Master ATIAM, IRCAM

Rapport de stage

R&D Perception & Sound Design

Transform analysis and music style transfer with deep neural networks

Author
Evgenia Eremeeva

Supervisors
Philippe Esling
Patrick Susini

20 August 2017

Année universitaire 2016-2017

Contents

I State of the art **3**

1 Introduction **3**
1.1 Motivations 3
1.2 Objectives and research problems 4

2 Deep Neural Networks **5**
2.1 Principles of machine learning 5
2.2 Supervised and unsupervised learning 7
2.3 Convolutional Neural Networks 8

3 Style transfer **10**
3.1 Concept of style transfer 10
3.2 Application to audio related tasks 13

4 Spectrotemporal analysis based on auditory cortical physiology **14**
4.1 STRF-representation 14

5 Dimensionality reduction methods **17**
5.1 From PCA to Multilinear-PCA 17

II Experimentations & results **20**

6 Preprocessing **20**
6.1 Spectral transform 20
6.2 Dataset 20
6.3 Analysis of audio transforms 21
6.4 Reduction of dimensionality 21

7 Models **21**

8 Music style transfer **22**

9 Results **23**

III Conclusion **24**

10 Discussion **24**
10.1 Advantages & shortcomings 24
10.2 Perspectives & future work 24

Abstract

Over the last years, deep networks have obtained impressive results for image processing. In particular, the Convolutional Neural Networks (CNN) [18] are currently the state-of-the-art in multiple learning tasks. However, it is somewhat not logical to apply the same methods straightforwardly to audio processing, as the nature of audio data is different from image data. Nevertheless, most models used in the audio processing community are direct applications of the findings in image processing. Indeed, the main idea behind the success of CNN is to rely on small convolutional kernels to achieve deeper compositionality in the network transforms. This choice allows to harness the stationarity property of an image, and also the fact that both dimensions of an image are equivalent.
However, most researches in Music Information Retrieval (MIR) rely on different types of spectral transforms, where the time and frequency dimensions are not equivalent and the the frequency axis provides less stationarity than the temporal one.
Hence, this internship aims to analyze these different spectral representations and to introduce the use of the *Spectro-Temporal Receptive Fields (STRF)* as an input to deep neural networks. The main obstacle towards this goal lies in the high-dimensionality of the STRF. Therefore, to solve this problem, we evaluate different pre-processing and decimation methods to obtain an efficient and compact representation of the STRF. The second stage of this research will be to apply these approaches to the task of *style transfer*. In the image processing field, this task has been defined as the ability to retain the *content* of an image, while applying the *artistic style* of another. Here, we discuss how this concept could be applied in the music processing field, while underlying the fact that the notion of *musical style* is largely more complex than the notion of *texture* used in image style transfer.

Résumé

Au cours de ces dernières années, les réseaux profonds ont obtenu des résultats impressionnants sur le traitement d'image. En particulier, les réseaux neuronaux convolutionnels (CNN) [18] constituent actuellement l'état de l'art des tâches d'apprentissages multiples. Cependant, il n'est pas pertinent d'avoir une approche similaire pour le traitement audio, car la nature des données audio est différente de celle des données visuelles. Néanmoins, la plupart des modèles utilisés au sein la communauté de traitement audio sont des applications directes des résultats obtenus pour le traitement de l'image. En effet, l'idée principale qui fait le succès des CNN est de s'appuyer sur des petits grains convolutionnels afin d'obtenir une composition plus fine pour les transformations du réseau. Ce choix permet d'exploiter la propriété de stationnarité d'une image, et aussi le fait que les deux dimensions d'une image sont équivalentes.
Cependant, la plupart des recherches sur la récupération des informations musicales (MIR) dépendent des différents types de transformations spectrales, où les dimensions du temps et de la fréquence ne sont pas équivalentes. L'axe de la fréquence fournit moins de stationnarité que l'axe temporaire.
Par conséquent, ce stage a pour objectif d'analyser ces différentes représentations spectrales et d'introduire l'utilisation des *champs de récepteurs spectro-temporels* (STRF) en tant que données d'entrée pour les réseaux neuronaux profonds. Le principal obstacle à cet objectif réside dans la grande dimensionnalité des STRF. Par conséquent, pour résoudre ce problème, nous évaluons différentes méthodes de prétraitement et de décimation afin d'obtenir une représentation efficace et compacte des STRF. L'étape suivante sera d'appliquer ces ap-

proches à la tâche de *transfert de style*. Dans le domaine du traitement d'image, cette tâche a été définie comme la capacité à conserver le *contenu* d'une image, tout en appliquant *le style artistique* d'une autre. Ici, nous discutons de la manière dont ce concept pourrait être appliqué dans le domaine du traitement de la musique, tout en sous-tendant que la notion de *style musical* est en grande partie plus complexe que la notion de *texture* utilisée dans le transfert de style d'image.

Part I
State of the art

1 Introduction

1.1 Motivations

Improving deep learning in MIR

The field of Music Information Retrieval (MIR) aims to understand high-level features from musical signals [4]. To that aim, multiple approaches have been proposed to perform tasks such as *genre classification* [36] or *style recommendation* . Recently, deep learning has been proposed as a way to efficiently learn such hierarchies of features directly from the data [2]. Since the first use of deep learning in audio processing, the MIR field has shown a constantly growing interest towards these approaches [11]. However, despite its success in image processing, the application of deep learning to the research in MIR seems to provide lesser ground-breaking results. Here, we hypothesize that this might come from the nature of the audio input data that is fed to the learning algorithms.
Indeed, the use of deep learning and most notably convolutional networks in image processing exploits several properties of image data that are keys to their success. One important property of the visual data is its stationarity. It means that the statistics (average, variance) of the different subsets are the same. Another significant fact is an equivalence in the dimensions. In the case of images, pixels are invariant by rotating and an the spinned image will remain the same image.
In order to address these shortcomings, we focus in this work on the representations of sound that we can use for deep learning with audio data. In particular, we study the Spectro-Temporal Receptive Fields (STRF) [5, 33], which is a transform based on an analysis of the representation of sound in the human brain. This representation possesses several advantages on existing spectral transforms. Firstly, it is inspired by human perception and consequently is closer to it. Secondly, it establishes a *more stationary* structure. Finally, this representation contains more informations about the sound than all other ones.

