# Generating data to train convolutional neural networks for classical music source separation

**Marius Miron, Jordi Janer, Emilia Gómez**
Music Technology Group, Universitat Pompeu Fabra, Barcelona
firstname.lastname@upf.edu

## ABSTRACT

Deep learning approaches have become increasingly popular in estimating time-frequency masks for audio source separation. However, training neural networks usually requires a considerable amount of data. Music data is scarce, particularly for the task of classical music source separation, where we need multi-track recordings with isolated instruments. In this work, we depart from the assumption that all the renditions of a piece are based on the same musical score, and we can generate multiple renditions of the score by synthesizing it with different performance properties, e.g. tempo, dynamics, timbre and local timing variations. We then use this data to train a convolutional neural network (CNN) which can separate with low latency all the renditions of a score or a set of scores. The trained model is tested on real life recordings and is able to effectively separate the corresponding sources. This work follows the principle of research reproducibility, providing related data and code, and can be extended to separate other pieces.

## 1. INTRODUCTION

Source separation assumes recovering the sources of the signals from a mixture. For audio applications the two main areas are speech enhancement or recognition, and separation of musical sources [1]. Regarding the latter, it allows for a range of interesting applications such as remixing, up-mixing or 3D concerts using VR technologies [2].

In this paper, we study the case of low latency monaural source separation of classical music recordings, where our goal is to extend the instrument enhancement applications developed during the PHENICX project [3, 4] to a low latency scenario, e.g. video streaming applications. In this scenario, we need to provide real-time source separation during a music concert, allowing the listener to focus on specific musical instruments. For that, we can benefit from previous information about the target piece and meta-data associated with it.

In this context, we assume that the sources are harmonic, playing harmonically and rhythmically related phrases, highly correlated in time and frequency which is a challenging scenario if compared to separating between pitched instruments and drums [5]. However, we rely on the assumption that the instruments for a given musical piece are known in advance, and one can include timbre information to guide the separation. This scenario is commonly known as timbre-informed source separation [6].

Matrix decomposition techniques have been traditionally used for timbre-informed music source separation. In the case of Non-negative matrix factorization (NMF), instruments are assigned a set of timbre basis which are previously learned and kept fixed during the separation stage [6]. Although successful, informed NMF approaches are computationally intensive, which makes them difficult to use in a low latency scenario.

Data-driven approaches using deep neural networks involve learning binary or soft masks corresponding to the target sources [7–11]. Moreover, a neural network framework can process short audio windows in streaming in a causal manner which makes these systems suitable for low-latency applications. Furthermore, compared to the NMF, neural networks such as convolutional neural networks or recurrent neural networks have the advantage of modeling the temporal context.

For the source separation approaches based on deep learning, the timbres corresponding to the instruments are learned from multi-microphone tracks. These approaches usually require large amounts of training data, which is scarce for music source separation. In addition, obtaining training data for classical music in form of real-life recordings is difficult because the mixtures should be based on isolated recordings [4, 12, 13]. However, as the classical music repertory has been traditionally assembled with scores, training data can be generated by synthesizing a target score according to several performance factors comprising (and not limited to) tempo, dynamics, timbre of the instruments, and synchronization between musicians, which induce local timing variations. The principle guiding the present study is that these factors differentiate between various renditions and characterize musical performances [14], and training a neural network with such synthetic data generates a more robust model which can be used to separate real-life renditions.

As various classical music pieces are played by different instruments and neural networks architectures are not flexible in accommodating a variable number of sources, it is not possible to train an universal model for classical music. Traditionally, in a timbre-informed case, models

