same other parameters as in the original implementation [16].

As an illustration, Figure 1 represents the CQ-FP obtained for different tracks with characteristic time signature. The first example has a 4/4 time signature, and clearly displays a rhythmic pattern present around 2 Hz (~120 bpm), as well as another periodicity around 4 Hz, which corresponds to a clear binary rhythm. The second example has a 3/4 time signature, and the rhythm of a waltz appears clearly: one beat at 1 Hz (60 bpm) and another at about 3 Hz. Finally, the third example has a 5/4 time signature (irregular), and we find this characteristic by observing bands at 1, 2 and 3 Hz. It can also be observed on each example that our rhythmic feature also represents the global structure of the piece over time, which is probably another relevant aspect in a VI context. Finally, as the modulation frequency dimension has a constant Q-factor, a change in tempo would not change the spacing between activations.

### 3.2 Lyrics features

We argue that accurate lyrics recognition is not required for version identification, and that identifying only a few common words, or even a few common character sequences, between tracks shall be sufficient to determine whether they are versions or not. We thus implemented a deliberately simple fully convolutional ALR system inspired by recent ASR system [21, 27].

[1] https://ircam-anasynth.github.io/papers/2022/abrassart

**Model** It is an 8-layers 2D-CNN with 3x3 kernels. Max-pooling 2x2 is applied on the first 2 layers to decrease time and frequency dimensions, and max-pooling 1x2 is applied on the next 6 layers only for frequency dimension. The first layer has 64 filters, doubled at each layer up to 512. A dropout with rate 0.3 is applied.

**Inputs/outputs** We use no pre-processing on the audio (no data augmentation, no voice separation). As classically done in ASR, we use 40 band Mel-spectrogram with 10ms timeframes as input feature. The model outputs a posteriorgram corresponding to the log-probabilities of the 'a...z' letters, the space symbol and the CTC blank symbol, i.e. 28 bins in total. An output example is shown Figure 2.

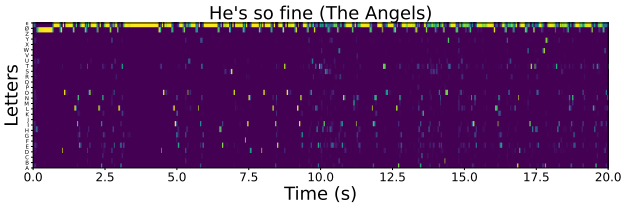



**Figure 2**: A posteriorgram obtained for 20 sec. of audio.

**Training and inference** The model is trained on the Dali dataset, which provide 12k+ polyphonic audio with lyrics annotations at the word level. We used audio chunks of 10 seconds, and used a CTC loss with Adam optimizer to train the model to align between audio and text. We evaluated its performances on a distinct Dali test subset using a greedy beam search decoding with no language model, and achieved a modest Character Error Rate (CER) of 0.496.

### 3.3 Convolutional architecture

The same simple architecture (with different configurations) has shown its ability to capture relevant melodic or harmonic similarity between versions [3,4]. It consists in a plain 5-layers Convolutional Neural Network (CNN), encoding each input feature using time and frequency max-pooling, while increasing the number of filters at each layer. The CNN output feeds a gated temporal attention mechanism [28], which was found to help the model to focus on relevant portions of the input features. We refer the reader to our previous work [8] for implementation details.

In this work, we keep the exact same generic architecture, and propose two new configurations to process also the rhythmic and lyrics features. We summarize the configuration used for each of the four features in Table 1.

| Features | Melodic [3] | | | | | | Harmonic [4] | | | | | | Rhythmic | | | | | | Lyrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Layers | n | k | ks | p | ps | d | n | k | ks | p | ps | d | n | k | ks | p | ps | d | n | k | ks | p | ps | d |
| 1 | 64 | 3x3 | 1x1 | 2x2 | 2x2 | 0.0 | 256 | 180x12 | 1x1 | 1x12 | 1x1 | 0.0 | 64 | 3x20 | 1x1 | 1x2 | 1x2 | 0.4 | 64 | 10 | 1 | 5 | 2 | 0.3 |
| 2 | 128 | 3x3 | 1x1 | 2x2 | 2x2 | 0.1 | 256 | 5x1 | 1x1 | 1x1 | 1x1 | 0.0 | 128 | 3x3 | 1x1 | 1x2 | 1x2 | 0.3 | 128 | 10 | 1 | 5 | 2 | 0.2 |
| 3 | 256 | 3x3 | 1x1 | 2x2 | 2x2 | 0.1 | 256 | 5x1 | 1x1 | 1x1 | 1x1* | 0.0 | 256 | 3x3 | 1x1 | 1x2 | 1x2 | 0.2 | 256 | 10 | 1 | 5 | 2 | 0.1 |
| 4 | 512 | 3x3 | 1x1 | 2x2 | 2x2 | 0.2 | 512 | 5x1 | 1x1 | 1x1 | 1x1 | 0.0 | 512 | 3x3 | 1x1 | 1x2 | 1x2 | 0.1 | 512 | 10 | 1 | 5 | 2 | 0.1 |
| 5 | 1024 | 3x3 | 1x1 | 2x2 | 2x2 | 0.3 | 512 | 5x1 | 1x1 | 1x1 | 1x1* | 0.0 | 1024 | 3x3 | 1x1 | 1x2 | 1x2 | 0.0 | 1024 | 10 | 1 | 5 | 2 | 0.0 |