However, a significant disadvantage of this representation is its very high dimensionality, as it computes multiple modulation types over a sound signal, leading to 4-dimensional tensors. But the capacities of learning offered by the STRF (especially the possibility of reconstructing the audio directly from this representation) encourages us to search for a method of reducing their dimensionality to use them as a basis for learning. Therefore, we will start by studying the properties of audio transforms, and try to find a way to handle the dimensionality of the STRF. Our first approach to this problem will be to study Principal Component Analysis (PCA) of this data. The other way that we will introduce in this internship is an investigation of the stationarity properties of the STRF.

Applying style transfer to music

In this internship, we will target methods able to perform musical style transfer.
The issue of style transfer in music which, in fact, can not be a *transfer of the musical style* in its general definition. Indeed, we associate a notion of musical style with parameters of voice,

instrument, intonation, structure, dynamic and etc. In the method set out in this section the task of definition of style or, more accurately, *texture*, is entrusted to artificial intelligence. By abuse of language we call it by analogy with an existing method in image processing [18] and which we would expand on domain of sound signals. It is logical to assume that the machine will need of parameters, expressed rather digitally and not intuitively. So, we should indicate the unchangeable (semantic content) components, transferring (style) components and combination process.
Some interesting attempts what can play a part of the components to recombine, have been made to answer this question, based on cognitive processes and perception [25, 41]. A method called *Neural Style Transfer* [37] based on recombining the content and style components through convolutional neural networks has been proposed successfully for image style transfer. Hence, we expect, by using deep neural networks, to be able to separate the components of *content* and *style* in music. However, as argued previously, this approach could only be successful if we find a transform fit to the peculiarities of convolutional networks.

1.2 Objectives and research problems

The objectives developed in this internship boil down to three main tasks

1. Analyzing different audio transforms for improving deep learning in MIR tasks;
2. Developing methods to allow the use of STRF in audio information retrieval with deep CNN networks, specifically:
    - Analyzing the harmonic stationarity of STR representation,
    - Reducing their dimensionality through Multilinear-PCA.
3. Using the results of the previous tasks to try to perform musical style transfer based on STRF and CNN.

Altogether, these tasks are aimed to use the STRF both for the typical tasks of recognition in the MIR field, but also to perform music generation in the style transfer problem.
We will test the efficiency of these representations on a classic task of genre classification realized by supervised machine learning using Convolutional Neural Networks.

Structure of this document

The remainder of this document is organized as follows. In the first part, we introduce the basic notions of machine learning and the different types of neural networks (Section 2), while showing their application in the MIR field. We focus on convolutional networks and give some examples of application tasks. Then, we discuss the notion of style in music and possible mathematical and algorithmic interpretation of style transfer (Section 3). In the next section, we talk about the different types of sound representation, while focusing on STRF and its advantages compared to Fourier transforms and others representations (Section 4). Finally, we talk about the problems of high-dimensionality in the data and how we could alleviate these issues (Section 5). In the second part of this report, we introduce the experimentations realized during the internship and some obtained results. We present the different types of preprocessing (Section 6), training models (Section 7) and neural texture transfer (Section 8) that were used along this work. Then, we provide a summary of our results (Section 9) and conclude this work by discussing the advantages, shortcoming and perspectives of the used methods (Section 10).

2 Deep Neural Networks

In this section, we give a brief introduction into machine learning and deep neural networks. We delineate the differences between supervised and unsupervised learning, based on the task at hand. We focus particularly on the convolutional type of neural networks as they are currently the state-of-the-art models for classification tasks.

2.1 Principles of machine learning

The objective of machine learning is to find an appropriate solution to complex problems by relying on statistical and computational methods. The choice of the method differs depending on the given task and the type of data. For machine learning problems, the task is defined by the data and which solution should be approximated from this dataset [8]. Among the common tasks in machine learning we can distinguish most notably *classification*, *prediction* (of unknown or missing values), *generation* and *density estimation*.
For all these tasks the data is characterized by its domain and its dimensionality. For instance, in the image classification field, the data sample is represented by a vector $\mathbf{x} \in \mathbb{R}^N$, where its features are the intensity values of different pixels.

Classification task example
In this internship, the classification task is actively used. The goal of this task is to categorize the given inputs $X \in \mathbb{R}^{D \times N}$ into C classes $\{1, ..., C\}$. Formally, the goal of learning is to estimate a function $f : \mathbb{R}^N \to \mathbb{N}$ that will assign a numeric label $y \in \{1, ..., C\}$ to each input vector $\mathbf{x} \in X$ so that $y = f(\mathbf{x})$. A seminal example of classification is the object recognition problem, which still represent one of the most studied problem in machine learning [12, 17].

Most machine learning algorithms require the definition and preprocessing of a dataset, the determination of the criterion of optimization, an optimization procedure and a model. It is interesting to note that we can perform various combinations of these components, thereby obtaining a rich variety of different approaches. The learning model is usually defined by a parametric function F_Θ determined by type of task, and that should be closely related to the function f that we try to estimate. In theory, the model is defined based on assumptions about the dataset, such as its distribution and corresponding constraints. In the following, we provide details on these different elements and explicit some types of procedures and models.

Criterion: loss function and risk

The main stage of machine learning is optimization, which requires the definition of a criterion that could inform us on how well the model behaves. This will allow us to define a learning procedure which tries to optimize this criterion. Hence, the *loss function* (also called *error function*), determines a measure of the errors produced by different algorithms. In statistical analysis, the loss function is used for parameter estimation and quantifies the difference between the answer estimated by the model and the real solution.