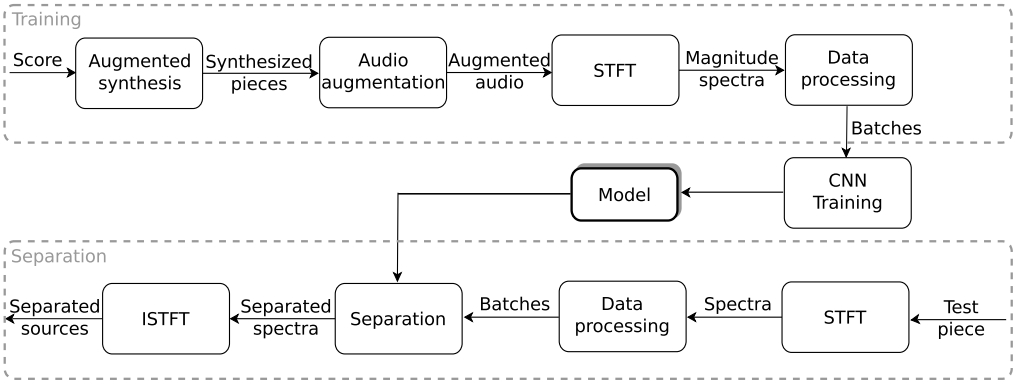


Figure 1. Proposed separation system

are trained for a fixed combinations of instruments [7–11]. For classical music, we constrain the timbre-informed system to the possible combinations of notes and instruments which exist in a set of scores which have the same instruments. We denote this case as score-constrained source separation. This is a more general case than score-informed source separation [15], as score is used solely during the training phase to synthesize renditions.

We propose a timbre-informed and score-constrained system to train neural networks for monaural source separation of classical music mixtures. We generate training data through synthesis of a set of scores by considering the following factors: tempo, timbre, dynamics and circular shifting (local timing variations). In addition, we extend the timbre-informed low latency system we proposed in [11] to the case of score-constrained classical music source separation. As argued in [11], the model we use has considerable less parameters, is easier to train, and can be used in a low latency scenario. We evaluate several training scenarios on Bach chorales pieces played by four instruments.

The remainder of this paper is structured as follows. In Section 2 we discuss the relation of the proposed method with the previous work. We present the architecture of the neural network in Section 3.1, the parameter learning in Section 3.2, the data processing in Section 3.3, and the proposed approaches for generating training data in Section 3.4. The evaluation dataset, method, parameters and results are discussed in Section 4. We conclude with an outlook on the presented system in Section 5.

## 2. RELATION TO PREVIOUS WORK

As mentioned before, training NMF basis for source separation is carried out either by score synthesis [13, 16] or by analytic approaches which assume learning registers for all considered instruments [6, 15]. In a similar way, we can synthesize the desired pieces or use samples for notes to train a neural network. However, in contrast to NMF, we need to train the network with examples containing the mixtures along with the targeted sources which cover possible cases one might encounter in real-life or in the test dataset. In addition, a deep learning model is optimized and heavily dependent on the training data and needs to cover more examples at the training stage than the parametric approaches embodied by NMF.

As a solution to data scarcity and to improve generalization and robustness, deep learning systems rely on data augmentation techniques by applying transformations to the existing data [17, 18]. Choosing realistic transformations depends on the possible axes on which data varies and on the task, e.g. pitch shifting, time stretching, loudness, randomly mixing different recordings or background noise. Although our approach can be seen as a data augmentation strategy, we generate new data according to transformations that make more sense for source separation, instead of augmenting existing data with techniques popular in other classification tasks [17, 18]. Furthermore, except circular shifting [7], we use different transformations than previous deep learning approaches. We consider music samples played with different dynamics and by different instruments (timbre), which is different than simply changing the loudness or the amplitude of a sample. Additionally, we mix the audio files and not the resulting spectrograms to account for phase cancellation.

Besides [10], deep learning approaches are evaluated in a cross-validation manner [7, 9, 11]. However, we claim that due to the small size of the classical music datasets used for evaluation, these methods are more likely to perform poorly in real-life scenarios where we can have different tempos, timbre, or dynamics. Therefore, varying training data with these factors adds robustness.

## 3. METHOD

The diagram for the proposed system is depicted in Figure 1. For the training stage, we depart from the score of a given piece which we synthesize at various combinations of tempos, timbres and dynamics. We then generate additional data by mixing circular shifted versions of the audio tracks for the corresponding sources. More details on data generation are given in Section 3.4.

### 3.1 Network architecture

The convolutional neural network (CNN) used in this paper is a variation of the one we formulated in [11] and comprises two convolutional layers, a dense ”bottleneck” layer having a low number of units, a second dense layer fol-

lowed by the inverse of the first two layers for each of the target sources. We kept similar filter shapes in order to preserve a low number of parameters and the low latency of the model.

