Publié dans *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR, Delft, The Netherlands, 4-8 novembre 2019*, à citer pour faire référence à ce papier.

ESTIMATING UNOBSERVED AUDIO FEATURES FOR TARGET-BASED ORCHESTRATION

Jon Gillick[1] **Carmine-Emanuele Cella**[2] **David Bamman**[1]

[1]School of Information, University of California, Berkeley
[2]CNMAT, University of California, Berkeley

jongillick@berkeley.edu, carmine.cella@berkeley.edu, dbamman@berkeley.edu

ABSTRACT

Target-based assisted orchestration can be thought of as the process of searching for optimal combinations of sounds to match a target sound, given a database of samples, a similarity metric, and a set of constraints. A typical solution to this problem is a proposed orchestral score where candidates are ranked by similarity in some feature space between the target sound and the mixture of audio samples in the database corresponding to the notes in the score; in the orchestral setting, valid scores may contain dozens of instruments sounding simultaneously.

Generally, target-based assisted orchestration systems consist of a combinatorial optimization algorithm and a constraint solver that are jointly optimized to find valid solutions. A key step in the optimization involves generating a large number of combinations of sounds from the database and then comparing the features of each mixture of sounds with the target sound. Because of the high computational cost required to synthesize a new audio file and then compute features for every combination of sounds, in practice, existing systems instead estimate the features of each new mixture using precomputed features of the individual source files making up the combination. Currently, state-of-the-art systems use a simple linear combination to make these predictions, even if the features in use are not themselves linear.

In this work, we explore neural models for estimating the features of a mixture of sounds from the features of the component sounds, finding that standard features can be estimated with accuracy significantly better than that of the methods currently used in assisted orchestration systems. We present quantitative comparisons and discuss the implications of our findings for target-based orchestration problems.

1. INTRODUCTION

In many music information retrieval and signal processing contexts, we are required to reason about signals that are themselves the sum of multiple sources. Whether the summing comes from instruments in a multi-track recording, voices in a group conversation, or simply from noise in the signal, we generally need to consider the full set of sources that make up an audio signal.

Much work in MIR deals with pulling apart the sources in a signal, either in the most straightforward sense via source separation [3,4], or through any of a number of classification tasks such as tagging [11, 31], instrument recognition [14, 18], or automatic transcription [16, 26]. A separate body of work deals with the inverse problem, that of putting sources together: work in applications like assisted orchestration [8] and automatic mixing [13, 25] aims to guide people through the task of combining signals together with the help of a machine in the loop.

In the cases of both separation and combination, tasks can be solved presumably because the source components and the summed signal are sufficiently correlated; the more correlated a source is with the mixture, the easier it is to identify, and as more signals are summed together, correlations between the combination and any single source tend to diminish. In a computational setting, these correlations are of course measured through a set of features of the signals, whether they be hand-engineered features like FFT and MFCC, or modern features learned by neural networks.

There are some cases, however, in which we can observe the source signals of a mixture, but it is impractical or impossible to actually compute the features of the signal in question; these are the cases that we investigate in this work. Broadly speaking, there are two primary settings in which we may be unable to observe the features of audio signals:

1. We do not have access to the signals.
2. Computing the features for *all* relevant signals is computationally expensive.

The first setting is commonly encountered in MIR, in which, as with many fields centered around media that may be under copyright or other protections, it is quite common

for researchers to have access to pre-computed features but not to raw data itself. For example, the audio files of the million songs that comprise the Million Song Dataset [5], which serves as a benchmark for many common MIR tasks, cannot be legally distributed. Instead the data contains common audio features like MFCC, chroma, note onsets, and spectral centroids. [1] Though this dataset and others like it are attractive because of their size and scope, they have been of limited use as source material for constructing additional audio mixtures. As semi-supervised and self-supervised approaches to machine learning have become more competitive with fully supervised systems, large datasets even of weakly labeled source material are becoming more useful for research in areas like source separation [15, 24]; estimating the features of mixtures might be one path towards making use of this data in new contexts.

