2021 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, OCT. 25–28, 2021, GOLD COAST, AUSTRALIA

SOURCE SEPARATION METHODS FOR COMPUTER-ASSISTED ORCHESTRATION

Anonymous

Anonymous

ABSTRACT

In this paper, we will study the possibility of adding source separation as a pre-processing step to the computer-assisted orchestration process. We first discuss the motivation of this addition and its potential to increase the quality of orchestrations of multi-layered sounds. Second, we select several state-of-the-art models for both music source separation (separation of instruments) and universal sound separation (separation of arbitrary sounds of different types), and compare their effectiveness for the task of orchestration. We assess which methods best suit the needs of orchestration by applying them on hand picked target sounds, orchestrating the separated outputs, and finally comparing them to the orchestration of the same target without separation. Our experiments show that the quality of orchestrations improves, both qualitatively and quantitatively, indicating that our approach is promising. Finally, we compare unsupervised methods to supervised methods for separation, and comment on the effect of training data selection on performance of supervised methods.

Index Terms— computer-assisted orchestration, sound source separation, Orchidea

1. INTRODUCTION

Computer-assisted composition is a field that focuses on the creation of computational tools to aid in the musical composition process [1, 2]. Within this field, target-based computer-assisted musical orchestration is an example of how musical creativity can be supported by tools such as machine learning.

Orchestration [3] is hard to define, however one could say that orchestration is the set of musical writing techniques that focus on timbre, or *tone color* of the sound. A simplistic definition of musical orchestration would be to say that it consists of the distribution of the different musical voices of a piece to the instruments of the orchestra. Orchestration has been traditionally taught in an intuitive and non-formalized way, and computer-assisted orchestration seeks to help composers by accelerating certain creative processes.

Among the different approaches and techniques of computer-assisted orchestration, one of them is *target-based orchestration* [4]. Target-based computer-assisted orchestration helps composers explore new timbral possibilities by combining instrumental samples in such a way as to mimic the timbre of a given target sound. This target corresponds to an audio extract, of the composer's choice, which can be musical or not. Formally, it is the process of creating an orchestral score that best matches an arbitrary target sound given a similarity metric and constraints.

Static orchestration is a type of orchestration that considers the target as a single vector of timbral features and creates only a single onset of note(s) as the solution, no matter if the target contains multiple onsets of sounds. In our experiments, we only perform static orchestrations.

Orchidea is a framework and set of tools that is currently considered the state-of-the-art for computer-assisted orchestration. It embeds the target in a feature space and uses a jointly-optimized heuristic and constraint solver to find a combination of samples that best match the target [5, 6]. All of our orchestrations are done using Orchidea's command line tools for batch processing, which is available for download at `http://www.orch-idea.org`

Sound source separation is the process by which a single audio file is separated into multiple sound sources. A perfect separation is able to dissect a sound exactly into its constituent parts, in which each part is an independent sound event. Music source separation is a specific application of source separation in which the input audio is expected to be music that is comprised of a subset of instruments or sounds. For example, one music source separation method could attempt to split a song into three parts: voice, bass, and drums. Another aimed at orchestral music may try to separate the input into families of instruments: woodwinds, brass, strings, and percussion. In contrast, universal sound separation does not operate under the assumption that the input is of a musical nature, and can be applied to arbitrary sounds. In this case, the number of sources expected is specified, and the method will attempt to divide the input into the given number of sources.

Using source separation techniques as a pre-processing step has been shown to improve results in several music processing tasks. Source separation can be used to remove part of the spectrum in order to enhance the clarity of the features of interest for the task at hand. For instance, in the task of automatic chord estimation from audio, some authors have chosen to remove part of the spectrum related to percussive sounds to improve chord accuracy [7]. Similarly some authors have investigated how beat tracking can be improved by using source separation as a pre-processing step for difficult songs with highly predominant expressive vocals [8].

Anonymous.

978-1-7281-6338-3/21/$31.00

Source separation can also be applied to separate different components of the signal and work on them separately to reduce the complexity of the task. For instance, in the context of tempo detection [9], the authors decompose the signal into sources to reduce rhythmic complexity, under the idea that some layers may be more rhythmically regular than the overall mix. Another example can be found in [10] where harmonic/percussive and solo/accompaniment source separation techniques are investigated as a pre-processing step to improve predominant instrument recognition in jazz music.

