Yorick LALY

2A SICOM
University year 2017-2018

Detection of sounds with deep learning

Tutor: Benjamin RICAUD, Benjamin.ricaud@epfl.ch

Confidentiality: None

Table of Contents

Glossary

CNN: Convolutional Neural Network
DANN: Deep Autoencoder Neural Network
STFT: Short Time Fourier Transform
To feed the network: giving a spectrogram in the input of the network
U-net: type of deep autoencoder
SGD: Stochastic Gradient Descent
Backpropagation: method to calculate the gradient for the update of the weights for the SGD and the neural network.

List of figures

Introduction

The detection of sound is a method to detect a particular sound, the sound of a guitar in a song or the cry of a baby in a baby phone. The detection of sound is important in music for creating new songs from old ones by replacing an instrument by another one. But also during live concert, with the separation of source, the song can be restore without the shot and scream from the crowd and the voice of the singer can be restored for him so he can hear his/her in the middle of the other instrument.

Nowadays, with the machine learning, computers are able to recognize objects in a picture or the handwriting. They can even beat great champion of chess or go game such as Alpha Go. Machine learning is used for autonomous car. Thanks to the machine learning, computers are able to learn different complex task really fast and with great results. Many researches on the machine learning have led to the machine learning, not only for art and music purpose but also for improving the quality of communication in the world with our smartphones or application such as Discord, TeamSpeak …

The goal of this report is to show how to detect the different instruments with the machine learning and separate them in the most efficient way and to present the results obtained with the neural network we used. To do that, this report shows the neural network we created and how we trained in order to have the best results. Then, we present the results that we had with this network.

This report is composed of 3 parts. The first one is about the neural network we chose and why we chose this type of network. In the second part, we talk about the training phase of our neural network. In the last part, we test our network and discuss the results we obtained. A conclusion of our work is present at the end of this report.

I – Deep Autoencoder Neural Network (DANN)

The convolutional neural network (CNN) is basically use for image detection, for example the CIFAR-10 and CIFAR-100 datasets which are datasets with respectively 10 and 100 different categories of objects or animals (cats, ships, ...) or for handwriting detection (MINST dataset). In order to separate the sources in an audio segment, we use a CNN which can give the same results as a feedforward neural network but with far less parameters [1]. We chose to create a CNN with thinner layer to compress more the information and we add skip connection layers.

a) Deep Autoencoder Neural Network (DANN)

To separate the different instruments in a song, we created our own neural network: a Deep Autoencoder Neural Network (DANN).