The input of the network is a STFT magnitude spectrogram $\mathbf{X} \in \mathbb{R}^{TF}$, where $T$ is the number of time frames (the time context modeled) and $F$ is the number of frequency bins. This input is passed through a vertical convolution layer comprising $P_1 = 30$ filters of size $(1, 30)$ with stride 5, thus yielding the output $\mathbf{X_1} \in \mathbb{R}^{P_1 T F_1}$, where $F_1 = (F - 30)/5 + 1$. Then, we have a horizontal convolution comprising $P_2 = 30$ filters of size $(\frac{2}{3} \cdot T, 1)$ with stride 1 which yields the output $\mathbf{X_2} \in \mathbb{R}^{P_2 T_2 F_1}$, where $T_2 = (T - \frac{2}{3} \cdot T)/1 + 1$. This layer is followed by a dense "bottleneck" layer of size $F_3 = 256$, which has an input of size $P_2 \cdot T_2 \cdot F_1$.

As we need to reconstruct the output for each of the sources $j = 1 : J$, where $J$ is the total number of separated sources, we perform the inverse operations for the horizontal and the vertical convolutions. To match the dimensions needed to compute the inverse of the second layer, we need to create a dense layer of size $P_2 \cdot T_2 \cdot F_1$ features for each source, thus having $J \cdot P_2 \cdot T_2 \cdot F_1$ units. Consequentially, for each of the estimated sources we perform the inverse operation of the convolution layers, i.e. the deconvolution, using the same parameters learned by these layers. The final inverse layer yields a set of estimations $\mathbf{E_j} \in \mathbb{R}^{TF}$, with $j = 1 : J$.

The convolutional layers have a linear activation function and the dense ones a rectified linear unit activation function, as in [11].

### 3.2 Parameter learning

Similarly to [7,11], the network yields the estimations $\mathbf{E_j} \in \mathbb{R}^{TF}$ from which we derive the soft masks $\mathbf{M_j} \in \mathbb{R}^{TF}$, for each source $j = 1 : J$:

$$\mathbf{M_j} = \frac{|\mathbf{E_j}|}{\sum_{j=1}^{J} |\mathbf{E_j}|} \tag{1}$$

Then, the soft masks are multiplied with the original magnitude spectrogram to obtain the corresponding estimated magnitude spectrograms for the sources: $\hat{\mathbf{X}}_\mathbf{j} = \mathbf{M_j} \times \mathbf{X}$.

The parameters of the network are updated using mini-batch Stochastic Gradient Descent with *AdaDelta* algorithm [19] according to the following loss function which minimizes the Euclidean distance between the estimated $\hat{\mathbf{X}}_\mathbf{j}$ and target $\mathbf{X_j}$ sources: $Loss = \sum_{j=1}^{J} \|\hat{\mathbf{X}}_\mathbf{j} - \mathbf{X_j}\|^2$

Because the target sources are harmonic, highly correlated and playing consonant musical phrases, we do not include the dissimilarity cost between the sources as in [11].

### 3.3 Data processing

**a) Training:** Generating training data assumes slicing the spectrograms of the mixture $\mathbf{X}(t, f)$ and the target spectrograms $\mathbf{X}_j(t, f)$ in overlapping blocks of size $T$, for the time frames $t = 1 : \hat{T}$ and frequency bins $f = 1 : F$, where $\hat{T}$ is the total number of time frames and $F$ is the number of frequency bins of $\mathbf{X}$. We summarize the procedure in Algorithm 1.

**Algorithm 1** Generating training data

```
for each piece in the training set do
    Compute STFT of the piece
    Initialize total number of blocks to B = (T̂ − T)/O, where O is the step size.
    for b=1:B do
        Slice X^b = X(b ∗ O : b ∗ O + T, :)
        Slice X_j^b = X_j(b ∗ O : b ∗ O + T, :)
    end for
end for
Generate batches by randomly grouping blocks.
```

**b) Separation:** Separating a target signal involves slicing a magnitude spectrogram. Then, we obtain an estimation for the sources by feed forwarding the blocks through the network, and we overlap-add the blocks to obtain the magnitude spectrograms. The steps are presented in the Algorithm 2.