**Table 1**: Configuration of the 5-layers CNN used for the features. n : number of filters, k : kernel size, ks : kernel stride, p : pool size, ps : pool stride, d : dropout. 1st dimension is time, 2d dimension is frequency. Convolutions are "same" for Me, Rh and Ly, and "valid" for Ha. *Ha uses dilation rate of 20 and 13 along time dimension on 3rd and 5th layers respectively.

**Rhythmic features** Given the tempo invariance on the periodicity dimension explained in Section 3.1, the first layer has a 20 bins kernel on this axis to capture all patterns within 2 octaves (for instance quarter, eight and sixteenth notes). All other layers have 3x3 kernel with max-pooling of size 2 on the periodicity axis.

**Lyrics features** We conducted various experiments to find the best kernel size to apply to the lyrics. It appeared that a rather short receptive field of 10 bins yield the best results. We kept it for all layers, with a mean-pooling of size 5.

Finally, a dense layer applied after the temporal attention block outputs a 512 bins embeddings, L2-normalized so that each track becomes a point on the surface of the unit hypersphere, bounding the distance between 2 points within the [0,2] interval.

### 3.4 Embedding concatenation

In this work, we investigated only a late embedding fusion scheme, which simply consists in concatenating each feature embedding, and to L2-normalize the concatenated result. It is straightforward to show that the distance between a pair of normalized concatenation of $n$ feature embeddings is the quadratic mean of the $n$ distances between each feature embedding pair. All feature combination scores in Section 4 have been obtained using this method.

The practical advantage of this simple concatenated embedding is twofold: it remains easy to store and to query in an large index, and the lookup can be done for some specific feature combinations only (zero masking the unwanted feature embeddings).

## 4. EXPERIMENT AND RESULTS

In this section, we present our experimental setup and results. For brevity, we will denote melodic, harmonic, rhythmic and lyrics features by their abbreviations Me, Ha, Rh, and Ly, respectively.

### 4.1 Experimental setup

We use the exact same protocol as in our previous work [8].

**Training** We trained our four models on the publicly available dataset $SHS_{5+}$ [1], which contains ~62k covers of ~7.5k works. We used the provided features for Me and Ha, and extracted Rh and Ly from the audio, as described in sections 3.1 and 3.2. We used a semi-hard triplet loss, using an Adam optimizer, an initial learning rate of $1e^{-4}$ and a batch size of 64. All other details are replicated from [8].

**Test** We tested our four models on $SHS_{4-}$ [1], containing ~50k covers of ~20k works. We also retrained models on $SHS_{5+/4-}$ and tested on Da-Tacos [2], containing 13k covers of 1k works and 2k confusing tracks [29]. As some samples overlap between Da-Tacos, $SHS_{4-}$ and Dali, we made sure that none of these samples were used for scoring the models, as done in [5].

For each feature, we used the corresponding trained model to compute each track embedding, and computed their pairwise distance matrix. For feature combinations, distance matrix is computed using the quadratic mean of each feature distance matrix, as described in Section 3.4.