We think source separation could be an important addition to improve orchestration as it allows for the orchestration of more complex sounds in a way that makes sense in an orchestral context. Orchestral music often uses multiple layers, so by separating the layers of a target and individually orchestrating them, the resulting orchestration may improve.

For example, consider a target that has a continuous drone sound throughout and a melody playing above this drone. Without source separation, Orchidea would detect each note of the melody as a new onset and cut the target in time at each new note. However, this segmentation in time would also affect the drone, cutting it in time and forcing it to have a new onset each time the melody changes notes. The resulting orchestration would have multiple onsets for the drone, even though only one onset is needed at the beginning of the drone. When source separation is applied to this target, the drone and melody would be orchestrated separately. Therefore, the drone could be orchestrated by a single onset that lasts for the duration of the drone. At the same time, the melody could have as many new onsets as needed without affecting the drone. The orchestration would be greatly improved by removing the unnecessary onsets in the orchestration of the drone.

The codebase for this paper can be found at: `anonymous`, and a selection of targets and orchestrations are available for listening at: `anonymous`.

2. METHODOLOGY

We compare the effectiveness of different source separation methods for the task of orchestration by applying the separation methods to various targets, orchestrating the separated output of the method, and finally comparing this to the orchestration of the target if no separation was performed (see Fig. 1). Following the previous example, consider a target sound that is a combination of a low droning sound and high-pitched whistle. With separation, these two sounds would be disentangled into two separate sub-targets and each would be individually orchestrated. The separated orchestrations would then be combined to create the solution. This would then be compared to the orchestration when no separation had been performed.

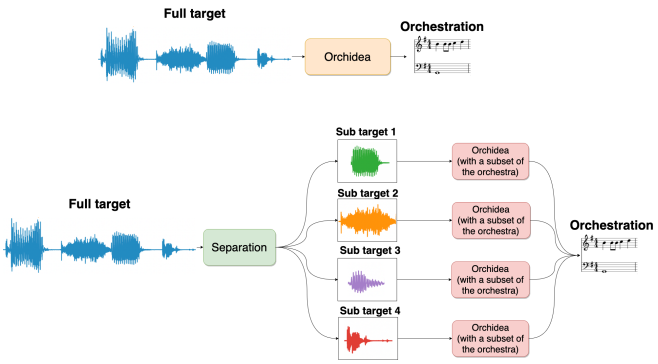

Fig. 1. Diagram comparing the process of orchestration with and without separation. At the top, the full target orchestration is created simply by orchestrating the target. At the bottom, the separation process is performed, and the resulting sub-targets are individually orchestrated with subsets of the orchestra before being recombined to obtain the full orchestration.

3. EXPERIMENTS

We test the effectiveness of five difference source separation methods, both supervised and unsupervised, on 300 custom made targets that are combinations of sounds from different databases. During testing, a target is input to a source separation method, which outputs four sub-targets. Then each sub-target is independently orchestrated with a randomly assigned subset of the full orchestra. These orchestrations are then combined to play simultaneously, creating a final orchestrated solution. Then the distance between the target and solution is calculated, giving us a metric to compare the various separation methods. See Fig. 2 for a diagram of this process.

For each target, the same random split of the orchestra is used to orchestrate the output of each separation method. We randomly create these subsets of the orchestra in order to avoid any biases that could come from a hand-picking the instruments used to orchestrate each sub-target.

3.1. Data

We created our own targets as combinations of four source sounds. The sources come from the NIGENS [11] and BBC [12] databases, and freesound.org [13]. We selected a total of 90 samples from these databases, choosing sounds that fit the following criteria:

1. Static sounds that do not change harmonically over time
2. Sounds in which there is at least some pitched content and not only noise

The sounds chosen include alarms, bells, engine noises, soundscapes, synthesizer chords, and sound effects. Each target that we used for testing was a combination of four

randomly chosen source sounds from the group of 90 sounds. When a target is created, the longest of the four sources is chosen to start playing at the beginning of the target. The other three sources are then randomly assigned different times to begin playing such that they will start between the beginning of the target and half-way through the longest source sound. This is done to ensure that the sources overlap and to minimize the amount of time in which there is only a single source sounding.