Formally, we define $\mathcal{M}(\theta)$ as our analytical model that depend on a set of parameters $\theta \in \Theta$, which will be applied to the input dataset X. Therefore, the relationship between an input example $\mathbf{x} \in X$ and the output of the model $\hat{y} \in Y$ is defined by the function f_θ: $\hat{y} = f_\theta(\mathbf{x})$, which is supposed to approximate the function that we want to model $f : y = f(\mathbf{x})$.

In the end, the loss function $L(f(\mathbf{x}), f_\theta(\mathbf{x})) : \mathbb{R}^N \to \mathbb{R}$ is a loss function that measures the error between the true output y and the predicted output $\hat{y}$. There are many different choices regarding L depending on the task and chosen model. The most common loss functions are

- $L(y, \hat{y}) = \mathbf{1}_{\hat{y} \neq y}$ – Binary loss function (classification);
- $L(y, \hat{y}) = (y - \hat{y})^2$ – Quadratic loss function (classification, prediction);
- $L_p(y, \hat{y}) = \|y - \hat{y}\|_p$, where $p \geq 1$ – p-norm loss function (regression, density estimation).

The optimization method will then depend on the chosen loss function. However, it is very important to note that the over-arching goal of machine learning would be to minimize the overall error across all possible unseen data, therefore minimizing the *expected risk*

$$\mathcal{R}_\theta(f) = \mathbb{E}_{\mathbf{x}} \left[L(f(\mathbf{x}), f_\theta(\mathbf{x})) \right] \tag{1}$$

However, this would require to have access to the overall true distribution of the data. Therefore, machine learning will try to minimize a simplified version of this problem by using the *empirical risk*

$$\hat{\mathcal{R}}_\theta(f) = \frac{1}{\|X\|} \sum_{\mathbf{x}_i \in X} L(f(\mathbf{x}_i), f_\theta(\mathbf{x}_i)) \tag{2}$$

Gradient descent

The most efficient method of optimization is the *gradient descent* [29] algorithm, which can be applied to differentiable functions. Let $f_\theta : \mathbb{R}^N \to \mathbb{R}$ be the function defining our model. Then, the gradient of this function $f_\theta(\mathbf{z})$ at a given point $\mathbf{z}$, with a given set of parameters $\theta = (\theta_1, ..., \theta_m) \in \mathbb{R}^M$ is a vector of its partial derivatives

$$\bigtriangledown f_\theta(\mathbf{z}) = \left(\frac{\partial f_\theta(\mathbf{z})}{\partial \theta_1}, ..., \frac{\partial f_\theta(\mathbf{z})}{\partial \theta_n} \right). \tag{3}$$

The gradient descent is an iterative process which aims to reduce the loss function over our model f_θ. Therefore, on the first iteration we initialize our parameters θ^0 to random values and then update their values by choosing the (gradient) direction that minimize the error

$$\theta^{(t+1)} = \theta^{(t)} - \lambda_t \bigtriangledown f_{\theta^{(t)}}(\mathbf{z}), \tag{4}$$

where $\lambda_t \in \mathbb{R}^+$ is the *learning rate*, which defines the magnitude of the update and $t \in \{1, ..., T\}$ is a number of iteration. The output of the algorithm can be defined as the final vector of parameters $\theta^{(T)}$, that minimize the loss function. Others variants of gradient descent algorithms are possible, depending on chosen learning rates and minimized functions [1, 29]. Notably, the Stochastic Gradient Descent (SGD), as well as its extensions are the most widely used in different tasks of machine learning for loss minimization [1].

2.2 Supervised and unsupervised learning

Machine learning includes two main paradigms: *supervised* and *unsupervised*. The major difference between these approaches lie in the presence or absence of a priori information on the examples. The supervised framework can also be called *predictive* learning, as it aims to find the correspondance between given inputs and output labels. In contrast, the objective of unsupervised learning, is to try to uncover the structure of data based directly on the examples themselves, in order to find interesting regularities. Almost all learning tasks are distributed between these two types.

Supervised learning

In supervised algorithms, the input data is divided into two main sets. First, the *training set* is defined by pairs $\mathcal{D}_M = (\mathbf{x}_i, y_i)_{i=1,..,M}$, where M is the number of examples. This dataset defines the matching between N-dimensional vectors $\mathbf{x}_i$ and a nominal label $y_i \in \{1, ..., C\}$. The values of $\mathbf{x}_i$ are also called *features* or *atributes*. This type of training set is one of the gold standard in supervised optimization processes. The other part of the data is a *test set* (defined similarly), which will allow to evaluate the performance of the model on unseen data. It consists of data which is aimed to simulate the behavior of the model in real-world conditions. This type of learning is usually applied to problems of classification and regression.

Classification task

As discussed before, the formalization of the classification problem is an estimation of the function f, such that $y_i = f(\mathbf{x}_i)$, with $(\mathbf{x}_i, y_i) \in \mathcal{D}_M, i = 1, ..., M$. The learning process approximates the prediction function $f_\theta(\mathbf{x})$ using a given labeled training dataset. In terms of probabilistic theory, the selection of a class for a given $\mathbf{x}$ is defined by the most probable label $c \in \{1, ..., C\}$ such that

$$\hat{y} = f_\theta(\mathbf{x}) = \underset{c \in \{1,...,C\}}{argmax} \left(p(y = c|\mathbf{x}, \theta) \right). \tag{5}$$

Finally, the main stage of the process is to evaluate the quality of the model based on predictions made for unseen data of the test set. The classification task is widely applied to many real-world problems, such as image classification, musical genre recognition, spam filtering, speech translation and others.

Overfitting

It is important to note that if the model is too flexible (the complexity of the function is superior to that of the problem), then the found solutions might be too specific to the training set. This phenomenon is called *overfitting* and means that the model will perform well on the training set, while giving unsatisfactory result on the test. To avoid this overfitting the training set is usually subdivided into a *validation* set, which allows to observe the behavior of the model on unseen data, in order to tune its parameters.