### 4.2 Results

We summarize in Table 2 the performances obtained on $SHS_{4-}$ and Da-Tacos by our models, reporting the metrics classically used for VI: Mean Average Precison (MAP), mean number of correct answers in the first 10 (MT@10) and mean rank of first correct answer (MR1).

| Train set | $SHS_{5+}$ | | | Pruned $SHS_{5+/4-}$ | | |
|---|---|---|---|---|---|---|
| Test set | Pruned $SHS_{4-}$ | | | Pruned Da-Tacos | | |
| Input feature | MAP | MT@10 | MR1 | MAP | MT@10 | MR1 |
| Me | 0.427 | 0.822 | 1131 | 0.363 | 4.064 | 97 |
| Ha | 0.538 | 1.003 | 982 | **0.488** | **5.256** | **63** |
| Rh | 0.099 | 0.231 | 2921 | 0.055 | 0.689 | 244 |
| Ly | **0.672** | **1.190** | **968** | 0.393 | 4.596 | 199 |
| Me+Ha | 0.693 | 1.256 | 453 | **0.626** | **6.668** | **32** |
| Me+Ha+Ly | **0.800** | **1.396** | 291 | 0.602 | 6.480 | 33 |
| Me+Ha+Rh | 0.688 | 1.250 | 413 | 0.557 | 5.994 | 33 |
| Me+Ha+Rh+Ly | 0.785 | 1.378 | **286** | 0.560 | 6.054 | 33 |
| Me+Ha (O) | 0.879 | 1.521 | 97 | 0.837 | 8.709 | 4 |
| Me+Ha+Rh (O) | 0.939 | 1.607 | 21 | 0.905 | 9.303 | **1** |
| Me+Ha+Ly (O) | **0.963** | **1.637** | **14** | **0.918** | **9.398** | **1** |
| Me+Ha+Rh+Ly (O) | **0.978** | **1.658** | **4** | **0.951** | **9.657** | **1** |

**Table 2**: Performance metrics obtained on $SHS_{4-}$ and Da-Tacos for input features and their combinations. O=oracle

We first examine the performances of each single feature. Me and Ha results are in line with our previous work [8].

**Rhythmic features** The performances obtained using Rh are clearly lower, which suggests that our rhythm feature is not providing relevant VI information, or that the rhythm patterns themselves are not specific enough to discriminate between versions and non-versions. As a consequence, adding Rh degrades our Me+Ha baseline. This is not entirely surprising, as many non-versions might exhibit similar rhythmic patterns (we will however see in Section 4.3 and Section 5 that Rh can be relevant in some cases).

[2] https://github.com/MTG/da-tacos

**Lyrics features** In contrast, the use of lyrics clearly outperforms the other features on $SHS_{4-}$. This confirms that versions often share the same lyrics, and that even a very inaccurate ALR system can be beneficial to VI. The combination Me+Ha+Ly improves by more than 10% the Me+Ha performances, which will be explained in Section 5.1.

On Da-Tacos, we observe much less clear-cut results. As already noticed by Vaglio et al. [30], Da-Tacos has about 20% of instrumental songs. A closer look to our results shows that these songs typically produce false positives. Following Vaglio et al., we pruned the instrumental songs from Da-Tacos, and recomputed the results for Ly, obtaining MAP=0.674, MT@10=5.931 and MR1=59, which is consistent with the values obtained for $SHS_{4-}$.

### 4.3 Oracle results

We also present in Table 2 the results obtained by an oracle. This oracle only considers the best performing feature to compute each pairwise distance: for each pair of tracks, it only uses the feature embeddings yielding the lowest (resp. highest) distance for versions (resp. non-versions).

Interestingly, it appears that the performances of an oracle using three or four features approaches the theoretical optimal values on our two datasets. This is particularly clear with the Me+Ha+Ly and Me+Ha+Rh+Ly combinations: the MAP tends to 1.0, i.e. most versions have been ranked in the first answers for each query. The MR1 also tends to 1, i.e. first answer is correct for most queries. The MT@10 also tends to the theoretical optimal values (in $SHS_{4-}$, each track has 2, 3 or 4 versions, and its optimal MT@10=1.695, while in Da-Tacos, each track has 0 or 12 versions, and its optimal MT@10=10).

The Figure 3 displays the contribution of each feature to the oracle score, i.e. which feature is the most relevant to determine if two tracks are versions ("Positive pairs") or non-versions ("Negative pairs").

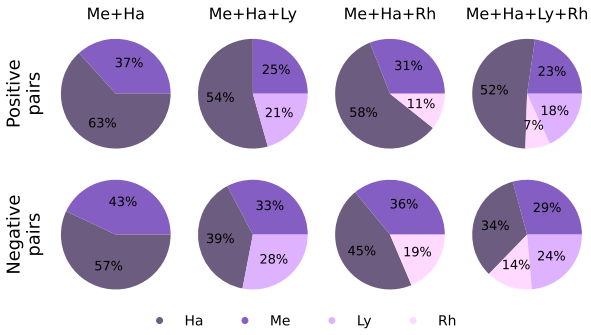



**Figure 3**: Most relevant feature proportions to identify positive and negative pairs on $SHS_{4-}$.

It appears that Ha is usually the most efficient feature to discriminate between versions or non-versions. However, both Me and Ly contribute importantly to identification (e.g. resp. 25% and 21% of the positive pairs, 33% and 28% of the negative pairs in the Me+Ha+Ly combination). Finally, and even though Rh alone yields poor results, its contribution is not negligible when combined with other features (e.g. 14% of the negative pairs in the Me+Ha+Rh+Ly combination).