The second setting in which we cannot observe audio features, which is the focus of this work, is the case where the computational cost of calculating an exponential number of audio mixtures is prohibitively high. This computational bottleneck often arises in the aforementioned body of work that attempts to automatically combine signals together during the course of tasks like target-based orchestration. In this context, learning algorithms need to explore a combinatorial space of potential solution sets, making it infeasible to compute the real features of all possible mixtures of signals. Moreover, methods for narrowing down this set of possible solutions, such as reinforcement learning algorithms, are generally iterative, requiring online evaluation of a reward function before the next set of candidates can be explored. Because these methods both have a large solution space and need to be evaluated iteratively, features must be computed *on the fly*, making fast feature computation, or accurate estimation, a necessity.

In this work, we take steps to explore the potential of machine learning models for predicting audio features of a mixture of sounds that we are unable to observe, focusing on the task of target-based assisted orchestration [2, 6, 8]. Concretely, we consider models of the following form: given a feature function f and M individual signals $S_1, \ldots, S_M$, we learn mappings from input features $f(S_1), \ldots, f(S_M)$ to the true feature of the mixture $f(S_1 + \ldots + S_M)$.

In experiments, we examine one standard feature that is known to typically behave linearly when summed (FFT magnitude spectra) and one feature that is less well suited to linear approximation (MFFC coefficients), investigate the ability of different models to predict each feature across a varying number of mixtures ranging from 2 notes to 30, and discuss the implications of our findings for target-based assisted orchestration as well as for the broader range of scenarios in which real audio features cannot be observed.

Code to reproduce our results can be found at `https://github.com/jrgillick/audio-feature-forecasting`.

[1] The full list of fields can be accessed here: `https://labrosa.ee.columbia.edu/millionsong/pages/field-list`

2. TARGET-BASED ASSISTED ORCHESTRATION

Musical orchestration, and in many cases, music production, consists largely of choosing combinations of sounds, instruments, and timbres that support the narrative of a piece of music. Strong orchestration can bring a composition to life by emphasizing, clarifying, or perhaps questioning the elements of the music, and through this process, orchestration can often be a difference-maker to critical or commercial success [20, 21].

Different musical styles and composition environments have different constraints (for example, scores for live performance should only require the sounds of the instruments in the group, whereas the sounds available for use on a recording are only limited by their stylistic relevance), but fundamentally, finding the right set of sounds is important regardless of the context. For composers and producers, employing MIR systems during the orchestration process holds the potential to help spark inspiration, solve challenging problems, or save time.

Though the orchestral setting has been explored extensively in previous work, assisted orchestration methods hold the potential for application in other styles. For example, *layering* drum samples is common practice in music production, and MIR-based tools for drum sample search are beginning to make their way into professional toolkits [2]; existing methods for query-based drum sample retrieval [23] could be extended to consider mixtures of drum samples.

3. RELATED WORK

Existing systems for target-based assisted orchestration compute spectral similarities using standard spectral features [8] or perceptual descriptors [2], along with evolution-based methods for exploring the solution space.

Most relevant to our experimental setting is the implementation of the publicly available state-of-the art Orchidea system [10], which is based in part on a study conducted in [9] on predicting timbral features of combined sounds. This study found that for 3 features (Spectral Centroid, Spectral Spread, and Main Resolved Partials), and for mixtures of up to 4 sounds, predicting the features of the mixture by a linear combination of the source features both achieved a low error and did not vary as a function of the number of mixture components.

Since computing a linear combination has very low computational cost, this finding enables real-time estimation of thousands of candidate mixtures for use in online reinforcement learning, making tools like Orchidea practical for real-world use. The effects on the features of mixing many more components, along with the behavior of higher-dimensional and richer features, however, have not yet been investigated.