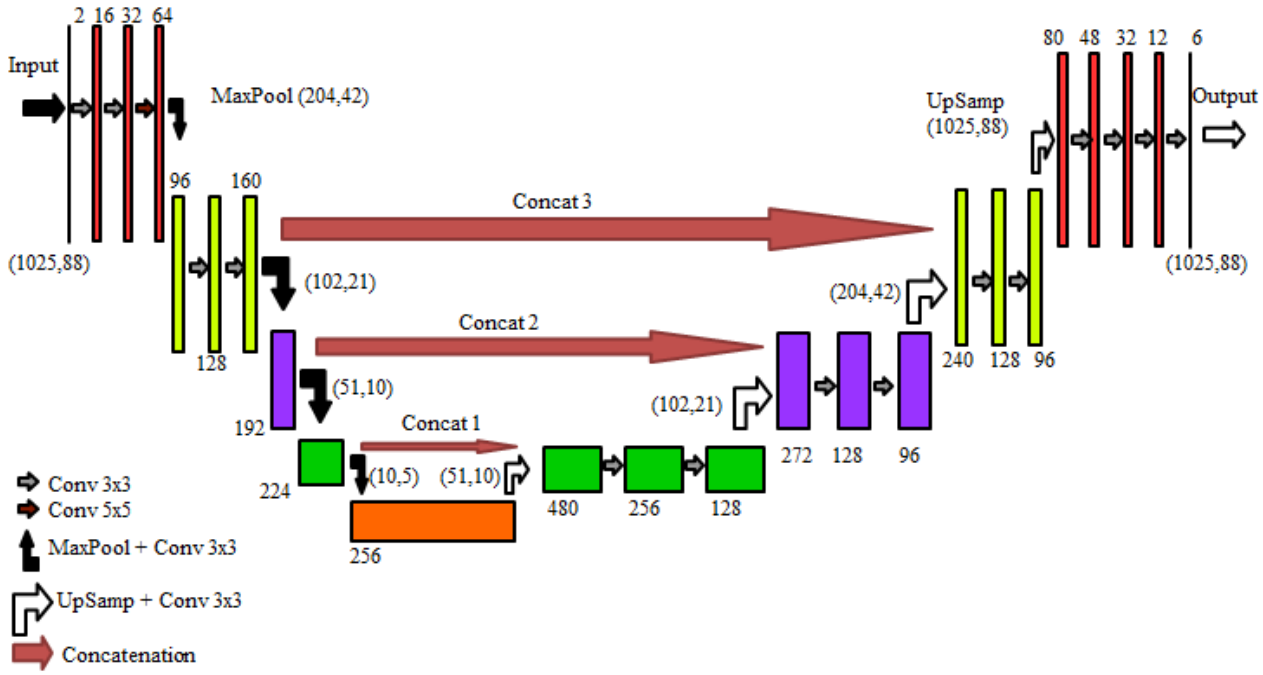
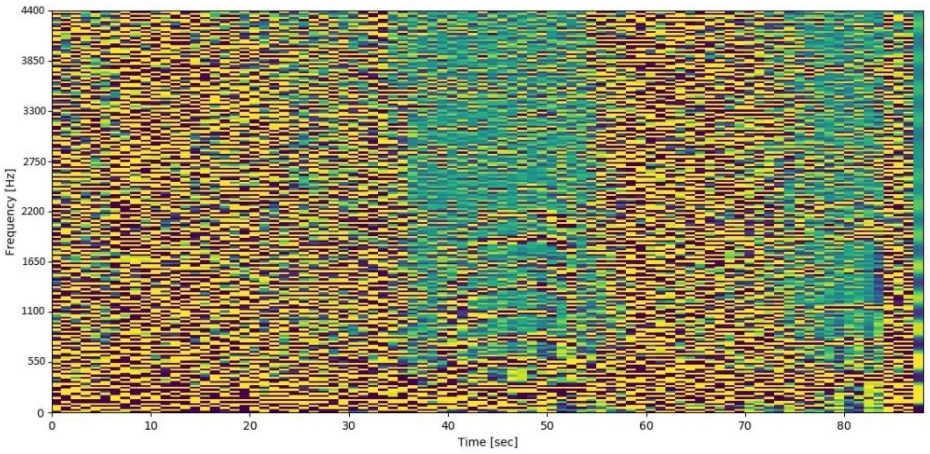
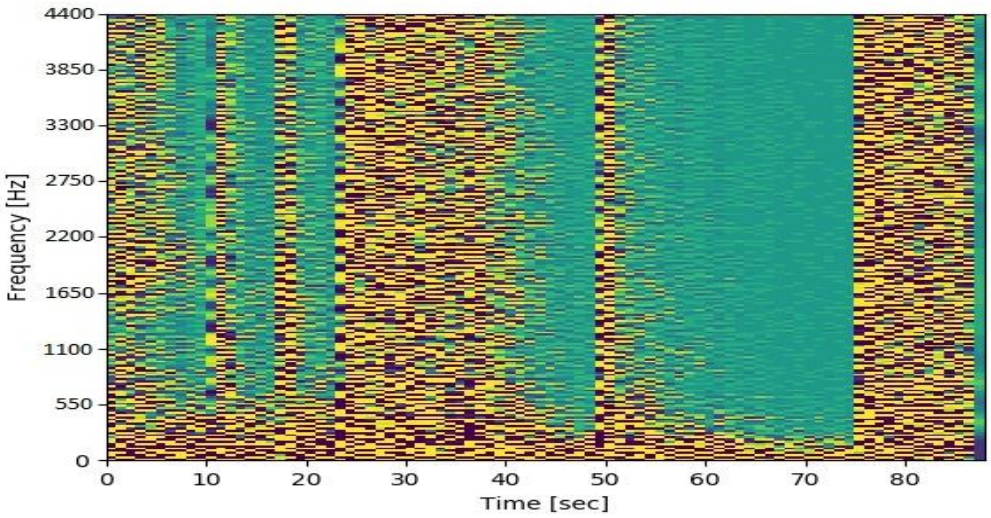


Figure 1: DANN

DANN is an asymmetrical U-net with skip connection layers. The inputs of the network are the real part and imaginary part of a spectrogram that we obtain by computing the Short Time Fourier Transform (STFT) of a song. The result of this STFT is the evolution of the frequencies in function of the time. The output of DANN is the different masks for each of the instruments we are trying to separate. Then, these masks are multiplied by the inputs. These masks will suppress the noise and conserve the instruments that they are trying to separate.



(a) Input spectrogram



(b) Output spectrogram (the input spectrogram has been multiplied by the mask)

Figure 2: (a) Input real part spectrogram before being fed to the network, (b) output real part spectrogram, the input spectrogram has been multiplied by the mask.

The input spectrogram in the figure (a) has been fed to the network. Here the output of the network is a mask to detect the drums. The figure (b) is the output spectrogram after masking. The mask has suppressed the noise from the other instruments which reveals the particularity of the drums in the frequency/time plan.

b) U-net

The U-net is a type of neural network. Its goal is to recreate the signal without the noise by down sampling and up sampling the input spectrogram.