### 4.4 Comparison with state-of-the-art

Performances obtained by recent VI systems are summarized in Table 3 (second column indicates the embedding size used by each system).

Our system using Me+Ha+Ly improves the state-of-the-art on $SHS_{4-}$ and on Da-Tacos (without instrumentals).

| Test set | | $SHS_{4-}$ | | | Da-Tacos | | |
|---|---|---|---|---|---|---|---|
| Model | Emb. | MAP | MT@10 | MR1 | MAP | MT@10 | MR1 |
| Doras et al. [8] | 512 | 0.660 | 1.080 | 657 | 0.635 | 6.744 | 30 |
| Vaglio et al. [30] | n/a | n/a | n/a | n/a | 0.804* | n/a | n/a |
| Du et al. [5] | 1536 | n/a | n/a | n/a | **0.791** | n/a | **19.2** |
| Me+Ha+Ly (ours) | 1536 | **0.800** | **1.396** | **291** | **0.818*** | **7.205*** | **16*** |
| | | | | | 0.602 | 6.480 | 33 |

**Table 3**: Sota comparison on $SHS_{4-}$ and Da-Tacos. *results obtained on Da-Tacos-Vocals (w/o instrumental tracks).

## 5. QUALITATIVE ANALYSIS

In this section, we illustrate qualitatively the previous quantitative results. We encourage readers to listen to the audio samples available on the paper companion website [1] as part of reading the paper.

### 5.1 Ly vs. Me+Ha examples

We intend here to illustrate how Ly improves the Me+Ha system. Figure 4 plots the distance obtained for Me+Ha and Ly between randomly sampled positive (green) and negative (red) pairs.

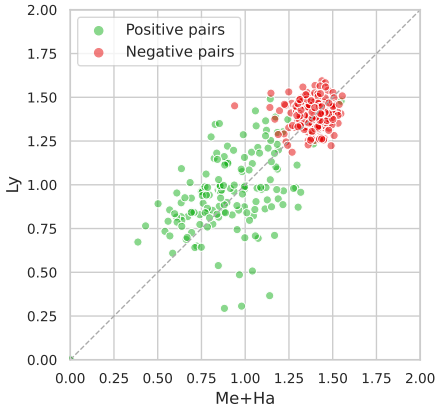
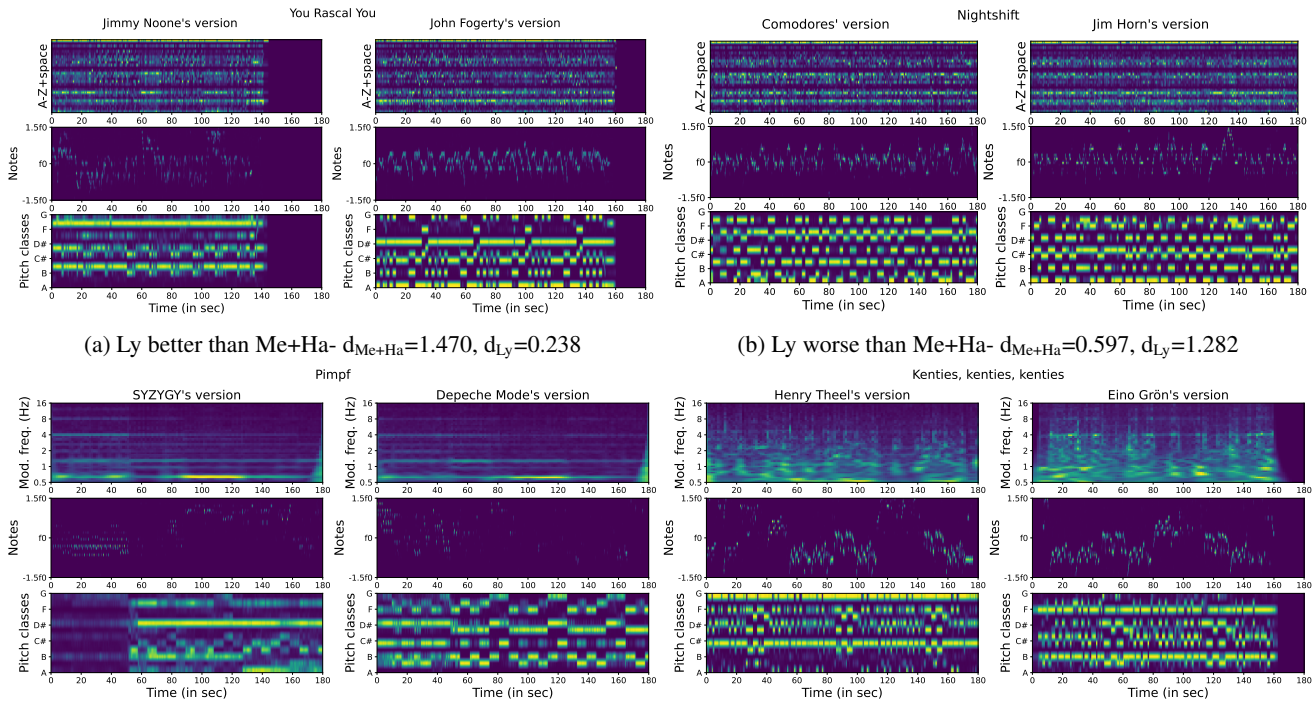