Unsupervised learning

Unlike supervised learning, there are usually no target values in the unsupervised case. The objective here is to find some relations directly between the attributes of the data. It is

possible to formalize this task as a density estimation problem. By contrast with supervised learning with a model defined by $p(y_i|\mathbf{x}_i, \theta)$, the unsupervised model can be defined by $p(\mathbf{x}_i|\theta)$, where θ is a parameter vector. So, the major difference of supervised learning is to try to model the data $\mathbf{x}_i$ directly.
This type of learning is commonly used in tasks of clustering, structure discovery, data compression and others. Numerous real-world applications include dimensionality reduction, image inpainting and source separation.

2.3 Convolutional Neural Networks

Neural Network

Inside the family of learning methods, a neural network is defined by sets of connected units (neurons) arranged in layers. The overall neural network field is inspired by the functioning of the human brain. A biological neuron interacts with its neighbors through its dendrites by receiving and sending stimuli to other neurons. When the cumulated signals received by a neuron exceed a given threshold, it produces and sends a, activation signal (response). A mathematical model of these processes in a multi-layer setup was first proposed by Frank Rosenblatt in 1957 and is called the *perceptron*. The basic types of neural networks are defined as multilayer perceptrons, where this multilayer structure allows to learn nonlinear relationships between the input and output of the network.

Generally, an artificial neuron is a function g defined in $\mathbb{R}^d$by expression:

$$g(x) = \sigma(\sum_{m=1}^{M} a_m x_m - a_0) = \sigma(a \cdot \tilde{\mathbf{x}}), \tag{6}$$

where $\mathbf{a} = (a_1, ..., a_M)^t$, a_0 is an activation threshold, $\mathbf{x} = (x_1, ..., x_M)$ are the stimulus received from neuron neighbors, $\tilde{\mathbf{x}} = (-1, x_1, ..., x_m)^t$ and σ is a *sigmoïd* ($\sigma(x) \underset{x\to-\infty}{\to} 0; \sigma(x) \underset{x\to\infty}{\to} 1$).
An artificial neural network (with one *hidden layer*) is a function $f : \mathbb{R}^d \to \mathbb{R}$:

$$f(x) = \sum_{k=1}^{K} w_k \sigma(\mathbf{a}_k \cdot \tilde{\mathbf{x}}) + b, \tag{7}$$

where K is a number of neurones, b is a bias and $w_1, ..., w_K \in \mathbb{R}$ are weights, characterizing the network.
The first and the last layers of a neural network are respectively called *input* and *output* layers. All intermediate layers called hidden and its values are unknown during the learning.

Convolutional Neural Network

Convolutional Neural Networks (CNN) are similar to multilayer neural networks, but rely on slightly different operations by using layers of *convolution* and *pooling*. Each layer is defined by multiple filters, convolved across a given input or the previous layer output. By replacing the linear transform of a neural network by a convolution and given that $g(\mathbf{x})^{(1)} = \mathbf{x}$, the activation of the layer $l + 1$ is described by:

$$z^{(l+1)} = w^{(l)} * g(x)^{(l)} + b^{(l)}; \tag{8}$$

$$g(x)^{(l+1)} = \sigma(z^{(l+1)}), \tag{9}$$

where $w^{(l)}$ is a vector of weights and $b^{(l)}$ is a bias for layer l.
Therefore, multiple convolution *kernels* are computed over the previous activations, leading to several *feature maps* per layer. When the data is sufficiently *stationary*, i.e have the same statistics in every parts, the convolution can be an appropriate operation of filtering. The filter overpasses a $N \times D$-dimensional data matrix and output a new feature map. Kernel size can vary from layer to layer. So, during the learning multiple filters can be applied. The particularity of the operation of convolution is that it provides a spatially-logical connection between the different elements of data and transmits it to the next layer. In the image processing, convolution layers allow to identify the shape parameters, detecting the vertical and horizontal directions of the lines. In the networks with high number of hidden layers, convolution encourages a sharing of parameters and realizes the *feature extraction*. [26].
The *pooling* layers then subsample the data into a lower dimensionality by choosing the maximum (or average) activations across a given window. Ultimately, all networks produce a vector of representation $g(x)^{(n)}$ (activations of the deepest units), which are the highest-order features extracted from the input.
By using convolution and pooling, CNN exhibits very interesting properties such as local spatio-temporal connectivity and local translation invariance for any given input [16]. On top of these convolution and pooling layers, CNNs usually rely on fully-connected layers to combine the features extracted from all convolution and pooling layers and so perform classification or regression.

CNNs are used for classification problems by evaluating the errors made by the network and then by backpropagating the gradients of these errors with respect to the parameters (weights and biases) into all the layers iteratively [2, 20].

Back-propagation

The interactions between layers in the network occur in two ways. One of them is the *forward pass*, which defines the output of a given layer depending on the previous one. This follows the activation functions defined previously (Equation 8). However, the key goal of learning algorithms is to optimize the parameters of this model. Therefore, the other important interaction between layers is to define the *error back-propagation*. This operation defines the relation between layers and the error function. This will then allow to update the weights of layers by using gradient descent. An explicit mathematical formulation of back-propagation process is given in [24].
The process of back-propagation described above allows to CNN to output the high-level features. In order to explore the non-linearity of these features in CNN, the full connected layers are used.

Deep convolutional neural network

The general idea of Deep Neural Network (DNN) is that they provide many hidden layers between the input and output layers [10]. Each layer provides increasingly higher-level and complex features by uncovering correlations in the previous layer through non-linear transforms. Figure 1 illustrates that the high number of hidden layers generate a bigger number of relations between neurons. This makes a learning algorithm more flexible and powerful.