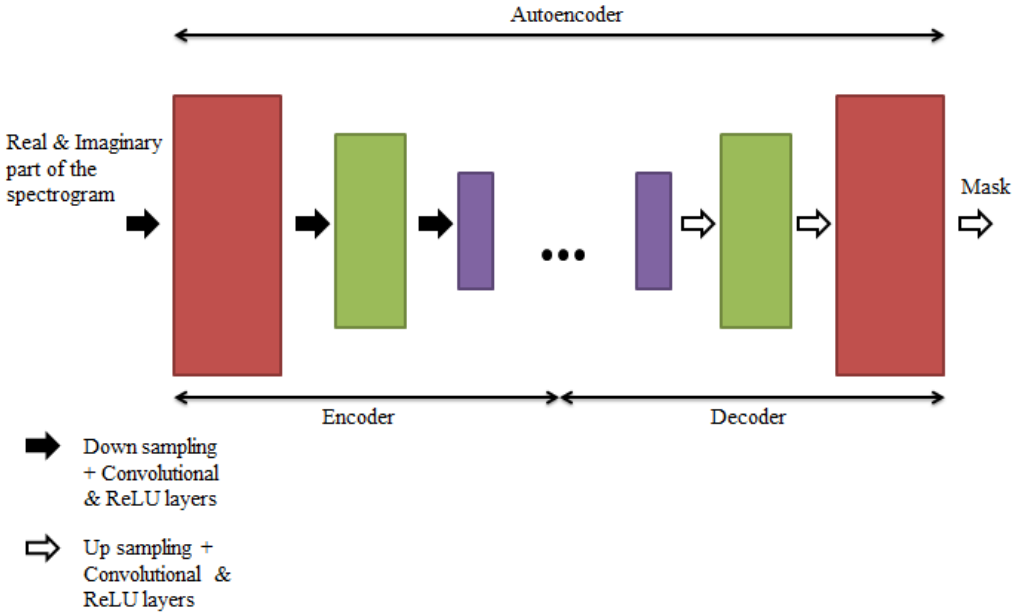


Figure 3: Scheme of a deep Autoencoder

The encoder part has to suppress the noise by down sampling the input spectrogram. To down sample the spectrogram, we use the MaxPool function which takes the highest value in a matrix:

$$MaxPool\left(\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\right) = 4 \tag{1}$$

By mixing the MaxPool function with the convolutional layers, the network will learn to increase the values corresponding to the instruments we want to separate and decrease the noise.

The decoder part has to recreate the input spectrogram with the down sampled spectrogram. The function that we used to up sample is UpSample:

$$UpSample\left(\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\right) = \begin{pmatrix}1{,}00 & 1{,}33 & 1{,}67 & 2{,}00\\ 1{,}67 & 2{,}00 & 2{,}33 & 2{,}67\\ 2{,}33 & 2{,}67 & 3{,}00 & 3{,}33\\ 3{,}00 & 3{,}33 & 3{,}67 & 4{,}00\end{pmatrix} \tag{2}$$

c) Skip connection layer

While we are down sampling, we are losing information on the noise but also information on the instrument that we are trying to separate. To avoid this problem we use skip connection layers, [2]. To do that, we take the filters from a layer of the encoder part and we inject it in the layer of the decoder part. It allows useful information recovering on the instrument we are trying to separate.

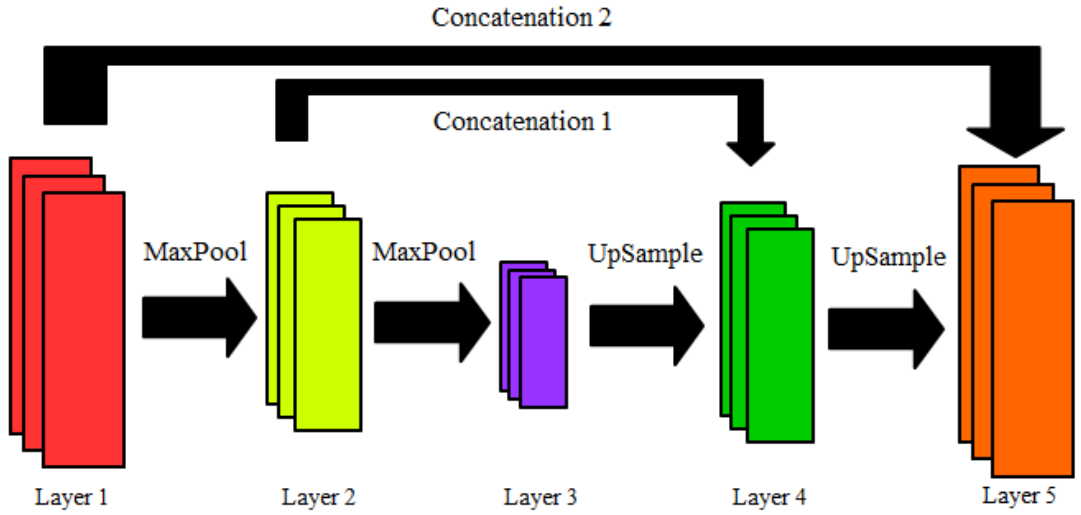


Figure 4: Neural network with skip connection layers

In this example, layer 1 is concatenated with layer 5 and layer 2 is concatenated with layer 4. We can observe the advantage of this technique with the way the neural network is learning.