**Figure 4**: Pairwise distances for Me+Ha vs. Ly (500 pairs from $SHS_{4-}$).

This plot confirms that Me+Ha and Ly are complementary, as already seen in Section 4.3. The dots situated away from the diagonal correspond to track pairs scored correctly by Me+Ha and incorrectly by Ly (or vice-versa). As certain features yield better results for certain songs, their combination will statistically improve the results, as we will now illustrate with some contrasted examples.

**Ly > Me+Ha** There are many versions whose musical style, melody and harmony differ greatly from the original, and where only the lyrics can help to identify them. This is illustrated on Figure 5(a), which shows that the Jimmy Noone's and John Fogerty's versions of "You Rascal You" are very different musically while the lyrics exhibit enough similarity to be correctly identified.




(a) Ly better than Me+Ha- $d_{Me+Ha}$=1.470, $d_{Ly}$=0.238

(b) Ly worse than Me+Ha- $d_{Me+Ha}$=0.597, $d_{Ly}$=1.282

(c) Rh better than Me+Ha - $d_{Me+Ha}$=1.084, $d_{Rh}$=0.459

(d) Rh worse than Me+Ha - $d_{Me+Ha}$=0.498, $d_{Rh}$=1.380

**Figure 5**: Examples for rhythm and lyrics : Rh > Me+Ha 5(c), Rh < Me+Ha 5(d), Ly > Me+Ha 5(a) and Ly < Me+Ha 5(b)

It also appears that our approximated ALR system is efficient for different languages. For instance, the versions of Asta Kask and of The Hep Stars of the song "I natt jag drömde", are very different in melody and harmony ($d_{Me+Ha}$=1.448) while the lyrics remain similar ($d_{Ly}$=0.342), despite the fact that the lyrics are in Swedish.

**Ly < Me+Ha** As already mentioned, using Ly will yield wrong results in presence of instrumental version (i.e. no lyrics). In the example shown in Figure 5(b), the version of "Nightshift" by the Commodores has lyrics while the one of Jim Horn's does not. We noticed that the system sometimes considers lead instruments as voices. However, this Ly false negative is correctly caught by the Me+Ha.

### 5.2 Rh vs. Me+Ha examples

Although less obvious than for Ly, combining Rh and Me+Ha also appears to be complementary in some cases.

**Rh > Me+Ha** Even though Rh yields poor performances in general, there are cases where it is the only feature available to identify versions. This is illustrated on Figure 5(c), which shows the Rh, Me and Ha features for two versions of "Pimpf". In this song, the melody is almost non-existent, and the harmony is very different between both versions. Only a few bass notes in the middle are salient enough to identify the song, and this short bassline appears similarly on the two FP features.

Rh might also be a good discriminating feature in other cases, e.g. for live concert versions. One version of "Mama's Little Baby" is recorded in studio while the other is a concert filmed from the audience. The Me+Ha distance between these versions is high ($d_{Me+Ha}$=1.329) because of the bad live recording quality. But, the drums are distinguishable enough to find similarity ($d_{Rh}$=0.714).

**Rh < Me+Ha** But Rh often yields wrong results. This is illustrated on Figure 5(d), which shows the features of two versions of "Kenties kenties kenties". Although melody and harmony are similar, the rhythm is very different, and the use of Rh produces a false negative.