[2] https://www.xlnaudio.com/products/xo

4. EXPERIMENTS

4.1 Data

For our experiments, we use the OrchDB dataset of individually recorded musical notes from a variety of orchestral instruments. OrchDB is a streamlined subset of the Studio Online (SOL), dataset that has been optimized for assisted orchestration [30]. Collected as part of the Studio Online project at IRCAM, the full SOL data set contains over 117,000 instrument samples, including extended techniques and contemporary playing styles. OrchDB, which contains a curated subset of these samples, has been used for assisted orchestration since 2008 [7]; the contents of the data are summarized below:

- OrchDB contains about 20,000 notes with lengths ranging from about 1 second to 30 seconds.
- Instruments include bassoon, clarinet, flute, horn, oboe, saxophones, strings, trombone, trumpet, and tuba.
- Approximately 30 different playing styles are represented in OrchDB, such as ordinario, pizzicato, pizzicato Bartók, aeolian, Flatterzung, col legno battuto; brass instrument samples include a variety of different types of mutes.
- Notes across the pitch range are included, along with a range of dynamics from *ppp* to *fff*, including sforzato and crescendo.

4.2 Mixtures of Notes

To train and evaluate models for feature estimation, we partition the dataset into nonoverlapping subsets for training, development, and testing, choose 6 different numbers of mixture components M between 2 and 30, and then for each M, we synthesize a dataset of new audio files by adding together the raw waveforms of M randomly chosen notes. Finally, we divide the summed signals by M to keep the amplitudes of the mixture in the same range as those of the source files.

For each value of M, we synthesize 7500 new audio mixtures for training, 2000 for development, and 2000 for testing, creating these new mixtures after partitioning our data so that no source file that appears in the training set can be chosen as part of a mixture in the test set. After synthesizing the mixtures, we compute and store the real FFT and MFCC features for every mixture for use in training and evaluating our models.

4.3 Predicting Unobserved Features

With this data in hand, we explore several models for predicting the features of a mixture of audio signals given the features of the individual signals. In all experiments, given a feature function f and M individual signals $S_1, \ldots, S_M$, each model is trained to learn a mapping from input features $f(S_1), \ldots, f(S_M)$ to the true feature of the mixture $f(S_1 + \ldots + S_M)$.

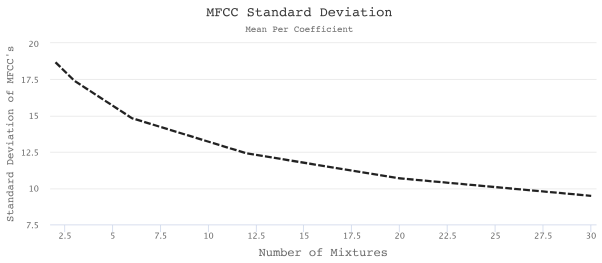

Figure 1. Standard Deviations (averaged across all 19 coefficients) of the real MFCC coefficients of mixtures of audio files. As M increases, the variance in the MFCC coefficients goes down.

5. MODELS

5.1 Features

For our modeling experiments, we choose two standard features: 1024-dimensional FFT magnitude spectra and 19-dimensional MFCC coefficients (we discard the first of 20 MFCC coefficients). Our choice of features is meant to capture the most common MIR settings, so we use the default FFT and MFCC dimensions specified in the Librosa library [22] and compute the features on audio files sampled at 22050 Hz using the default window size (2048 samples) and hop size (512 samples) of the Librosa implementations. We then follow [8] in flattening both the FFT and MFCC features from 2-dimensional time-frequency representations into 1-dimensional feature vectors by taking a linear combination of the features at each frame, weighted by the RMS energy at the corresponding frame.

This method of averaging over time allows us to summarize the spectral characteristics of signals with different lengths using a single feature feature vector, while at the same time ignoring the quieter parts of the signal. In addition, we preserve the interpretability of the FFT and MFCC features through this process, which is particularly useful for inspecting and analyzing our model outputs. Of course, the downside of this preprocessing step is that we discard all time-domain information, so we are unable to predict anything about the envelope or movement of the sound. Depending on the source material and the downstream application, different preprocessing choices might be more appropriate than averaging over time; for example, unpitched percussive sounds require different modeling choices from pitched material. Since our data consists of mostly pitched notes from orchestral instruments, however, we follow the convention of the assisted orchestration literature by focusing on timbre independent of time.

Finally, before training or evaluating models, in order to best align our quantitative results to the expected perceptual results with regards to timbre, we normalize the FFT feature vectors such that the maximum value is 1. Although in the FFT case, relative magnitudes are known to be more correlated with perception of timbre than the raw amplitudes are, magnitudes of MFCC coefficients are important descriptors of timbre, so we do not normalize the

MFCC's, instead predicting the real values.