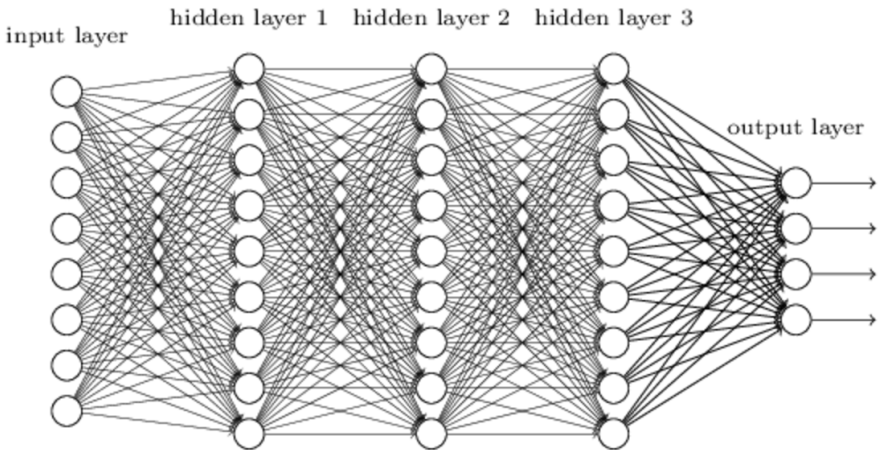

Figure 1: DNN posses a big number of hidden layers. Image taken from [28]

The recent breakthroughs in *deep learning* provided astonishing results, strongly outperforming state-of-the-art models in information retrieval tasks [2, 31].

Receptive fields

The notion of *receptive fields* is also taken from neuroscience. It represents a region of neurons with their receptors being sensible to the specific stimulus received from nearest neurons. The influence of various stimulus can change the state of this neuron. In visual systems, the mechanism of receptive fields is involved in feature detection [15]. In deep CNNs, receptive fields are a very important concept. Technically, this is an hyperparameter corresponding to the filter size. Some researches analyzed the models of artificial receptive fields (Effective RF) and its impact on the neurons during the learning process [23, 13]. The authors pay attention to the correct choice of the size of receptive fields and the differences in neuron responses from deeper layers. Indeed, the features have a low resolution on the first layers and higher *resolution* in deeper layers. So, we can consider receptive fields as feature detectors, which increase the detection capacities of the CNN from low-level to high-level layers. The formal definition and some example of applications in image processing like edge or orientation detection are given explicitly in [15].

3 Style transfer

In this section, we introduce the concepts of style transfer and the state-of-the-art in image processing. Then, we will talk about its application to musical signals. As previously discussed, we will use the term *style transfer* loosely when discussing *musical style*, as an analogy to the existing works in images.

3.1 Concept of style transfer

Neural style transfer

Traditional methods of style transfer in image processing mainly focus on non-parametric patch-based texture synthesis and transfer [40]. Preferably, style transfer algorithms should retrieve the semantic content from the target example and combine it with the texture content

of the origin example. Hence, the idea is to resample small patches of pixels from an original source texture image (representing the *style*) and try to fit these to a corresponding target image (representing the *content*). Different methods were proposed to improve the quality of the patch-based synthesis and constrain the structure of the target image [7, 38, 37]. Recently, neural style transfer [18] has demonstrated remarkable results for image stylization by taking advantage of the powerful representation offered by deep CNNs. The main problem here is to find which parts of the network could discriminate between the content and the style in order to generate an output image with appropriate quality.
To that end, the neural transfer algorithm will use the activation of convolutional filters and different layers. Indeed, as previously discussed, these filters can be seen as high-level feature detectors. Hence, when one filter activates it represents a specific content at a specific location. Regarding style, the reccurent co-activation of low-level filters have been shown to faithfully describe the texture content of an image [7]. This information can be found directly by computing the Gram matrices of the neural activations from different layers of a CNN to represent the artistic style of an image. Then, it uses an iterative optimization method to generate a new image from white noise by trying to match as closely as possible the neural activations of the content image while matching the Gram matrices of the style image.
Although several works have been proposed to further improve the original neural style transfer [19], its application to audio data remains open because of the wide difference in the definition of audio style. In the following, we expound the application of neural style transfer as defined in [18].

Content representation

Each layer in a convolutional network is defined by a set of filters. The complexity of each filter (regarding the level of information that it processes) depends on the depth of its corresponding layer. Hence, when propagating an image through a network, at the each level this image produce a particular activation of the filter responses $F^l \in \mathcal{R}^{N_l \times M_l}$, where a layer l has N_l feature maps of size M_l. So, for each layer l, F^l_{ij} is the value of the activation of filter i at position j. We call F^l an activation matrix.
If $\tilde{p}$ is a source image and $\tilde{x}$ is a target image, we define as P^l_{ij} and F^l_{ij} their respective activation matrices at layer l. If we compare these activations, we can evaluate the difference in the content between $\tilde{p}$ and $\tilde{x}$, leading to the content loss function

$$\mathcal{L}^l_{content}(\tilde{x}, \tilde{p}) = \frac{1}{2} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} (F^l_{ij} - P^l_{ij})^2. \tag{10}$$

It is interesting to note that initially $\tilde{x}$ can be a white noise (image) or any other random set of variables. But, with decreasing error the structure of $\tilde{x}$ will converge to $\tilde{p}$, revealing the same features (lines, object edges, depth).
For the error minimization, the gradient descent method is used. It can be calculated by relying on standard back-propagation from the derivative of the squared-error loss with respect to the source image

$$\frac{\partial \mathcal{L}_{content}}{\partial F^l_{ij}} = \begin{cases} (F^l_{ij} - P^l_{ij}), & \text{if } F^l_{ij} > 0 \\ 0, & \text{if } F^l_{ij} < 0. \end{cases} \tag{11}$$

This process allows to modify the filter response of a target image, so that its activations gets closer to the response of the original image. As the layer used gets deeper, representations

become more focused on the high-level information of the picture (general outlines), while obliviating its low-level details (local pixels and textures) [3]. Thence, we will treat the high level features as a *content representation* of source image.