3.2. Separation methods

We consider source separation models trained to tackle two distinct separation problems. The first is music source separation (Open-Unmix, Demucs, ConvTasnet). For these architectures, we used models that were pre-trained on the MUSDB18 dataset [14]. Thus, these models output 4 stereo tracks that correspond to broad instrumental categories defined in the SiSEC 2018 Mus evaluation campaign: *vocals, drums, bass* and *other* [15]. It is important here to make the distinction that we did not train these models ourselves. Therefore, we cannot guarantee that the data the models were trained on is optimal for solving our problem. However, one of the goals of this paper is to explore whether pre-trained music source separation models can separate non-musical sounds in a way that improves orchestration. The second type of model we tested are ones that perform universal sound separation (TDCN++, NMF), which [16] defines as *separating mixtures of arbitrary sounds of different types*. These methods more closely match our problem of separating non-musical target sounds.

3.2.1. Music source separation

Open-Unmix [17] is an open source implementation of a deep music source separation model, based on [18]. It is actually composed of multiple models that are trained for each instrumental target. Each of these models is trained on the specific target using the same architecture, based on a three layer bidirectional LSTM network. Open-Unmix operates on the Short Time Fourier Transform (STFT) of the input mixture, which is first standardized using the global mean and standard deviation for every frequency bin across all frames. It predicts the target by multiplying the output of the LSTM network with the magnitude spectrogram of the input, essentially applying a mask. Open-Unmix also uses Wiener filtering as a last processing step.

The **Demucs** [19] architecture is a deep learning model for musical source separation. It takes the stereo mixture as input, and outputs 4 stereo waveforms corresponding to the four categories: vocals, bass, drums and other. Demucs is based on a convolutional encoder, a bidirectional LSTM, and a convolutional decoder. The encoder and the decoder are linked with skip U-Net connections. Both the encoder and the decoder are made of 6 blocks, each made of a combination of a convolutional layer with kernel size 8, followed by ReLU activation. Finally there is another convolution with kernel size 1, followed by gated linear units (GLU). The U-network structure allows the decoder to directly access the input signal and to transfer its phase, improving results.

We also used the Conv-Tasnet[20] implementation of [19], who adapted the architecture, originally designed for speech separation at 8 kHz, for the task of music source separation. Conv-Tasnet is a model based on a convolutional encoder-decoder, with a time-dilated convolutional network separation module, estimating masks based on the encoder output. This is why we refer to it as **TDCN**, in accordance with [16]. It uses the audio waveform as input and performs a series of 1-D convolutions at different resolutions with residual connections in order to generate a pool of masks that are then applied on the embedding of the initial waveform to output the separated targets. A final 1-D convolutional layer is used as a decoder to recover the waveforms of each sub target. This type of architecture replaces RNNs in modeling sequential data, as series of dilated convolutions are a good way to capture temporal patterns.

3.2.2. Universal sound separation

Non-negative matrix factorization [21, 22] is an unsupervised learning method, which takes an input non-negative matrix, in this case a spectrogram S, in $\mathbb{R}^{N\times M}$ and decomposes it into a basis matrix A in $\mathbb{R}^{N\times J}$ and a component matrix X in $\mathbb{R}^{J\times M}$ such that $S \approx A \times X$. We use the coordinate descent algorithm from the scikit-learn Python package based on [21] with J set to 4, which selects for 4 features in the dataset. We then output 4 mono waveforms containing each isolated feature F_i, where $F_i = A_i \times X_i$ for $i = 1, 2, 3,$ or 4.

TDCN++, a version of Conv-Tasnet, was improved by [16] in order to be used for general sound separation, and not only for speech. The main architecture remains the same as in [20], and the changes only affect the masking network. More specifically, feature-wise normalization replaces global normalization, residual connection ranges are extended to improve the flow of information, and learnable scale parameters are introduced to weight the outputs of each residual layer. Those modifications help the model to better extract features without drastically increasing its size. TDCN++ presents the interesting property of having almost the same architecture as TDCN but trained with generic sounds. Thus, it should naturally be better suited for our task, and the comparison between those two models in particular can give good insights about the impact of the data used for training.

3.3. Testing

The procedure we use for testing is as follows:

1. The target is created as a combination of four randomly chosen sources, which are offset to begin playing at different times.

2. The target, *without any separation performed*, is orchestrated using the entire orchestra. This creates what we call the *full target orchestration*.