5.2 Baseline

As a baseline, we compute the element-wise mean of the feature vectors over the entire training set. This vector is computed once for each value of the number of mixture components M. Models that perform better than this baseline can be said to be capturing some useful information about how the features sum together. One important factor to take into account when evaluating results it that as we increase M, mixing more and more notes together, the variance in the features of the mixtures decreases, making the predictive task appear easier. The MFCC features, because they are much lower dimensional than the FFT's, are especially effected by this change in variance; the higher dimensional FFT features exhibit the same trend but to a smaller extent, as they can capture a wider range of combinations of signals. For this reason, in Section 6, we report error metrics as a percentage relative to the error metric of this baseline at the corresponding value of M. Concretely, an error of 0.5 would mean that, averaged over the test set, the sum of squared errors of our predictions was equal to half of the sum of squared errors obtained by always predicting the mean of the data set.

5.3 Linear Combination

The first model we examine is the linear combination of features proposed in [9], which is currently used in state-of-the-art assisted orchestration systems [10]. This model implicitly assumes that for a feature f, the feature of the sum is approximately equal to the sum of the features:

$$w_1 f(S_1) + \ldots + w_M f(S_M) \approx f(w_1 S_1 + \ldots w_M S_M) \quad (1)$$

When features are linear or can be well approximated linearly, this method can be a strong baseline. Especially with high dimensional features like our 1024 FFT magnitudes, subtle details that might be difficult to summarize in an intermediate representation can be easily preserved with a linear model.

For this model, we combine source features in two ways, first by taking the element-wise mean of the M feature vectors as shown in Equation (2) and second by weighting the features by the corresponding RMS energies $a_1 \ldots a_M$ of the component signals as in Equation (3):

$$\bar{f} = \frac{1}{M} \sum_{f_i} \quad (2)$$

$$\tilde{f} = \frac{\sum f_i a_i}{\sum a_i} \quad (3)$$

5.4 MLP

Particularly for nonlinear features, it is reasonable to expect that nonlinear models have the potential to make better estimates. We train multilayer perceptron (MLP) neural networks to predict both FFT and MFCC features. For these models, we use a single hidden layer, and we minimize the mean squared error (MSE) between the predictions and the targets. For the FFT models, in order to constrain the network to output magnitudes between 0 and 1, we use a ReLU activation followed by a L_∞ normalization layer as the last stage in our network. Although we found empirically that sigmoid activations gave similar accuracies, these choices match better with the intuition of normalization performed in preprocessing. We train all our neural network models with Tensorflow [1] and Keras [12], Dropout [27], and the Adam optimizer [19].

Because we are interested in testing our methods on a variable number of audio mixtures M, we train separate MLP models for each value of M. As M increases, the input size and number of parameters in the network increases accordingly; with a feature of dimension D and a hidden layer of size H, the first layer of these networks has $D \times M \times H$ trainable parameters.

5.5 LSTM

As we increase the number of mixtures M, a recurrent network architecture is a natural choice to reduce the number of parameters needed. Intuitively, if a network can learn to estimate the sum of two signals, the same network should be able to process M signals in sequence over M steps by estimating the sum of one signal with the sum of all the signals processed so far.

5.5.1 Ordered Sets

Because the true sum of M signals is independent of the order in which they are combined, we experiment with two approaches inspired by the literature on sequence models for sets. First, even when no true ordering exists, previous results demonstrate that the ordering of inputs to factorized probabilistic models still affects the ability of models to learn [29]. In the case where two semantically valid orderings exist, empirical results from machine translation show that simple changes to the ordering, such as reversing the words in a sentence, can significantly affect model performance [28]. Based on these results, for this variant of the model, we sort the signals by their L_2 norm before passing them to LSTM model, such that the source signal with the highest energy is observed at the final timestep before outputting a final prediction.

5.5.2 Unordered Sets

Although previous work points to the benefits of ordering the signals in a consistent way, fixing an ordering prevents us from implementing a simple but potentially powerful form of data augmentation - randomly shuffling the order of mixtures during training. We empirically test the relative benefits of these two options, reporting results for both ordered and unordered inputs with the same LSTM archicture.