Style representation

As discussed previously, we aim to use the information associated to the texture of a target picture as a notion of style. We are interested by the correlations between different feature responses with respect to the feature maps of each layer. These correlations are defined by Gram matrices $G^l \in \mathcal{R}^{N_l \times M_l}$

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{kj}, \tag{12}$$

where F^l_{ik} and F^l_{kj} are feature maps i and j in layer l. We can see that the Gram matrices represent the correlation between given features across any spatial position (as it sums their co-activations). Following the same method of gradient descent, we can optimize the Gram matrix of the random image to be closer to a target Gram matrix of a given layer l

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} (G^l_{ij} - A^l_{ij})^2 \tag{13}$$

where G^l and A^l are respectively the style representations of the input and generated pictures at layer l. The gradient of E_l is computed using standard back-propagation

$$\frac{\partial E_l}{\partial F^l_{ij}} = \begin{cases} \frac{1}{N_l^2 M_l^2} (F^l)^T (G^l_{ij} - A^l_{ij}), & \text{if } F^l_{ij} > 0 \\ 0, & \text{if } F^l_{ij} < 0. \end{cases} \tag{14}$$

Interestingly, we can combine the information from the features at different layers; in order to obtain a multiscale representation of style from a given image. If we suppose $\tilde{a}$ is the source texture image, then the total loss function becomes

$$\mathcal{L}_{style}(\tilde{a}, \tilde{p}) = \sum_l \omega_l E_l, \tag{15}$$

where ω_l is a weighting coefficient that can be chosen arbitrarily, and $\tilde{p}$ represents the pixel information of the source image that can be optimized through complete back-propagation.

Content and style matching

Formally, the style transfer looks like a trade-off between component and style components:

$$\mathcal{L}_{total}(\tilde{x}, \tilde{a}, \tilde{p}) = \alpha \mathcal{L}_{content}(\tilde{x}, \tilde{p}) + \beta \mathcal{L}_{style}(\tilde{a}, \tilde{p}) \tag{16}$$

The proportions of each of two components of a new picture depends on parameters α and β. The question of the optimal relation between these two representations is open and goes into the field of the subjective worldview.
The complete scheme of the artistic image style transfer is by [18] presented on the Figure 2.

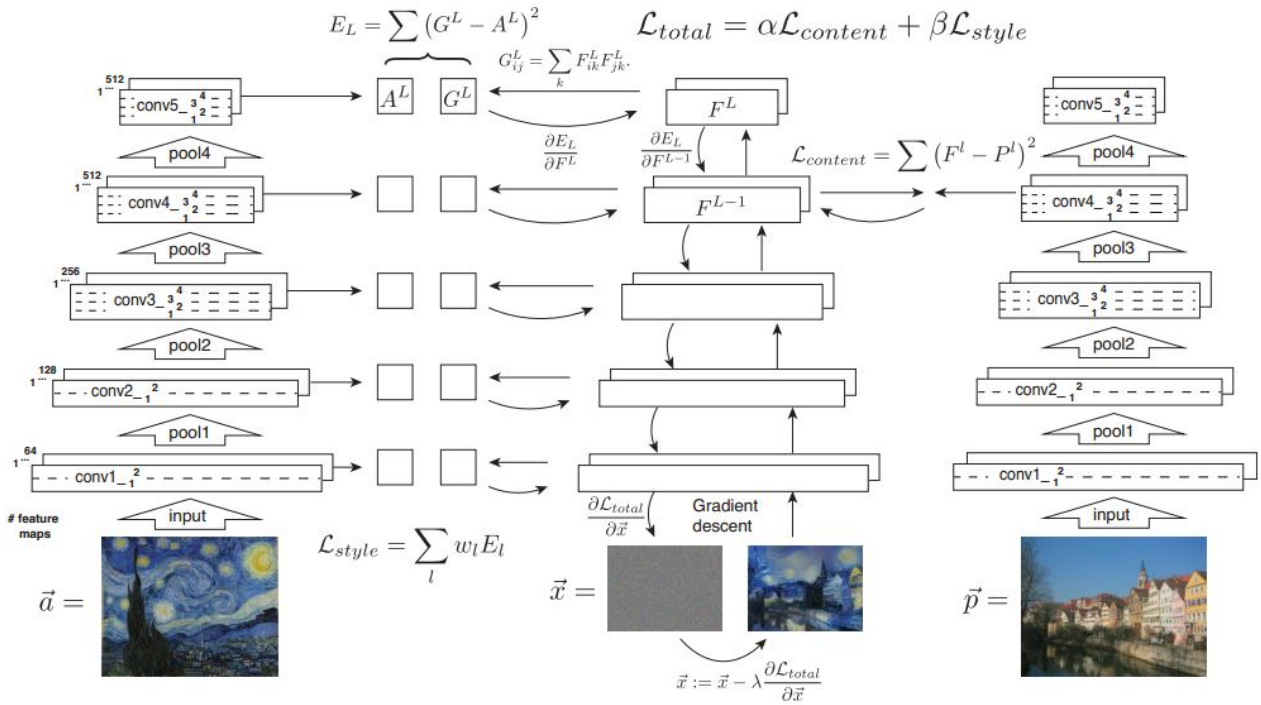

Figure 2: The complex schema of transfer algorithm. The style prototype $\vec{a}$ passes through the CNN and the representations A^l for all l are stored. The content prototype $\vec{p}$ passes through the network, and content representation P^l is stored. $\vec{x}$ - a white noise image passes through the network and results style features G^l and content features F^l. On one's hand, on each level the loss between A^l and G^l computes and gives $\mathcal{L}_{style}$; on the other hand, the loss $\mathcal{L}_{content}$ is computed. The total loss $\mathcal{L}_{total}$ is a linear combination of $\mathcal{L}_{content}$ and $\mathcal{L}_{style}$. By back-propagation $\vec{x}$ is iteratively updated. Image taken from [18]

3.2 Application to audio related tasks

The neural algorithm of the texture transfer (and its variations [6, 18, 19]) can be applied to different types of data. The most important is that the dataset can be represented as multidimensional tensors. Thus, we can apply this algorithm to the sound. As input, we can use spectrograms, representing the spectral transforms.

Notion of *musical content*

The "content aspect" is very well mirrored in the music domain and can be extracted using CNN. In order to represent the content of a musical composition we employ the activation matrices of feature maps. Each matrix corresponds to activation of zones where main trends of spectrogram can appear. In music composition these trends can represent the melody, general structure, the most important frequencies.