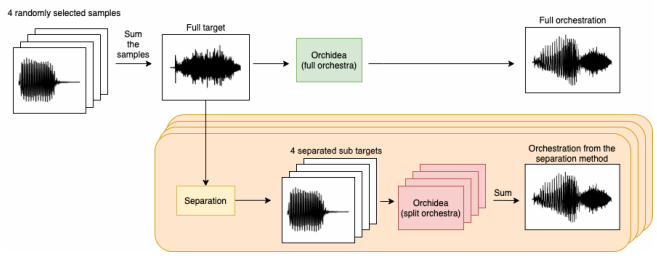

Fig. 2. Diagram showing the testing process. Note that in both orchestrations that use a split orchestra, they use the same division of the orchestra to orchestrate their four sub-targets.

3. Each separation method is performed on the target, each splitting the target into four sub-targets.

4. The full orchestra is randomly split into four equal sized subsets, each containing 7 to 8 instruments.

5. For each method, the four sub-targets are separately orchestrated, each using a different subset of the orchestra, and then each orchestration is combined to play simultaneously. This creates the *separated orchestrations*.

6. The distances between target and orchestration are calculated for the full target orchestration and the separated orchestration.

The orchestrations performed are static orchestrations, meaning a single onset of notes is created for each target, no matter if the target itself has multiple onsets. The Orchidea-SOL database of orchestral samples is used with Orchidea to create the orchestrations [23]. This database contains recordings of extended playing techniques, which better fits the often noisy or inharmonic nature of our targets. The full orchestra, of which non-overlapping subsets were selected to orchestrate the subtargets, contains the following instruments: 8 violins, 4 violas, 3 cellos, 1 bass, 2 oboes, 2 flutes, 2 clarinets in B♭, 2 bassoons, 2 trumpets, 2 trombones, and 2 French horns.

As described in Section 3.1, each target is comprised of four sources that are offset in time to begin playing at different times. Figure 3 illustrates the effect this offset has on the full target and separated orchestrations. In the full target orchestration, all onsets occur at the beginning of the orchestration. Since there is no knowledge of different sources, Orchidea cannot place the different orchestrations at the appropriate offsets.

For the separated orchestrations, the onsets of the sub-target orchestrations can be placed at the correct offset if the separation method has correctly separated out the sources. For this reason, despite only performing static orchestrations, the resulting separated orchestrations are offset in time compared to the full target orchestration.

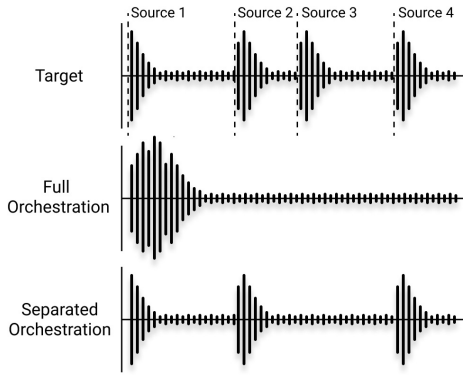

Fig. 3. Example waveforms showing onsets of the four sources in the target, and possible onsets in the full and separated orchestrations. In this example, the separation method could not distinguish between sources 2 and 3 and identified them as a single sub-target. Therefore, the separated orchestration has only 3 onsets, one of them being an orchestration that combines sources 2 and 3. Note that this is a simplified representation, in the actual targets sources overlap in time and are not as discrete as this diagram shows.

3.4. Evaluation

We compare the effectiveness of different separation methods by comparing how well they work for orchestration. The output of a method is orchestrated, and these orchestrations are compared. A quantitative evaluation is performed through the use of a distance metric that measures the spectral distance between target and solution.

The distance metric cuts the target and solution into successive frames that are 4,000 samples in length, which is approximately 90 milliseconds at a sampling rate of 44.1 kHz. The spectral distance, as defined in Eqn. 1, is calculated between corresponding frames of the target and solution. The spectral distance metric is proposed in [6] as part of the cost function used in Orchidea during the optimization, and used in [24] to compare accuracies of orchestrations. The equation takes in the magnitude spectrums of the FFTs of the target x and of the solution $\tilde{x}$. Then for each bin k of these spectrums, it calculates the absolute difference between the values. The differing values of λ_1 and λ_2 allow the metric to penalize the solution in different ways. For our purposes, we used $\lambda_1 = 0.5$ and $\lambda_2 = 10$, which penalizes a solution that overshoots the harmonic energy of the target.

$$d(x, \tilde{x}) = \lambda_1 \sum_k \delta_{k1}(x_k - \tilde{x}_k) + \lambda_2 \sum_k \delta_{k2}|x_k - \tilde{x}_k| \quad (1)$$

where $\delta_{k1} = 1$ if $x_k \geq \tilde{x}_k, 0$ otherwise; and $\delta_{k2} = 1$ if $x_k < \tilde{x}_k, 0$ otherwise.