## 6. CONCLUSION

It was shown previously that VI systems combining melody and harmony yields promising performances. In this paper, we proposed to consider also rhythmic and lyrics features to improve these results further. We showed that an existing rhythmic feature commonly used for genre classification is only helpful in a few cases, such as live version identification. But we also showed that an approximate lyrics representation can improve the performances of existing melody and harmony-based systems. We explained these results by the fact that detecting correctly only a few character sequences appears to be enough to distinguish versions and non-versions. We showed that our system combining these features establishes new state-of-the-art on two public datasets. More importantly, we indicated that these feature combinations provide enough information to approach the theoretical optimal performances obtainable on these datasets.

In our future work, we will investigate a more elaborated fusion scheme in order to train our model to behave as an oracle: our objective is to teach the system how to choose between available features to pick only the most relevant one for each pair of tracks. This might answer the question of whether the concept of musical version can be reduced to its melodic, harmonic, rhythmic, and lyrics dimensions.

## 7. ACKNOWLEGMENTS

Authors would like to thank the anonymous reviewers for their helpful comments, and for suggesting the Figure 3.

## 8. REFERENCES

[1] F. Yesiler, G. Doras, R. M. Bittner, C. J. Tralie, and J. Serrà, "Audio-based musical version identification: Elements and challenges," *IEEE Signal Processing Magazine*, vol. 38, no. 6, 2021.

[2] Z. Yu, X. Xu, X. Chen, and D. Yang, "Learning a representation for cover song identification using convolutional neural network," *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[3] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2019.

[4] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[5] X. Du, K. Chen, Z. Wang, B. Zhu, and Z. Ma, "Bytecover2: Towards dimensionality reduction of latent embedding for efficient cover song identification," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 616–620.

[6] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[7] Z. Yu, X. Xu, X. Chen, and D. Yang, "Temporal pyramid pooling convolutional neural network for cover song identification," *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[8] G. Doras, F. Yesiler, J. Serrà, E. Gomez, and G. Peeters, "Combining musical features for cover detection," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2020.

[9] A. Vaglio, R. Hennequin, M. Moussallam, and G. Richard, "The words remain the same - cover detection with lyrics transcription," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2021.

[10] C. Douwes, P. Esling, and J.-P. Briot, "Energy consumption of deep generative audio models," *arXiv preprint arXiv:2107.02621*, 2021.

[11] M. Slaney, K. Weinberger, and W. White, "Learning a metric for music similarity," in *International Symposium on Music Information Retrieval (ISMIR)*, vol. 148, 2008.

[12] B. McFee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 8, pp. 2207–2218, 2012.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, 2015.

[14] J. Foote, M. Cooper, and U. Nam, "Audio retrieval by rhythmic similarity." in *ISMIR*. Citeseer, 2002.

[15] S. Dixon, F. Gouyon, G. Widmer *et al.*, "Towards characterisation of music via rhythmic patterns." in *ISMIR*, 2004.

[16] E. Pampalk, "Computational models of music similarity and their application in music information retrieval," Ph.D. dissertation, 2006.

[17] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, "On rhythm and general music similarity." in *ISMIR*, 2009, pp. 525–530.

[18] H. Foroughmand and G. Peeters, "Deep-rhythm for tempo estimation and rhythm pattern recognition," in *International Society for Music Information Retrieval (ISMIR)*, 2019.

[19] E. Trentin and M. Gori, "A survey of hybrid ann/hmm models for automatic speech recognition," *Neurocomputing*, vol. 37, no. 1-4, pp. 91–126, 2001.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[21] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[22] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: a large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm." in *19th International Society for Music Information Retrieval Conference*, ISMIR, Ed., 2018.

[23] C. Gupta, E. Yılmaz, and H. Li, "Acoustic modeling for automatic lyrics-to-audio alignment," *arXiv preprint arXiv:1906.10369*, 2019.

[24] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," *arXiv preprint arXiv:1902.06797*, 2019.

[25] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics alignment and transcription in polyphonic music: Does background music help?" in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 496–500.

[26] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 627–636.

[27] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully convolutional speech recognition," *arXiv preprint arXiv:1812.06864*, 2018.

[28] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series." in *CCIA*, 2018.

[29] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. Gómez, and X. Serra, "Da-tacos: A dataset for cover song identification and understanding," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2019.

[30] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. d'Alché Buc, "Audio-based detection of explicit content in music," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 526–530.