**Algorithm 2** Separation of a piece

```
Compute complex valued STFT; keep the phase.
Initialize total number of blocks to B = (T̂ − T)/O, where O is the step size.
for b=1:B do
    Slice X^b = X(b ∗ O : b ∗ O + T, :)
    Feed-forward X^b through the CNN, obtaining the magnitude spectrograms X̂_j^b, for the sources j = 1 : J.
end for
for j=1:J do
    Compute X̂_j by overlap-adding X̂_j^b.
end for
Use the phase of the original signal and compute the estimated sources with the inverse overlap-add STFT.
```

If we consider that the sources $y_j(t), j = 1 : J$ are linearly mixed, such that $x(t) = \sum_{j=1}^{J} y_j(t)$, then we can use the original phase of the audio mixture to obtain the signals associated to the sources $y_j(t)$, with an inverse overlap-add STFT, as in [13].

### 3.4 Data generation

The proposed system is trained with data generated by synthesizing a set of scores at different tempos, using samples of different timbres and dynamics, and then applying circular shifting to the resulting audio files, to account for local timing variations. In this paper, we consider four factors:

**a) Tempo:** The renditions of a classical music piece vary in terms of tempo. Therefore, in order to make the model more robust to variations in tempo, we synthesize the score at different tempos, i.e. we adjust the note duration for each note corresponding to each instrument in the score. The possible tempo variations are represented in the vector $a = \{a_1, a_2, \ldots, a_A\}$, where $A$ is the total number of considered tempos.

**b) Timbre:** Different recording conditions, instruments, players or playing styles can yield differences in timbre. A single timbre variation comprises samples of the notes in

the designated range of an instrument played by a different musician and recorded in different conditions. All timbre variations are stored in the vector $c = \{c_1, c_2, \ldots, c_C\}$, where $C$ is the total number of timbre variations. Hence, when synthesizing a note of a given instrument we can choose between the $c$ samples corresponding to a different timbre.

**c) Dynamics:** The dynamics of a piece can change between renditions of a piece. Furthermore, variations in dynamics induce changes in loudness and timbre. Thus, we synthesize using samples representing various levels of dynamics in order to make our model more robust to this variable. The dynamics variations are represented in the vector $d = \{d_1, d_2, \ldots, d_D\}$, where $D$ is the dynamic range.

**d) Circular shifting:** In contrast to tempo changes which account for global tempo variations, circular shifting accounts for local timing variations. The synthetic pieces lack human expressiveness or even small errors which one might encounter in a real performance and for which we try to account for using this transformation. Circular shifting takes place after the synthesis of the audio and is applied to each of the target sources. Considering an audio signal of $S$ samples, a circular shift is a permutation $\sigma$ with $\hat{S}$ samples such that $\sigma(i) \equiv (i + \hat{S}) \mod S$, for all samples $i = 1 : S$.

We can have various combinations between different shifting steps for the sources. For each combination we generate a new mixture by summing up the circular shifted audio tracks. The possible circular shifts are represented in the vector $e = \{e_1, e_2, \ldots, e_E\}$, where $E$ is the total number of considered circular shifts.

The space comprising all possible combinations between the considered variables and $j = 1 : J$ target instruments is the Cartesian product $a \times c \times d \times e \times j$, having in total $A{\cdot}C{\cdot}D{\cdot}E{\cdot}J$ possibilities. If the number of combinations is too large and we can not generate all of them, we randomly sample from the Cartesian space.

In the Algorithm 3 we detail the procedure used to generate a rendition, i.e. a multi-track recording comprising audio vectors $x_j$ for each instrument $j = 1 : J$. We depart from a given score which has a set of notes $n_j = 1 : N_j$, where $N_j$ is the total number of notes for instrument $j$. Then we synthesize each note considering a given combination comprising a tempo $\hat{a}$, a set timbres $\hat{c} = (\hat{c}_1, \ldots, \hat{c}_j)$, dynamics $\hat{d} = (\hat{d}_1, \ldots, \hat{d}_j)$ and circular shifting $\hat{e} = (\hat{e}_1, \ldots, \hat{e}_j)$ for each instrument $j = 1 : J$.