We compute the graph of the evolution of the loss during the learning phase for the equivalent neural network with one skip connection layer, called Concat 1, using our own neural network:

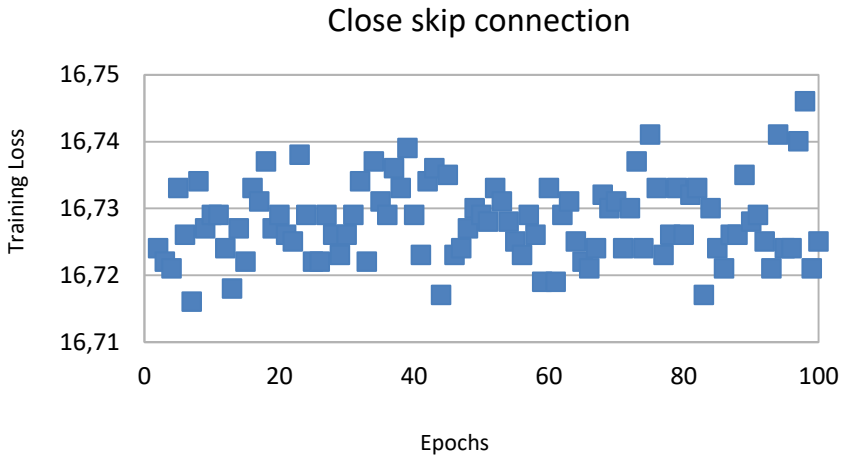


Figure 5: Evolution of the training loss without skip connection layer

With a close skip connection layer, the neural network can't learn properly because the amount of information lost is too high and the output mask is too different from the ground truth because the weights are randomly initialized. So the equivalent DANN without skip connection layer won't be able to learn.

We compared different skip connection layers to see how far each layers have to be and how many. The comparison is made by the observation the evolution of the loss of the test dataset while the network is learning the train dataset.

The loss is calculating by computing the L1Loss with the following equation:

$$Loss = \sum_{i,j}^{m,n} \left| M_{l_{i,j}} * I_{i,j} - N_{l_{i,j}} \right| \quad (3)$$

With:

- I: the input spectrogram of size (m,n)
- M_i: the output mask of the ith instrument of size (m,n)
- N_i: the spectrogram of the ith instrument we want to separate of size (m,n)

- Comparison between Concat 2 and Concat 3:

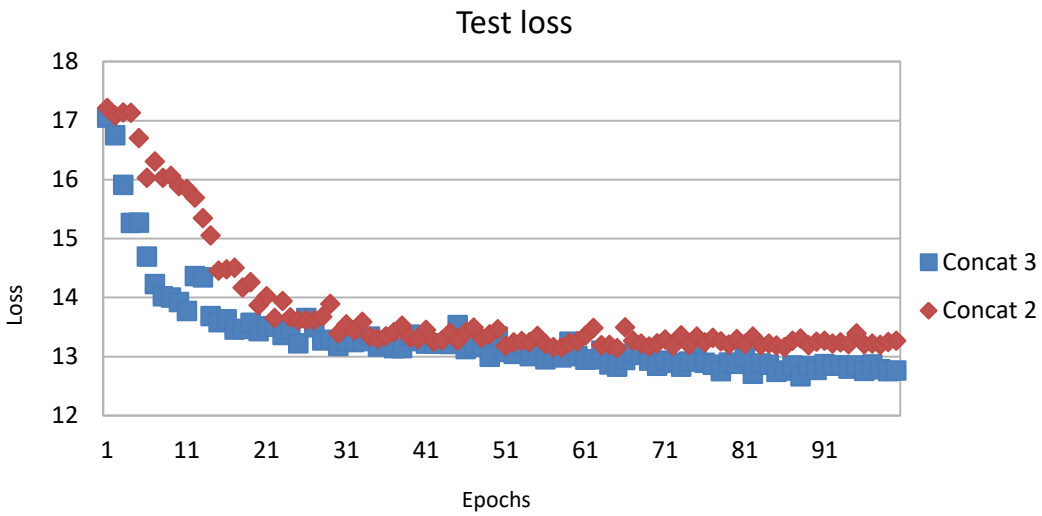


Figure 6: Evolution of the test loss for a network with only the skip connection layer Concat 2 and a network with only Concat 3

A network with the skip connection layer Concat 2 will learn faster but it converges to a higher value of the loss than the skip connection layer Concat 3.