5.5.3 Residual Connections

Finally, we experiment with one more variation of our LSTM model, in which we add a residual connection [17]

between the model inputs at each timestep and the outputs of the LSTM layer, which allows information to pass directly from the input to the final layer without having to be mediated by the nonlinear structure of the LSTM. Intuitively, to the degree to which features are linear, this connection should provide the model with the option to directly sum up the features as part of its computation.

6. RESULTS

We train and evaluate all of the models across 6 different numbers of mixtures M ranging from 2 to 30, summarizing the results in Tables 1 and 2 and displaying the trends across values of M in Figures 2 and 3.

6.1 Predicting FFT Features

As demonstrated in Figure 2, the linear combination outperform the neural methods for values of M between 2 and 12, but the LSTM models make up ground and ultimately begin to overtake the linear combinations at $M = 20$ mixtures. All of the models in the FFT setting trend up in error towards the baseline as the number of mixtures increases; with $M = 30$, all models except for the residual LSTM cross the threshold of the baseline. These results indicate several findings:

- While the ordering in which the mixtures are passed to the LSTM model does not appear to make a significant difference here, the residual LSTM model outperforms the rest of the neural methods at all values of M, demonstrating increasingly large gains as the number of mixtures goes up. This suggests that the residual connection may be enabling the model to exploit the linearity of features when it is advantageous to do so, while maintaining flexibility to make better estimations once the signal from the linearity of the feature fades.
- In confirmation with previous findings [9], these results suggest that linear approximations of FFT features can be quite accurate. As the number of mixtures increases, however, these estimates worsen; by $M = 30$, the linear approximations are no better than random.
- Although estimating a high dimensional feature like the FFT is clearly a challenging task as many streams of audio are mixed together, these results show that neural models do possess the potential to estimate these features to some degree even in settings with many different sources.

6.2 Predicting MFCC Features

Unlike in the case of the FFT features, all of the neural models outperform the linear combination for both small and large numbers of mixtures, and as shown in Figure 3, with more than 6 mixtures, linear combinations of MFCC features no longer contain a useful signal. We detail our findings from the MFCC experiments below:

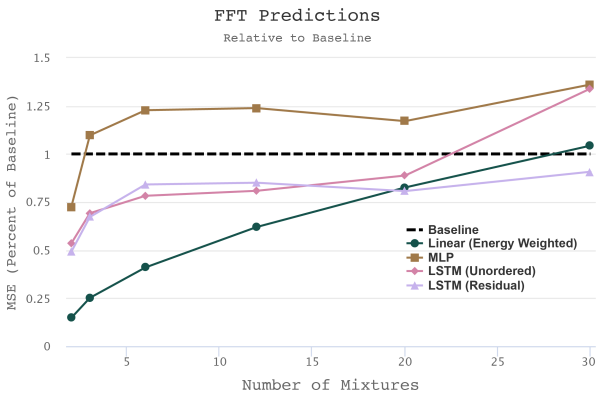

Figure 2. The linear models work well for predicting FFT features of small numbers of mixtures, but at around $M = 20$ mixtures, the best performing LSTM model overtakes the linear combination.

- Because MFCC features are nonlinear, it is not surprising that nonlinear models are able to predict them better than the linear combination. Relative to the baseline, however, we can see that for mixtures of 2, 3, and even 6 different sources, a linear combination of MFFC's can still be reasonably accurate. This suggests that in some cases, MFFC features do behave approximately linearly when summed.
- In contrast to the FFT setting, the residual LSTM does not appear to offer any gains in comparison with the other LSTM models. Perhaps because of the much smaller dimension of the features, the Unordered LSTM model, which we train with data augmentation by randomizing the order in which mixtures of processed, performs best.
- As M continues to increase, the accuracies of the LSTM models flatten out rather than continuing to approach the baseline. This trend suggests that even when dozens of notes are mixed together, we may be still able to estimate certain features of these mixtures based only on the features of the source files.