**Algorithm 3** Generating a rendition from a score

```
Initialize the tempo â, timbre ĉ, dynamics d̂ and circular shifting ê variables.
for j = 1 : J do
    Initialize the audio vector x_j
    for each note n_j = 1 : N_j do
        Adjust the note duration in the score to the tempo â.
        Query the database/audio engine for a sample of timbre ĉ_j with the dynamics d̂_j.
        Synthesize the audio vector corresponding to the given note n_j and paste it in x_j at the onset and offset times.
    end for
    Apply the circular shift ê_j to x_j.
end for
```

## 4. EVALUATION

### 4.1 Datasets

For evaluation purposes we use ten Bach chorales from the Bach10 dataset, played by bassoon, clarinet, saxophone, violin, in the Bach10 dataset [12] which has been widely used in the evaluation of source separation. Each piece has a duration of $\approx 30$ seconds and is accompanied by the original score and a version of this score which is aligned with the rendition.

We synthesize the scores in the Bach10 dataset with two methods:

**a) Sibelius:** We use the library provided by software Sibelius [1], which uses sample-based synthesis. In this case we have $C = 1$ timbre and $D = 1$ for dynamics. Moreover, we use three levels of tempo $a = \{80, 100, 120\}$ BPM, and three of circular shift $e = \{0, 0.1, 0.2\}$ seconds. The dataset is made available through Zenodo [2].

**b) RWC:** In this experiments we query the RWC database [20] for samples corresponding to the notes played by the instruments in Bach10 for the original score $a = \{100\}$ BPM. The samples are played by three different musicians $c = \{1, 2, 3\}$, at three levels of dynamics $d = \{$*forte, mezzo, piano*$\}$, and various styles. For this experiment we picked the *normal* style of playing. The circular shifting considered for this method is $e = \{0, 0.1, 0.2\}$ seconds.

Because we want to isolate the influence of timbre and dynamics from the influence of tempo, we also synthesize the ten pieces using the score perfectly aligned with the audio using the synthesis methods **a** and **b**,

### 4.2 Evaluation setup

**a) Evaluation metrics:** We used the evaluation framework and the metrics are described in [21] and [22] : *Source to Distortion Ratio* (SDR), *Source to Interference Ratio* (SIR), and *Source to Artifacts Ratio* (SAR).

**b) Parameter tuning:** For the STFT we used a Blackman-Harris window. We experimentally set the length of the window to $4096$ samples, which, at a sampling rate of 44.1 KHz corresponds to 96 milliseconds (ms), and a hop size of $512$ samples ($11ms$). We observed that, for the Bach10 dataset, the higher the STFT resolution the better the results in separation, especially for the instruments which have a lower range, such as bassoon.

We kept the time context modeled by the CNN to $T = 30$ frames, which showed best performance in [11] with the step size $O = 25$. The number epochs, which represents the number of times all the examples of the dataset are seen by the network, is experimentally determined for each training experiment. As a general rule, we stop training if the cost between two epochs drops below $0.05$. The size of a mini-batch is set to 32.

Additionally, various hypotheses were tested and were not proven to improve the separation: CNN architectures having separate filters for each instrument, $l_2$ regularization and dropout for the dense layers, and Kullback-Leibler

[1] http://www.avid.es/sibelius

[2] https://zenodo.org/record/321361#.WKxZKd-i7J8

and Itakura Saito distances instead of Euclidean distance in the cost function.

**c) Hardware and software tools:** Our work follows the principle of research reproducibility, so that the code used in this paper is made available [3] . It is built on top of Lasagne, a framework for neural networks using Theano [4] . We ran the experiments on a computer with GeForce GTX TITAN X GPU, Intel Core i7-5820K 3.3GHz 6-Core Processor, X99 gaming 5 x99 ATX DDR44 motherboard.

### 4.3 Experiments

#### *4.3.1 Convolutional Neural Networks*

In order to test various data generation approaches, we train the CNN system in Section 3.1 with different data:

**a) CNN Bach10:** We train with all the 10 pieces in the Bach10 dataset.

**b) CNN leave one out (LOOCV) on Bach10:** We train with nine pieces of Bach10 and test on the remaining piece, repeating this for all the 10 pieces of the dataset.

**c) CNN Sibelius:** We train with all the synthetic pieces in the generated dataset described in Section 4.1a.

**d) CNN Sibelius GT:** We train with all the synthetic pieces in generated dataset described in Section 4.1a, synthesized with the score perfectly aligned with the rendition.