- Comparison between Concat 3 and Concat (2 + 3):

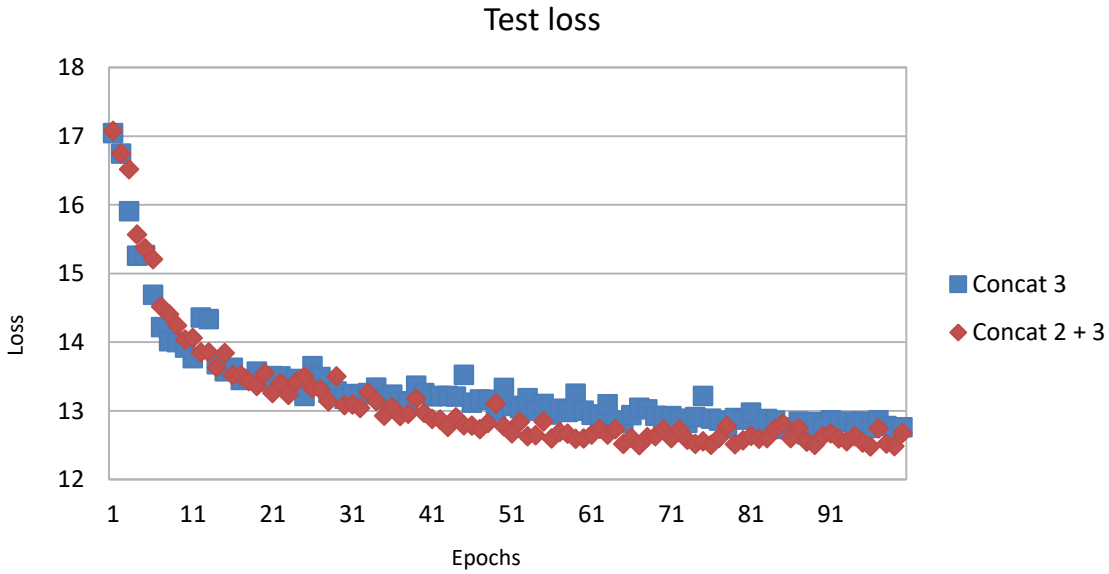


Figure 7: Evolution of the test loss for a network with Concat 3 only and a network with Concat 2 and Concat 3

The network with Concat 2 and Concat 3 is slower to learn but it converges to a better value for the loss than the network with Concat 3.

With two skip connection layers, DANN has a better precision because the two layers bring more information on the instrument than one layer.

We don’t compute a skip connection layer with further layers because the layer that will be concatenate has not been down sampled and is very close to the input spectrogram and it will add all the instruments we want to suppress from the song.

So we chooses to have 3 skip connection layers in order to be the more precise on the spectrogram since the human ear is able to hear 10 Hz variation. However DANN will be slower in its learning phase. We will apply different audio treatment on the songs to make the learning faster.

II – Training

a) Dataset

The dataset we are using is the musdb 2018 dataset [3]. This dataset is composed of 100 training songs and 50 testing songs. All the songs are free of charge. The training songs are rock'n'roll, pop rock and metal songs with different instruments like vocals, guitars, drums, bass, ... The testing songs are from different type of music. This data allows us to separate from those songs the vocals, the drums and the bass by providing for each songs, the song with only the vocals, only the drums, only the bass and only the other instruments. The songs are sampled at 44,1 kHz and are stereo song.

b) Pre and Post treatment

Since the songs are stereo, we transform them into mono song by calculating the average of the two channels. The songs are sampling at 44,1 kHz but we resample them at 8,8 kHz in order to accelerate the learning and the training phase. It doesn't affect a lot the quality of the song even if everything above 4,4 kHz is filtered. We also apply a gain on the song in order to improve the calculation of the loss. To obtain the spectrogram, we calculate the STFT of the song with a Hamming window 2048 points along with an overlap of 1024 points and a Fourier transform of 2048 points. The output of the function is the complex version of the spectrogram of size 1025 in frequency, we compute to separate the real part and the imaginary part that both are the inputs of the network. We cut the spectrogram of the song in band of 88 points in time which corresponds to 2 seconds of the song. From the train dataset, we have 2191 spectrograms of size 1025 in frequency and 88 in time. We initialize random batch of size 16, it means that among the 2191 spectrograms we randomly select 16 spectrograms and the 16 spectrograms are feed to the network in the same time. Then program will compute the average loss of the 16 spectrograms.

The outputs of the network are the masks for the real and imaginary parts of the vocals, the drums and the bass. Then we multiply these masks with the input spectrogram. We compute to have the complex spectrogram of the 3 instruments. After processing all the spectrogram bands in the network, we stick them together in the same order we cut them and we calculate the inverse STFT. The songs are then resample to 44,1 kHz and we apply a different gain for each instrument to increase the volume of the songs.

c) Training phase and overfitting

The training phase is the phase where the network learns to recognize and separate the instruments from each other. We trained the network using backpropagation with the Stochastic Gradient Descent (SGD) with the parameters: learning rate = 0,000 1 and momentum = 0,9. The loss function is the L1Loss with the equation in (3).