# A DIFFUSION-INSPIRED TRAINING STRATEGY FOR SINGING VOICE EXTRACTION IN THE WAVEFORM DOMAIN

**Genís Plaja-Roglans** **Marius Miron** **Xavier Serra**
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{genis.plaja, marius.miron, xavier.serra}@upf.edu

## ABSTRACT

Notable progress in music source separation has been achieved using multi-branch networks that operate on both temporal and spectral domains. However, such networks tend to be complex and heavy-weighted. In this work, we tackle the task of singing voice extraction from polyphonic music signals in an end-to-end manner using an approach inspired by the training and sampling process of denoising diffusion models. We perform unconditional signal modelling to gradually convert an input mixture signal to the corresponding singing voice or accompaniment. We use fewer parameters than the state-of-the-art models while operating on the waveform domain, bypassing the phase estimation problem. More concisely, we train a non-causal WaveNet using a diffusion-inspired strategy while improving the said network for singing voice extraction and obtaining performance comparable to the end-to-end state-of-the-art on MUSDB18. We further report results on a non-MUSDB-overlapping version of MedleyDB and the multi-track audio of Saraga Carnatic showing good generalization, and run perceptual tests of our approach. Code, models, and audio examples are made available. [1]

## 1. INTRODUCTION

Singing voice extraction, which involves separating the vocal source from music recording mixtures, has received a lot of attention from the Audio Signal Processing (ASP) and Music Information Retrieval (MIR) communities in the recent years. The problem can be modelled in the waveform domain [1–4], the frequency domain [5–7], or a combination of both [8–10]. In general, spectrogram-based approaches have been more popular despite having to deal with the problem of the complex phase, usually leading to artifacts or unnaturalness of the separated sources. Within the Music Demixing Challenge (MDX) framed in ISMIR 2021 [11], diverse submissions achieved state-of-the-art source separation performance, in the majority of cases being multi-branch networks combining features from both time and frequency domains [8–10], proposing therefore solutions to account for the problem with the phase. Nonetheless, these models tend to be heavy-weighted and include engineered strategies to improve the predicted outputs.

While the problem of source separation has been shown to be challenging on the waveform domain, promising results have been reported [2, 3, 12], opening the door for the development of models that bypass the problem with the complex phase. However, as the performance improves, the model size and complexity accordingly grow.

In this work we propose a training and sampling strategy inspired on the recently emerged denoising diffusion models [13] to perform end-to-end singing voice extraction. Denoising diffusion models are a novel class of generative models theoretically grounded in the non-equilibrium statistical physics that can gradually convert one distribution into another using a Markov chain [14], while learning to perform the reverse process. More concisely, numerous works use diffusion models to convert a signal from a particular data distribution to a simple one (e.g. isotropic Gaussian noise) by gradually adding samples of the said simple distribution. Subsequently, the model is trained to reverse the perturbation process and generate data samples of the original distribution using the easily tractable noise as input [15–17]. Diffusion models have recently emerged as a versatile and high-performance method for data generation, outperforming classical generative approaches for the task of image generation [13]. In the fields of ASP and MIR, diffusion models have also shown promising performance for speech synthesis [16, 18], speech restoration and enhancement [13, 15, 19–21], audio super-resolution [22], singing voice synthesis [23], and symbolic music generation [24]. Despite that, the literature does not include many attempts to use diffusion models for source separation, being [25], to the best of our knowledge, the only attempt.

Despite the use of deterministic signals as diffusion perturbation in place of Gaussian noise has shown promising results in reverting different arbitrary types of image noise and performing image morphing [26], to the best of our knowledge, no exploration of this idea in the audio domain has been reported to date. In this work, we build on top of DiffWave, a versatile diffusion model for speech synthesis [16] that is based on a non-causal WaveNet architecture that has been previously used for music source separation [12]. We introduce an end-to-end approach to model

[1] github.com/genisplaja/diffusion-vocal-sep

the task of singing voice extraction as a diffusion-alike process, by converting between two audio data distributions that share similar content, in this case the singing voice and the corresponding mixture. We propose to use a deterministic diffusion perturbation, a music mixture, to gradually transform its corresponding isolated singing voice into the mixture while learning to conduct the reverse process at inference. Given the formulation of the diffusion process, we train a model that learns to estimate the perturbation at different ratios. Subsequently, we leverage from the parametrization of the reverse process of diffusion models to chain these estimations and sample, given an input mixture, improved vocal source separation in terms of artifacts and interferences compared to the vanilla trained network.

## 2. METHOD

### 2.1 Diffusion process

We assume that the waveform-domain signal corresponding to the mixture $m$ is the sum of the singing voice $v$ and the accompaniment $a$, such as: $m = v + a$. Our goal is to estimate $v$ given $m$. In the following sections we formalize our method by relating it with the diffusion theory in the literature. In Figure 1 we display the two main steps of our method: the diffusion and the reverse process.