**e) CNN RWC:** We train with all the synthetic pieces in the generated dataset described in Section 4.1b. Since the number of the possible combinations between the factors $a, c, d, e$ is too large, we randomly sample $400$ points.

**f) CNN RWC GT:** We train with all the synthetic pieces in the generated dataset described in Section 4.1b, synthesized with the score perfectly aligned with the rendition. In this case, because there are less factors to vary, we randomly sample $50$ points.

#### *4.3.2 Score-constrained NMF*

We compare the proposed approaches with an NMF timbre-informed system based on the multi-source filter model [4, 6] which includes timbre models trained on the RWC dataset. Because we deal with a score-constrained scenario and for a fair comparison, the gains of the NMF are restricted to the notes in the score, without taking into account the time when the notes are played. Thus, each row of the gains matrix corresponding to a note is set to $1$ if a given instrument plays a note from the score. The other values in the gains matrix are set to $0$ and do not change during computation, while the values set to $1$ evolve according to the energy distributed between the instruments. The NMF parameters are kept fixed as in [4]: $50$ iterations for the NMF, beta-divergence distortion $\beta = 1.3$, STFT length of window of $96ms$, and a hop size of $11ms$.

### 4.4 Results and discussion

We present the results of the evaluated approaches in Section 4.3.1 and Section 4.3.2 on the Bach10 and Sibelius datasets. The separated audio files and the computed measures for each file and instrument are made available through Zenodo [5] .

We verify that when the training and test datasets coincide, the CNN approach achieves the best performance. We denote this case as "Oracle".

#### *4.4.1 Separation results with real recordings: Bach10*

We evaluate the considered approaches on the Bach10 dataset. Overall results are presented in Figure 2, while instrument specific ones are provided in Figure 3.

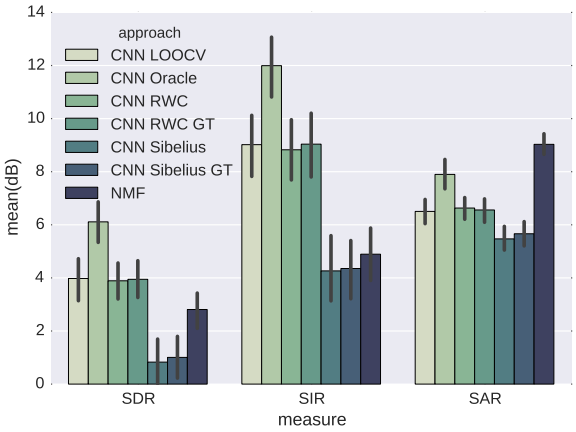


Figure 2. Results in terms of SDR, SIR, SAR for Bach10 dataset and the considered approaches: CNN and NMF [6]

The LOOCV approach is trained with examples from the same dataset, having the same timbre and style, thus achieves $\approx 4dB$ in SDR. This illustrates the fact that training the neural network on similar pieces, played by the same musicians in the same recording conditions is beneficial for the system.

The approach CNN RWC, which involves synthesizing the original score with samples comprising a variety of instrument timbres and dynamics, yields as good results as the LOOCV approach $\approx 4dB$ SDR. On the other hand, if the training set comprises less samples and less variations in timbre and dynamics, the we have considerable lower results, as in the CNN Sibelius approach which has $\approx 1dB$ SDR. In fact, the CNN RWC approach has higher SIR than CNN Sibelius ($9dB$ compared with $4dB$). Thus, learning to separate more diverse examples reduces the interference.

Synthesizing the score perfectly aligned with the rendition does not improve the results for the considered approaches: CNN RWC GT and CNN Sibelius GT. For this particular CNN architecture and the modeled time context ($300ms$), synthesizing the original score with circular shifting to compensate for local timing variation achieves results as good as the ideal case when synthesized a perfectly aligned score. This is encouraging considering that a perfectly aligned score is difficult to obtain. Furthermore, a score-following system introduces additional latency.