The goal of the backpropagation is to minimize the loss value. The closer the loss value is to 0, better will be the separation of the instruments for the songs from the train dataset. However the

separation for a song out of the train dataset won't be very good because the network didn't learn to separate the instruments from this song. We need to avoid this problem called overfitting.

We process to train our network and to calculate the loss for each epoch from the train dataset but also from the test dataset in order to see at which point we have to stop the training phase to avoid the overfitting.

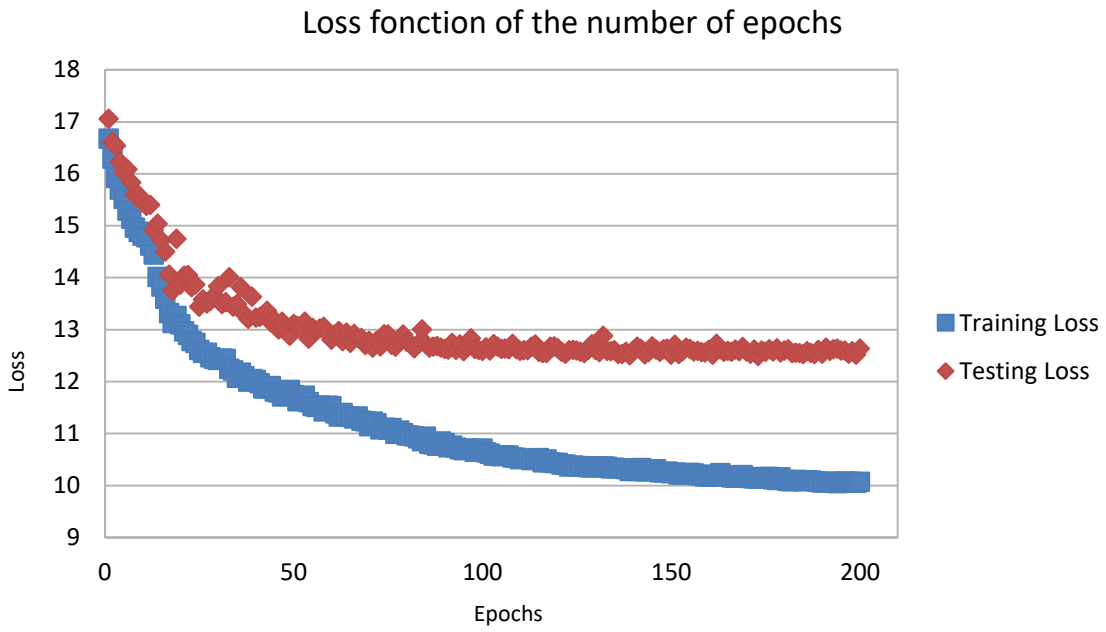


Figure 8: Evolution of the training loss and the testing loss during the training phase.

In this figure, we can see that after nearly 100 epochs the testing loss stop decreasing and remain constant while the training is still decreasing. It means that at this point, the network can't learn more from the train dataset to improve its separation with the test dataset but it keeps learning to have a better separation for the songs from the train dataset.

So we stopped the training phase after 100 epochs for our network and start the testing phase. The steps are the same for both the phases but we didn't compute a batch for the test dataset since we want to compute the loss for each songs from the test dataset and we want to be able to recreate them in order to listen to them to better understand why the separation is good or not with one song than with another.

To do that, we have to test our network with the 50 songs from the test dataset and define different signal to ratios in order to judge the quality of our work.

III – Test & Results

a) Equations

To evaluate the quality of the separation, we use the different ratios according to: signal to interferences ratio (SIR), signal to distortion ratio (SDR) and signal to artifacts ratio (SAR) [4]. The interferences are the instruments we want to suppress, for example, if we want to separate the vocals from the rest of the instruments, the interferences are the bass, the drums and the other instruments. The artifacts are the noise made by the neural network but also by the pre and post treatment.

To calculate those ratios, we need to make a few hypotheses:

- the instruments that we are separating are mutually orthogonal
- we don't have any other noises than those from the interferences and artifacts

We define the jth output instrument as:

$$s_{output_j} = s_{target} + e_{interf} + e_{artif} \quad (4)$$

With:

- s_{target}: a distorted version of the song with only the jth instrument that we obtain by calculating the projection of the jth output instrument on the song with only the jth instrument.
- e_{interf}: noise from the instruments we want to suppress.
- e_{artif}: noise from the neural network and the pre and post treatment.

- Signal to Distortions Ratio:

$$SDR = 10log_{10}\left(\frac{\|s_{target}\|^2}{\|e_{interf}+e_{artif}\|^2}\right) \quad (5)$$

The SDR shows the impact of both the interferences and the artifacts on the output song.

- Signal to Interferences Ratio:

$$SIR = 10log_{10}\left(\frac{\|s_{target}\|^2}{\|e_{interf}\|^2}\right) \quad (6)$$

Higher the SIR is, better is the separation between the instrument we want to separate and the other instruments.