Notion of *musical style*

In the algorithm of the artistic image style transfer, the representation of "style" is a result of Gram matrix optimization. Indeed, this matrix represents the change of the basis of the space where it is defined. It means that the modification of the Gram matrix impacts the change of the space. Actually, the modifications involve the texture aspect of the image. The

result of the transfer of this concept in the sound processing domain is a not style, but a *local musical texture* transfer. We call this method *musical style transfer* by language abuse.

4 Spectrotemporal analysis based on auditory cortical physiology

The human auditory system embeds complex sound processing mechanisms which are able to obtain, process and transfer audio signals to the brain. This system can detect, recognize and separate a wide variety of signals such as speech, environmental sounds or music. To understand these complex neural representations in the primary auditory cortex (A1), researchers have studied the responses of the brain to dynamic sounds, leading to the Spectro-Temporal Receptive Field (STRF) [35] representation.

4.1 STRF-representation

Fundamentals of STRF

The STRF are used to approximate the sound representation computed by neurons in the human auditory cortex. This representation tries to fully characterize the response of neurons and their temporal dynamics when sounds with rich spectro-temporal content are heard by humans. Neurons react differently, in particular, with different latencies between sound covering different parts of the spectrum. The STRF are computed from the responses to elementary *ripples*, a family of sounds with drifting sinusoidal spectral envelopes [35]. The collection of neural activations generated by these ripples is the spectro-temporal *transfer function*. Hence, the function $STRF(f,t)$ will describe the response of neurons to the stimulus at frequency f and time t. This is depicted in Figure 3, where we can see that the activation of a given neuron can be either reinforced by some part of the spectrum (called *excitatory influence* of a given frequency at a given time, represented here by a plus circle) or inhibited by other regions (called *inhibitory influence* represented by minus circles).

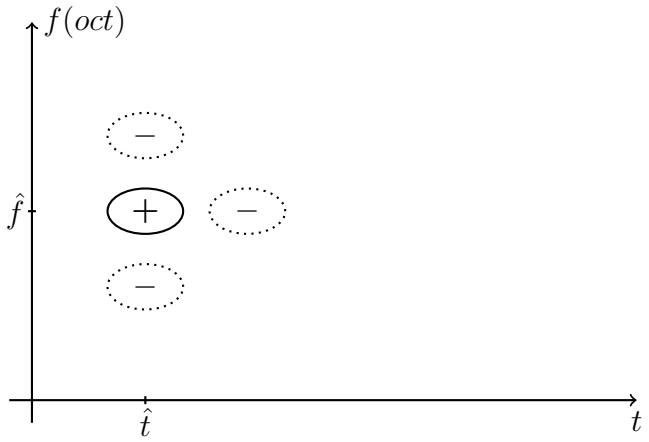

Figure 3: Schematic plot of a $STRF(f,t)$ representation; plus circles represent spectrum regions that participate to the activation of a neuron (excitatory influence) and minus circles inhibit its activation (inhibitory influence)

In order to obtain these functions, the moving ripple spectral profile that is used as input is presented in Figure 4. Here, the black regions depict zones of the ripples excited by certain

frequencies and white regions designate zones where the response of ripples is suppressed. In general, any broadband and dynamic sound with complex spectro-temporal envelopes can be expressed as a linear sum of individual ripples [5].

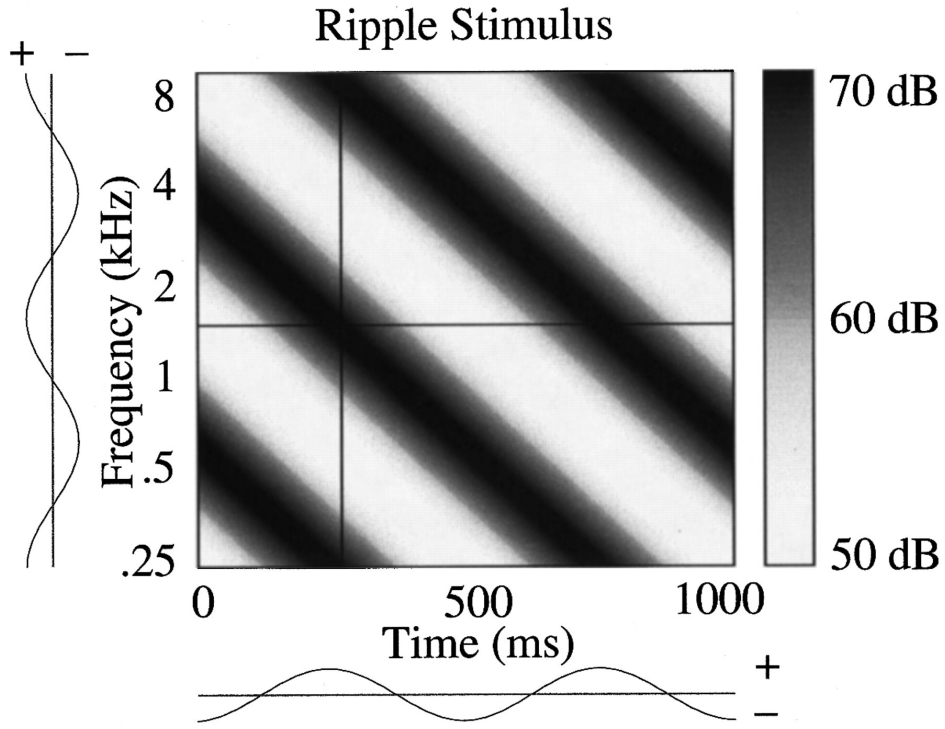

Figure 4: Envelope of a moving ripple. Image taken from [5]