[3] https://github.com/MTG/DeepConvSep

[4] http://lasagne.readthedocs.io/en/latest/Lasagne and http://deeplearning.net/software/theano/Theano

[5] http://zenodo.org/record/344499#.WLbagSMrIy4

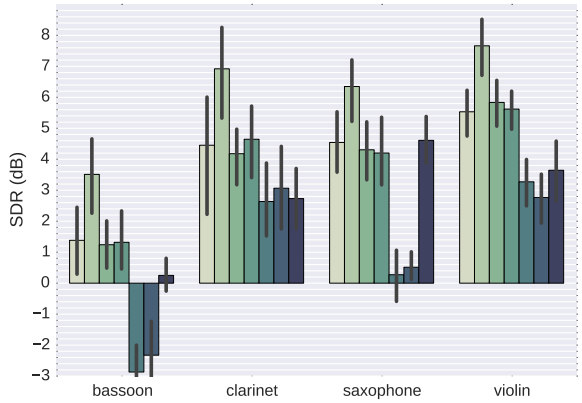


Figure 3. Results for each instrument in terms of SDR for Bach10 dataset and the considered approaches: CNN and NMF [6]

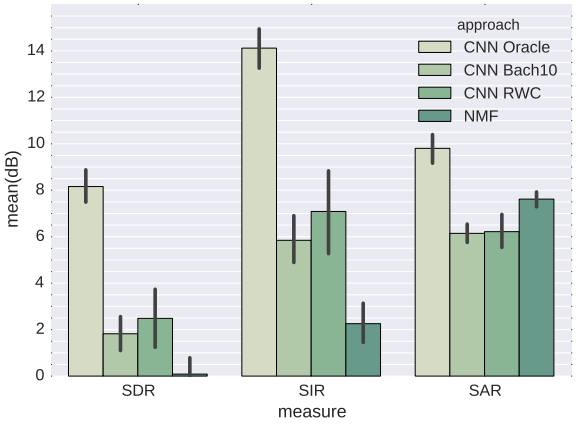


Figure 4. Results in terms of SDR, SIR, SAR for Sibelius dataset and the considered approaches: CNN and NMF [6]

The score-constrained NMF separation which uses timbre models trained on the RWC dataset, has lower SDR than proposed approach CNN RWC. The NMF system has higher SAR, less artifacts, at the expense of having lower SIR, hence more interference between the instruments.

As seen in Figure 3, the separation for all instruments benefits from having examples with more diverse timbre or dynamics in the dataset, as CNN RWC has higher SDR than the CNN Sibelius across all instruments. The results for bassoon are lower across all approaches. This is in line with the results yielded by other state of the art approaches evaluated on this dataset [6,12] and can be due to the lower register of this instrument and the poor resolution of STFT for lower frequencies.

#### *4.4.2 Separation results with synthesized recordings: Sibelius*

We now evaluate the considered approaches on the synthesized Sibelius dataset. Because "GT" approaches do not differ from their baselines, we decide not to include them here. The overall results are presented in Figure 4.

When tested on Sibelius dataset which has different tempos, timbres, dynamics than the training Bach10 dataset, the approach CNN Bach10 decreases in SDR with $\approx 4dB$. In fact, all data driven approaches have lower performance on unseen data. This raises questions on the validity of cross-fold evaluation methodology of source separation using small datasets.

As seen in Figure 4, the CNN RWC and NMF yield lower results on Sibelius synthetic dataset than on the real-life renditions of Bach10. In this case, the CNN RWC is $2.5dB$ higher in SDR than the score-constrained NMF. This is in line with the results obtained in Section 4.4.1. Thus, training this CNN with synthetic data of different timbre, dynamics and circular shifting, has better performance than NMF method, being less computationally intensive.

The results for the Oracle in Section 4.4.2 (Figure 2) are higher than Section 4.4.1 (Figure 4), because the synthesized dataset Sibelius lacks the diversity of dynamics and local tempo variations present in real-life performances of Bach10, which are more difficult to model.

We assess the computation time of the proposed framework, implemented in Python, in comparison with the NMF framework [4, 6], implemented in Matlab, on a 2013 MacBook Pro with 2.5Gz Intel Core I5 and 16Gb RAM. Separating with CNN took on average 0.76 of the length of the audio, while the NMF framework took 4.6 of the length of the audio.

## 5. OUTLOOK

We proposed a method to generate training data for timbre-informed source separation methods using neural networks, in the context of classical music. We departed from a set of scores and we synthesized new renditions by varying tempo, timbre, dynamics and local time variations.