6.3 Computation Time

While the exact computation time of FFT or MFFC features depends on the implementation, the length of the audio files, and the availability of parallel processing, estimating features with the networks we explore is, in practice, significantly faster than computing the real features. Though it is beyond the scope of this paper to report results on a comprehensive list of hardware and software configurations, as a point of reference, Table 3 displays running times for parallel computation on our research server containing 20 CPU's and one Tesla K40 GPU.

7. CONCLUSIONS

In this work, we experiment with neural models for predicting unobserved audio features based on precomputed

Model	2 Mixtures	3 Mixtures	6 Mixtures	12 Mixtures	20 Mixtures	30 Mixtures
Baseline	1	1	1	1	1	1
Linear (Mean)	0.44	0.60	0.73	0.85	0.99	1.16
Linear (Energy-Weighted)	**0.15**	**0.25**	**0.41**	**0.62**	0.83	1.04
MLP	0.72	1.10	1.23	1.24	1.17	1.36
LSTM (Unordered)	0.54	0.69	0.78	0.81	0.89	1.34
LSTM (Ordered)	0.55	0.73	0.85	0.88	0.86	1.35
LSTM (Residual)	0.49	0.67	0.84	0.85	**0.81**	**0.91**

Table 1. Mean Squared Error for predicting FFT features across different numbers of mixtures.

Model	2 Mixtures	3 Mixtures	6 Mixtures	12 Mixtures	20 Mixtures	30 Mixtures
Baseline	1	1	1	1	1	1
Linear (Mean)	0.43	0.58	0.81	1.03	1.32	1.54
Linear (Energy-Weighted)	0.36	0.59	0.94	1.30	1.69	2.02
MLP	0.42	0.55	0.71	0.79	0.88	0.93
LSTM (Unordered)	**0.30**	**0.46**	**0.57**	**0.63**	**0.71**	**0.70**
LSTM (Ordered)	**0.30**	**0.46**	0.61	0.66	0.73	0.73
LSTM (Residual)	0.32	0.47	0.64	0.71	0.77	0.77

Table 2. Mean Squared Error for predicting MFCC features across different numbers of mixtures.

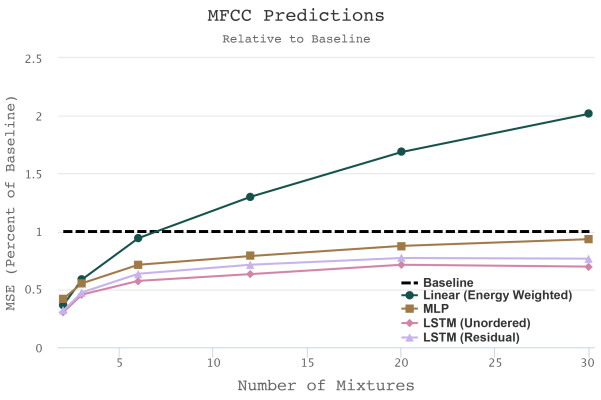

Figure 3. The neural models outperform the linear combinations significantly, widening the gap as M increases.

Feature	**Real** (CPU x 20)	**LSTM** (CPU x 20)	**LSTM** (GPU x 1)
FFT (Mix 2)	14.71	0.32	0.07
FFT (Mix 30)	14.71	4.75	1.10
MFCC (Mix 2)	73.50	0.03	0.01
MFCC (Mix 30)	73.50	0.34	0.15

Table 3. Time in seconds to compute or estimate energy-weighted FFT or MFCC features for the 2000 audio files in the test set using parallel processing. FFT (Mix 30) refers to the FFT feature of a mixture of 30 audio files, which requires 30 autoregressive LSTM steps. LSTM refers to the Residual LSTM model.

features of source files in a mixture, examining the cases of FFT features, which should behave linearly when summed, as well as MFCC's, which are known to be nonlinear. We find that in the case of nonlinear features, LSTM models significantly outperform the methods currently in use for feature estimation, and further, that while the linear predictors perform well for small numbers of mixtures, as we mix more and more signals together, the neural models begin to outperform the linear methods as well.

Our results suggest that we may be able to improve current assisted orchestration systems [10] by replacing feature estimation components with LSTM-based nonlinear predictors. As with any real-world problem that involves perceptual similarity rather than comparisons in a feature space, however, more work is needed to understand how these models may interact with other components of systems they may be embedded in. Deep neural network models can and do adapt to any correlations present in the data, so understanding how these models are making there estimates may be important.