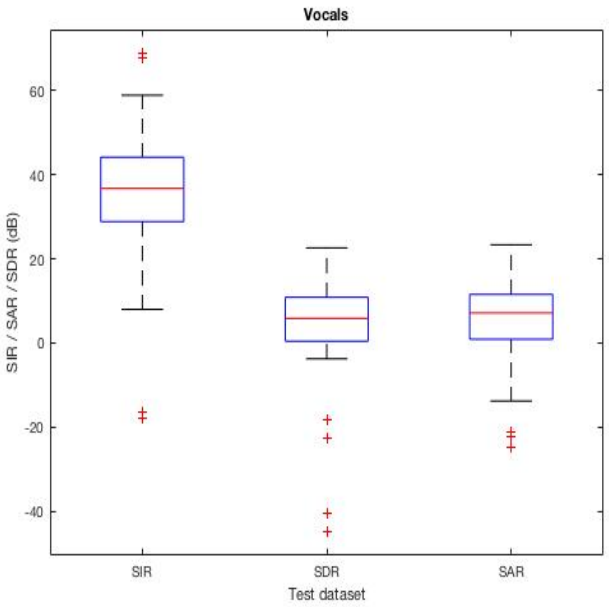
- Signal to Artefacts Ratios:

$$SAR = 10log_{10}\left(\frac{\|s_{target}+e_{interf}\|^2}{\|e_{artif}\|^2}\right) \quad (7)$$

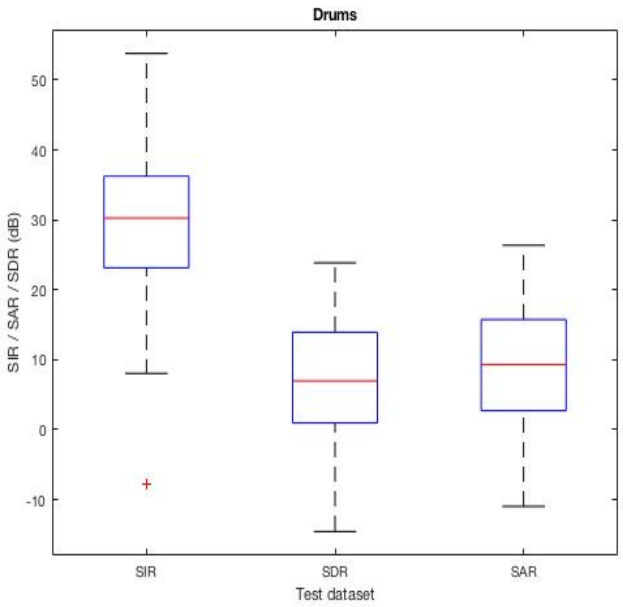
The SAR shows how the network and the treatment affect the output song.

b) Results

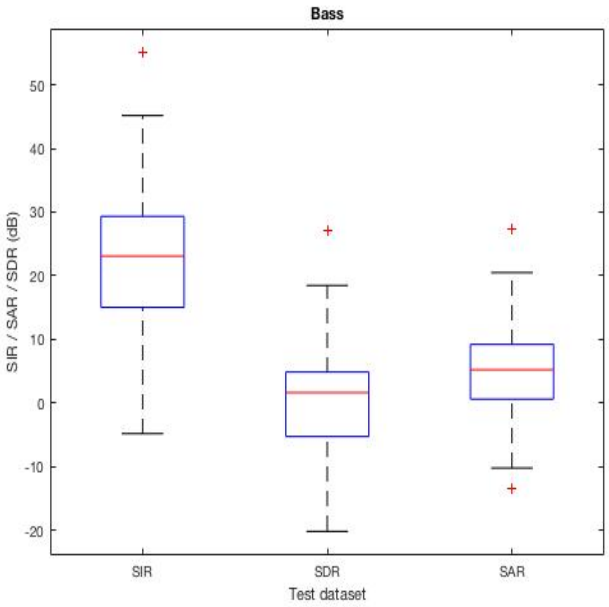
We feed the network with the songs from the test dataset and calculate the 3 ratios for each song and we obtain these results:



(a) SIR / SDR / SAR for the vocals



(b) SIR / SDR / SAR for the drums



(c) SIR / SDR / SAR for the bass

Figure 9: (a) SIR / SDR / SAR for the vocals (b) SIR / SDR / SAR for the drums (c) SIR / SDR / SAR for the bass for the 50 songs from the test dataset.

For the 3 instruments, the results are similar. On average, the SIR is high which means that the separation of each instrument from the others is good and is higher than the SIR from. However, the SDR and the SAR are low which can be explained by the fact that the bass and the drums are often background instruments so it is harder to separate them.