We tested two synthesis approaches on real-life performances and on synthesized pieces of Bach chorales. We showed that evaluating in a cross-validation manner on a small dataset yields results that can not be generalized on different renditions which depart from the same score.

We underline the importance of timbre and dynamics in generating training data. Correspondingly, the approach having more varied timbre and dynamics achieved higher performance, surpassing a score-constrained NMF method [4, 6]. The proposed approach is similar to data augmentation and makes the model more robust to variations which one can expect in real cases. Thus, having a more diverse training set avoids overfitting.

As future work, we have to determine the optimum number of random samples to be used for training, as a trade-off between performance and training time. Then, we plan on testing the current framework on more complex mixtures, as orchestral music. Furthermore, the current method can be extended to a score-informed scenario by having the score as input to the network.

**Acknowledgments**

The TITANX used for this research was donated by the

NVIDIA Corporation. This work is partially supported by the Spanish Ministry of Economy and Competitiveness under CASAS project (TIN2015-70816-R). We thank Agustin Martorell for his help with Sibelius and Pritish Chandna for his useful feedback.

## 6. REFERENCES

[1] E. Vincent, C. Févotte, R. Gribonval, L. Benaroya, X. Rodet, A. Röbel, E. Le Carpentier, and F. Bimbot, "A tentative typology of audio source separation tasks," in *4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA)*, 2003, pp. 715–720.

[2] J. Janer, E. Gómez, A. Martorell, M. Miron, and B. de Wit, "Immersive orchestras: audio processing for orchestral music vr content," in *Games and Virtual Worlds for Serious Applications (VS-Games), 2016 8th International Conference on*. IEEE, 2016, pp. 1–2.

[3] E. Gómez, M. Grachten, A. Hanjalic, J. Janer, S. Jordà, C. F. Julià, C. C. S. Liem, A. Martorell, M. Schedl, and G. Widmer, "Phenicx: Performances as highly enriched and interactive concert experiences," in *SMC*, 2013.

[4] M. Miron, J. Carabias-Orti, J. Bosch, E. Gómez, and J. Janer, "Score-informed source separation for multichannel orchestral recordings," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.

[5] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[6] J. J. Carabias-Orti, T. Virtanen, P. Vera-Candeas, N. Ruiz-Reyes, and F. J. Canadas-Quesada, "Musical Instrument Sound Multi-Excitation Model for Non-Negative Spectrogram Factorization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1144–1158, Oct. 2011.

[7] P.-S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson, "Singing-voice separation from monaural recordings using robust principal component analysis," in *ICASSP*. IEEE, mar 2012, pp. 57–60.

[8] E. Grais, M. Sen, and H. Erdogan, "Deep neural networks for single channel source separation," in *ICASSP*. IEEE, may 2014, pp. 3734–3738.

[9] A. Simpson, G. Roma, and M. Plumbley, "Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2015, pp. 429–436.

[10] S. Uhlich, F. Giron, and Y. Mitsufuji, "Deep neural network based instrument extraction from music," in *ICASSP*. IEEE, 2015, pp. 2135–2139.

[11] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," *International Conference on Latent Variable Analysis and Signal Separation*, 2017.

[12] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–12, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5887382

[13] J. Fritsch and M. Plumbley, "Score informed audio source separation using constrained non-negative matrix factorization and score synthesis," *ICASSP*, pp. 888–891, 2013.

[14] G. Widmer and W. Goebl, "Computational models of expressive music performance: The state of the art," *Journal of New Music Research*, vol. 33, no. 3, pp. 203–216, 2004.

[15] S. Ewert and M. Müller, "Score-Informed Voice Separation For Piano Recordings." *12th International Society for Music Information Retrieval Conference*, no. 12, pp. 245–250, 2011.

[16] J. Ganseman, P. Scheunders, G. Mysore, and J. Abel, "Source separation by score synthesis." in *International Computer Music Conference*, 2010.

[17] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *16th International Society for Music Information Retrieval Conference*, 2015.

[18] J. Salamon and J. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[19] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[20] M. Goto, "Development of the RWC music database," in *ICA*, 2004, pp. 553–556.

[21] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1462–1469, jul 2006.

[22] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and Objective Quality Assessment of Audio Source Separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2046–2057, Sep. 2011.