	Average distance
Demucs	25.11
Open-Unmix	24.48
TDCN	26.44
NMF	**22.44**
TDCN++	24.38
Full target orchestration	24.96

Table 1. Average across 300 targets showing the distance between target and orchestration for various methods. For Demucs, Open-Unmix, TDCN, NMF, and TDCN++, the value shown is the average distance between the target sound and the "separated orchestration" created from the output of the given separation method (see Sec. 3.3). Similarly, "Full target orchestration", as defined in Sec. 3.3, is the average values between target and orchestrated solution. Lower values are better.

3.5. Results

The average distances for full target and separated orchestrations are displayed in Table 1. These results are the average distance between target and orchestration for 300 different targets. We found that NMF performed the best out of all separation methods. TDCN++, Open-Unmix, and NMF performed better than the full target orchestrations, meaning that separating targets using these methods leads to a better orchestration than if no separation is applied. However, Demucs and TDCN performed worse.

4. DISCUSSION

In this paper, we addressed the problem of using source separation as a pre-processing step for computer-assisted orchestration. The results show that performing source separation followed by a static orchestration of each individual sub-target improved the spectral distance between the original target and its orchestrated version in three out of five of the methods tested. We believe this shows that source separation is a useful pre-processing stage for computer-assisted orchestration.

In order to fully understand the impact of source separation on the orchestration process, we also conducted a qualitative evaluation of the generated orchestrations by acoustic inspection. This evaluation showed that source separation significantly improves the quality or orchestrations from a musical standpoint. The next paragraphs will try to explain this in more detail.

A major disadvantage of the orchestrations generated without any source separation is that, when two or more sources overlap in time, the orchestration algorithm cannot distinguish between them. As such, the orchestration appears to be more rigid: the whole orchestra suddenly changes configuration when a new source dominates, without any blending with the previous dominating source. This is against a well know orchestration principle called *dovetailing* [3, pp. 467-473]. Dovetailing is a well known practice aimed at creating a blend of the instruments used during orchestration, in order to transition between timbres. Technically, dovetailing is the swapping and overlapping of musical lines between different instruments: musical lines are scattered across multiple instruments, connected with overlapping pitches.

An interesting result that we discovered after acoustic inspection is that the orchestrations generated after source separation tend to favor dovetailing. Since only a portion of the orchestra is assigned to each source and since the sources are inherently asynchronous with each other, the final orchestration is more fluid and appears more interesting, from a musical standpoint.

Quantitatively, the one unsupervised method we tested, NMF, outperformed all the supervised methods. This suggests that methods like NMF can still be useful for specific problems when available data is limited. It is logical that unsupervised methods will perform well for problems in which there is limited data, as they require less data than supervised methods. It is important to notice that for the supervised methods, we used pre-trained versions of the neural models. All of these models, except for TDCN++, were trained on musical data. However, we tested them on a wide range of targets, many of which were not strictly musical. This could explain why Demucs, Open-Unmix, and TDCN performed worse than NMF, and why TDCN++, which was trained on arbitrary sounds, outperformed the other supervised methods. Therefore, we may infer from our results that models trained on generic sounds generate a better separation for orchestration when compared to models trained only on musical sounds. Since unsupervised methods perform well, it is reasonable to think that supervised methods trained on an appropriate dataset of arbitrary sounds could yield better overall performance for orchestration.

4.1. Future work

As we stated previously, we believe that one of the reasons the supervised methods did not perform as well is because of the data they were trained on. Therefore, a natural next step is to train these models with sounds that are more appropriate for computer-assisted orchestration.

As we stated in Section 1, we performed only static orchestration in our experiments. In contrast to static orchestration, dynamic orchestration considers a time series of features, which allow the orchestration to evolve over time and contain multiple onsets of notes. Orchidea is able to perform dynamic orchestration by cutting the target in time into multiple segments, and then orchestrating each segment. However we chose to use only static orchestrations in order to better observe the effects of source separation. The reasoning for this is that we wish to disentangle the two difficult problems of source separation and time segmentation. By only performing static orchestrations, we reduce the complexity of the problem by removing time segmentation and can better observe

the effects of source separation. Future work could include applying source separation to dynamic targets and performing dynamic orchestration.