The SAR values are low because of the neural network but also and mainly because of the pre and post treatment. By down sampling to 8,8 kHz and up sampling to 44,1 kHz, we lose in quality but the training is faster since we have 5 times less samples.

These ratios allow us to judge the quality of our network but another test allows us to understand why we do have good or bad results: the listening test.

We have good results for songs which are the same kind of music as the songs from the train dataset (rock'n'roll, pop rock, metal). But the results is even better if the number of instruments in the song is low because it means that there is less noise that interfere with the instrument and less noise to suppress.

The bad results come from songs in which the instrument is used as a background instrument in the middle of lots of other instruments, in this case, the suppress of the noise by using a mask has its limits. Bad results also come from electro songs because the network has not been trained to separate instruments from this type of music. Also, electro songs are songs generated by computer, the fact is that there is no conventional instruments in it as a guitar, a bass …

During the listening, we can also hear some reverberation. The reverberation could be caused by the up sampling to 44,1 kHz but also by the separation of the complex spectrogram in a real value spectrogram and imaginary value spectrogram. After being feed in the network, the phase of the spectrogram could have been modified that may is why we can hear some reverberation.

Conclusion

The goal of this report was to show how to detect and separate instruments in a song using deep learning. The first part shows the neural network we used and the importance of using a deep autoencoder and skip connection layer in order to improve the speed of learning of the network but also the quality of the output songs. The second part explains how to increase the speed of the calculation during the training phase by down sampling the songs without impacting too much the quality of the songs. The second part also shows the risk of overfitting and how to avoid this problem. In the last part, the results show that the detection and separation of the instruments is good but there are distortions and reverberation in the songs due to the treatment we realized to increase the learning phase speed. This part shows the limit of the network since it can't separate the instruments if the songs are not the same type of music that the ones using during the training phase and it is harder for the network to separate background instruments. The network is also not able to separate the same type of instruments in a song, for example: separate a bass from another.

However, all those limits can be broken; we can still work on the improvement of the detection and separation of sound by working with a bigger and more diversified dataset with more songs from different types of music in order to improve the separation for all songs but also add more instruments in the separation. Another way to improve the separation is to preselect the intervals of variation in frequency of the different instruments, for example the violin is at a higher frequency than the bass. An option to add to detection of sound would be to separate the same type of instruments, to separate a guitar from another guitar that both would play different melody.

References:

[1] Emad M. Grais and Mark D. Plumbley (13-10-2017) "*SINGLE CHANNEL AUDIO SOURCE SEPARATION USING CONVOLUTIONAL DENOISING AUTOENCODERS*"
[Online] https://arxiv.org/abs/1703.08019
[2] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba (30-07-2018) "*The Sound of Pixels"*
[Online] http://sound-of-pixels.csail.mit.edu/
[3] Rafii, Zafar and Liutkus, Antoine and Fabian-Robert Stöter and Mimilakis, Stylianos Ioannis and Bittner, Rachel (12-2017) "*The MUSDB18 corpus for music separation*"
[Online] https://sigsep.github.io/datasets/musdb.html#tools
[4] Emmanuel Vincent, Rémi Gribonval, Cédric Févotte. Performance measurement in blind audio source separation. IEEE Transactions on Audio, Speech and Language Processing, Institute of Electrical and Electronics Engineers, 2006, 14 (4), pp.1462–1469. <inria-00544230>

Abstract

Thanks to machine learning, computers are able to recognize objects or handwriting in a picture but also recognize instruments in a song. This report presents how with deep learning and neural network, computers can detect and separate instruments in a song in an effective way without spoiling too much the quality of the songs.

This document shows that using convolutional neural network, deep autoencoder and skip layer connection we can separate the instruments of a song from its spectrogram by computing its Short Time Fourier Transform and feeding the network. The output of the network is a mask that is applied on the input spectrogram and suppresses the other instruments. Because of the down sampling of the song to accelerate the training phase, the quality of the songs is lower but the separation works, even if it works better for songs with few instruments.

Résumé

Grâce à l'apprentissage automatique, les ordinateurs sont capables de reconnaître des objets ou des écritures manuscrites dans une image, mais aussi de reconnaître des instruments dans une chanson. Ce rapport présente comment, avec l'apprentissage automatiques et les réseaux de neurones, les ordinateurs peuvent détecter et séparer les instruments d'une chanson de manière efficace sans trop gâcher la qualité des chansons.

Ce document montre qu'en utilisant un réseau de neurones composés de couches de convolution, un auto-encodeur et une connexion par saut de couche, nous pouvons séparer les instruments d'une chanson de son spectrogramme en calculant sa transformée de Fourier à court terme et en alimentant le réseau. La sortie du réseau est un masque qui est appliqué sur le spectrogramme d'entrée et supprime les autres instruments. En raison du sous-échantillonnage du morceau pour accélérer la phase d'entraînement, la qualité des morceaux est moindre, mais la séparation fonctionne, même si elle fonctionne mieux avec des morceaux contenant peu d'instruments.