Beyond tasks like assisted orchestration in which we cannot always observe the features of an audio file because of computational limitations, we hope that future work may be able to take advantage of the methods for feature estimation explored here in order to make creative use of data like the Million Song Dataset, for which precomputed features are available but raw data cannot be distributed.

8. ACKNOWLEDGMENTS

The research reported in this article was supported by the Hellman Family Faculty Fund and by resources provided by NVIDIA. We also thank the anonymous reviewers for their valuable feedback.

9. REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Aurelien Antoine and Eduardo Miranda. A perceptually orientated approach for automatic classification of timbre content of orchestral excerpts. *The Journal of the Acoustical Society of America*, 141(5):3723–3723, 2017.

[3] Shoko Araki, Francesco Nesta, Emmanuel Vincent, Zbyněk Koldovskỳ, Guido Nolte, Andreas Ziehe, and Alexis Benichoux. The 2011 signal separation evaluation campaign (sisec2011):-audio source separation. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 414–422. Springer, 2012.

[4] Adel Belouchrani, Karim Abed-Meraim, J-F Cardoso, and Eric Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on signal processing*, 45(2):434–444, 1997.

[5] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. *In Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591––596, 2011.

[6] Marcelo Caetano, Asterios Zacharakis, Isabel Barbancho, and Lorenzo J Tardón. Leveraging diversity in computer-aided musical orchestration with an artificial immune system for multi-modal optimization. *Swarm and Evolutionary Computation*, 2019.

[7] Grégoire Carpentier. *Approche computationnelle de l'orchestration musciale-Optimisation multicritère sous contraintes de combinaisons instrumentales dans de grandes banques de sons*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2008.

[8] Grégoire Carpentier, Gérard Assayag, and Emmanuel Saint-James. Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach. *Journal of Heuristics*, 16(5):681–714, 2010.

[9] Grégoire Carpentier, Damien Tardieu, Jonathan Harvey, Gerard Assayag, and Emmanuel Saint-James. Predicting timbre features of instrument sound combinations: application to automatic orchestration. *Journal of New Music Research*, 39(1):47–61, 2010.

[10] Carmine-Emanuele Cella and Philippe Esling. Open-source modular toolbox for computer-aided orchestration. 2018.

[11] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *In Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[12] François Chollet et al. Keras, 2015.

[13] Brecht De Man and Joshua D Reiss. A semantic approach to autonomous mixing. *Journal on the Art of Record Production (JARP)*, 2013.

[14] Antti Eronen and Anssi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, volume 2, pages II753–II756. IEEE, 2000.

[15] Zhe-Cheng Fan, Yen-Lin Lai, and Jyh-Shing R Jang. Svsgan: Singing voice separation via generative adversarial network. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 726–730. IEEE, 2018.

[16] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *In Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Eric Humphrey, Simon Durand, and Brian McFee. Openmic-2018: an open dataset for multiple instrument recognition. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Stephen McAdams. Perspectives on the contribution of timbre to musical structure. *Computer Music Journal*, 23(3):85–102, 1999.

[21] Stephen McAdams. Timbre as a structuring force in music. In *Proceedings of Meetings on Acoustics ICA2013*, volume 19, page 035050. ASA, 2013.

[22] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[23] Adib Mehrabi, Keunwoo Choi, Simon Dixon, and Mark Sandler. Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 356–360. IEEE, 2018.

[24] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[25] Joshua Reiss and Øyvind Brandtsegg. Applications of cross-adaptive audio effects: automatic mixing, live performance and everything in between. *Frontiers in Digital Humanities*, 5:17, 2018.

[26] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 591–597, 2016.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[29] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[30] Rolf Wöhrmann and Guillaume Ballet. Design and architecture of distributed sound processing and database systems for web-based computer music applications. *Computer Music Journal*, 23(3):73–84, 1999.

[31] Yusuf Yaslan and Zehra Cataltepe. Audio music genre classification using different classifiers and feature selection methods. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 573–576. IEEE, 2006.