Finally, one or multiple of the separation methods we tested should be implemented in Orchidea in order to improve the orchestrations Orchidea is capable of creating.

5. REFERENCES

[1] J.D. Fernandez and F. Vico, "Ai methods in algorithmic composition: A comprehensive survey," *Journal of Artificial Intelligence Research*, vol. 48, pp. 513582, Nov 2013.

[2] Christopher Ariza, "Navigating the landscape of computer aided algorithmic composition systems: a definition, seven descriptors, and a lexicon of systems and research," in *ICMC*, 2005.

[3] S. Adler, *The Study of Orchestration*, The Study of Orchestration. W.W. Norton, 2016.

[4] Yan Maresz, "On computer-assisted orchestration," *Contemporary Music Review*, vol. 32, 02 2013.

[5] Carmine-Emanuele Cella and Philippe Esling, "Open-source modular toolbox for computer-aided orchestration," in *Timbre conference*, 2018.

[6] Carmine-Emanuele Cella, "Orchidea: a comprehensive framework for target-based assisted orchestration," *submitted to Journal of New Music Research, under review*, 2020.

[7] Jeremy Reed, Yushi Ueda, Sabato Marco Siniscalchi, Yuuki Uchiyama, Shigeki Sagayama, and Chin-Hui Lee, "Minimum classification error training to improve isolated chord recognition.," in *ISMIR*, 2009, pp. 609–614.

[8] José R Zapata and Emilia Gómez, "Improving beat tracking in the presence of highly predominant vocals using source separation techniques: Preliminary study," in *Proc. 9th Int. Symposium on Computer Music Modeling and Retrieval, London*. Citeseer, 2012, pp. 583–590.

[9] Parag Chordia and Alex Rae, "Using source separation to improve tempo detection.," in *ISMIR*, 2009, pp. 183–188.

[10] Juan S Gómez, Jakob Abeßer, and Estefanía Cano, "Jazz solo instrument classification with convolutional neural networks, source separation, and transfer learning.," in *ISMIR*, 2018, pp. 577–584.

[11] Ivo Trowitzsch, Jalil Taghia, Youssef Kashef, and Klaus Obermayer, "Nigens general sound events database," Feb. 2019.

[12] BBC, "BBC sound effects archive," Accessed: 2021-03-03.

[13] Frederic Font, Gerard Roma, and Xavier Serra, "Freesound technical demo," in *ACM International Conference on Multimedia (MM'13)*, Barcelona, Spain, 21/10/2013 2013, ACM, pp. 411–412, ACM.

[14] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner, "Musdb18-a corpus for music separation," 2017.

[15] Fabian-Robert Stter, Antoine Liutkus, and Nobutaka Ito, "The 2018 signal separation evaluation campaign," 2018.

[16] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin W. Wilson, Jonathan Le Roux, and John R. Hershey, "Universal sound separation," *CoRR*, vol. abs/1905.03330, 2019.

[17] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji, "Open-Unmix - A Reference Implementation for Music Source Separation," *Journal of Open Source Software*, vol. 4, no. 41, pp. 1667, Sept. 2019.

[18] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 261–265.

[19] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis R. Bach, "Music source separation in the waveform domain," *CoRR*, vol. abs/1911.13254, 2019.

[20] Yi Luo and Nima Mesgarani, "Tasnet: Surpassing ideal time-frequency masking for speech separation," *CoRR*, vol. abs/1809.07454, 2018.

[21] Andrzej Cichocki and Anh-Huy Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions*, vol. 92-A, pp. 708–721, 03 2009.

[22] Cdric Fvotte and Jrme Idier, "Algorithms for nonnegative matrix factorization with the beta-divergence," 2011.

[23] Carmine Emanuele Cella, Daniele Ghisi, Vincent Lostanlen, Fabien Lvy, Joshua Fineberg, and Yan Maresz, "Orchideasol: a dataset of extended instrumental techniques for computer-aided orchestration," 2020.

[24] Carmine Emanuele Cella, Luke Dzwonczyk, Alejandro Saldarriaga-Fuertes, Hongfu Liu, and Hélène-Camille Crayencour, "A Study on Neural Models for Target-Based Computer-Assisted Musical Orchestration," in *2020 Joint Conference on AI Music Creativity (CSMC + MuMe)*, Stockholm, Sweden, Oct. 2020, Proceedings of the 2020 Joint Conference on AI Music Creativity.