**Diffusion.** The diffusion process assumes perturbing the training data with different scales of noise iteratively following a Markov chain [13]:

$$q(x_1, ..., x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

The input signal $x_0$ is gradually perturbed by incrementally adding a particular signal in small $T$ diffusion steps. This process results into a latent variable $x_T$ of same distribution of the perturbation. The standard diffusion schema, introduced in [13] and subsequently used in most of the diffusion research, perturbs $x_0$ with random Gaussian noise, therefore $x_T$ is an isotropic Gaussian noise distribution. In our case, the input signal $x_0$ is initialized with the isolated singing voice $v$, and perturbed by incrementally adding the mixture $m$. This results in $x_T$ being a mixture-alike signal, containing both voice and accompaniment. Therefore, $q(x_t|x_{t-1})$ in Eq. 1 is an operation to add a small amount of perturbation $m$ to the given signal $x_{t-1}$, moving to the next diffusion step $x_t$. We use $m$ as perturbation to account for the formal diffusion design in [13], in which the latent variable $x_T$ is expected to belong to the same data distribution as the perturbing noise. The level of perturbation at each diffusion step is controlled by $\beta_t$, which is a small positive coefficient within a fixed noise schedule denoted $\beta$. That said, using the parametrization proposed in [13], we can compute any given diffusion step using:

$$q(x_t|x_0) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}m \tag{2}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Note also that the most common inference input of a singing voice extraction model – which in our case is $x_T$ – is a mixture. Given Eq. 2, the perturbation is a mixture $m$ to ensure $x_T \approx m$, otherwise the said condition may not be given.

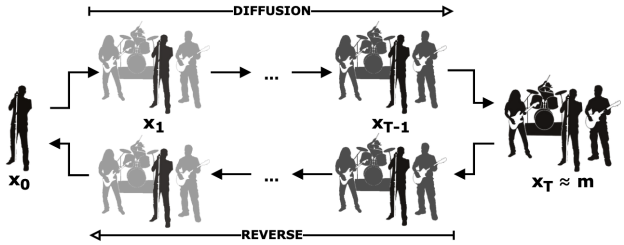



**Figure 1**. Overview of our diffusion-inspired training approach for the case of singing voice extraction.

Modelling musical signals using a mixture of Gaussian functions has been previously explored in [27]. Note also that architectures similar to DiffWave have been already used to model mixture, accompaniment, and vocal signals [12, 28] (for further detail see Section 2.2).

**Reverse process.** The reverse process aims at iteratively reverting the perturbation added by the diffusion:

$$p_\theta(x_0, ..., x_{T-1}|m) = \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{3}$$

We propose to parameterize the variable $p_\theta(x_{t-1}|x_t)$ as $\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon(x_t, t)\right)\frac{1}{\sqrt{\alpha_t}}$, being $\epsilon(x_t, t)$ the perturbation estimated by the model at a given diffusion step $t$. This parametrization is leveraged from the diffusion sampling process in [13]. However, we remove the deviation parameter from the original parametrization, which in our deterministic noise case yields worse predictions. This process can be seen as an iterative refinement of the latent variable $x_T$ to convert it to $x_0$, in our case to iteratively transform an input mixture to its corresponding singing voice source.

**Training.** The training objective of the original diffusion process is to maximize the log likelihood of: $p_\theta(x_0) = \int p_\theta(x_0, ..., x_{T-1}|x_T)p_{latent}(x_T)dx_{1:T}$ [13], considering stochastic noise as perturbation. Since in this context it is not possible to calculate the said integral, in the literature this problem is approached by maximizing its variational lower bound (ELBO) [13]. The reader is referred to [16], for a detailed development of this maximization. In our case, relying on the ELBO is not required given the deterministic perturbation. Therefore, by using pairs of $(x_t, x_0)$ [13], and referring to Eq. 2, we are able to effectively optimize the model with parameters $\theta$ using the following objective [13]:

$$L(\theta) = \left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}m, t))\right\|^2 \tag{4}$$

where $\epsilon$ is the target or true perturbation, whereas $\epsilon_\theta$ is the perturbation the model estimates based on $\bar{\alpha}, t$, the mixture $m$ and the input $x_0$. In the case of extracting the singing voice, we use the accompaniment $a$ instead of mixture $m$ as the target corresponding to $\epsilon$. Therefore, we do not include the voice into the target of the training process, thus we alleviate the loss of vocal quality that may occur after several reverse steps.