Cochlea & auditory spectrogram

On the first early stage, sound is processed by the cochlea, a spiral in the inner ear, transforming sound vibrations to neural impulses. The unidimensional pressure waveform is turned into 2-dimensional templates of neural activity (called *auditory spectrogram*) which we can represent on the typical time-frequency axes. Thus, the cochlea have the mission in the auditory system of early audio treatment and play a role similar to that of a filterbank for the full spectral analysis. More explicitly, when the acoustic signal $s(t)$ goes through the cochlea, it is processed by filters, each of which can be determined by its impulse response $h(t, x)$, leading to the following operation

$$y_{coch}(t, x) = s(t) \otimes_t h(t, x), \tag{17}$$

where $\otimes_t$ is convolution operation in time domain. The result $y_{coch}(t, x)$ is transmitted to the auditory nerve, where hair cells perform a compression operation $g(t)$ and later on the membrane acts as a low-pass filter $w(t)$ on the input. Hence, the resulting signal $y_{an}(t, x)$ can be modeled as follows

$$y_{an}(t, x) = g(\partial_t y_{coch}(t, x)) \otimes_t w(t). \tag{18}$$

Next, this signal goes through a lateral inhibitory network $y_{LIN}(t, x)$ (described in detail in [30]), which can be approximated by the positive derivative of the auditory nerve signal $y_{LIN}(t, x) = max(\partial_t y_{AN}(t, x), 0)$. Finally, the output of this early stage is

$$y_{fin}(t, x) = y_{LIN}(t, x) \otimes_t \mu(t, \tau), \tag{19}$$

where $\mu(t, \tau)$ is a short window with parameter τ denoting the midbrain phase loss.
The auditory spectrogram is clearly discriminative of the type of stimulus, with high specificities between noise, separated tones, high and low frequencies, harmonic complexes, speech and music. Moreover, it can reflect intonations and nuances, such as a vibrato or bowing characteristics in violin sounds [33].

Auditory cortex and spectrotemporal analysis

The next stage of processing will exhibit the modulations inside both the spectral and temporal components of the auditory spectrogram $y_{fin}(t, x)$. This signal is processed by the primary auditory cortex through another filter bank, which is sensitive to different modulations parameters. These parameters describe slow to fast temporal *rates* and dense to expanded spectral *scales*. It is important to note that the STRFs are centered on particular frequencies CFs associated with the amplitude, spacing and phase. These characteristics reflect the perception of various timbre and help to distinguish instruments, voices and speech.

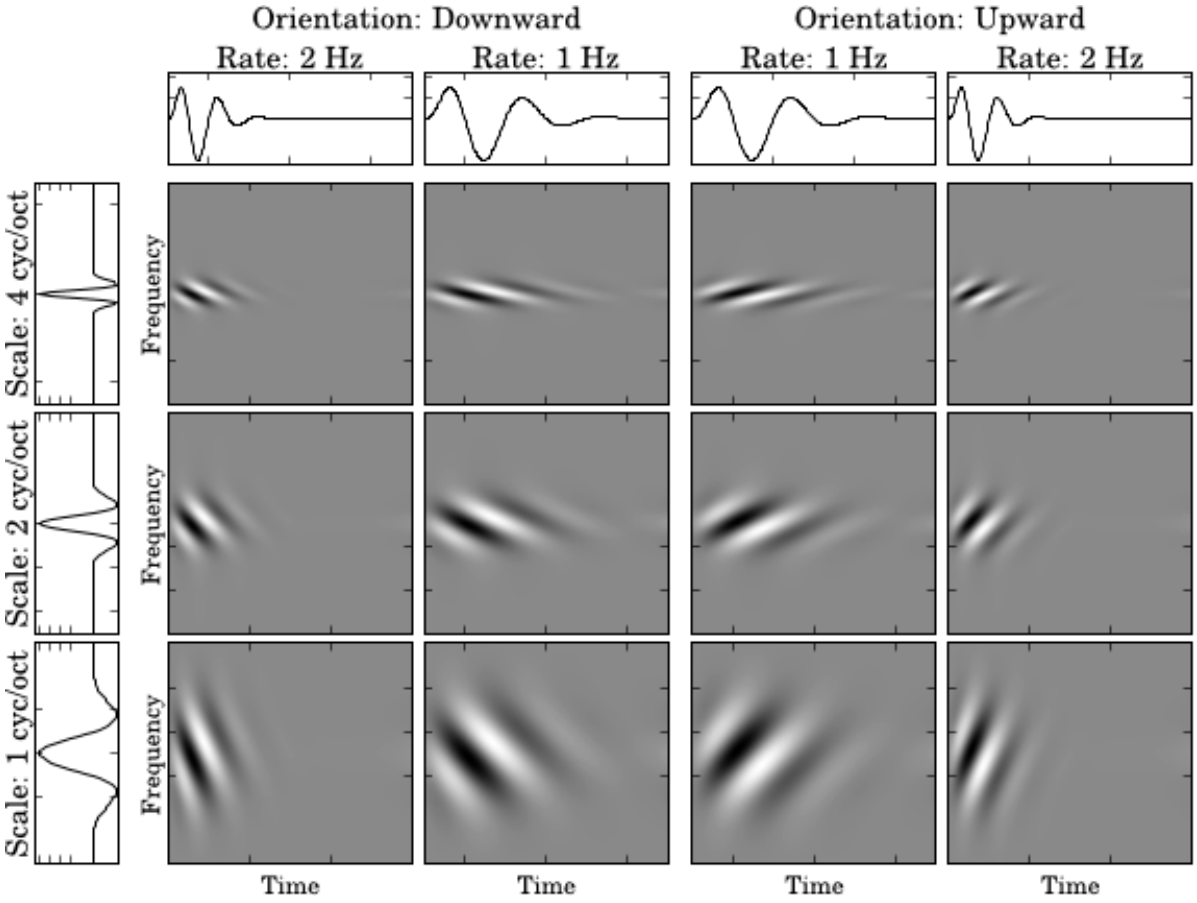

Figure 5: Example of a STRFs. Image taken from [34]

Mathematical formulation

Now that we have transfer functions $STRF(f, t)$ that model the processing of the auditory system, given an input signal spectrogram $S(f, t)$ (computed with any type of spectral transform), we can compute the set of neural activations $O(t)$ by [21]

$$O(t) = \int \int STRF(f, t) S(f, t - \tau) d\tau df + \epsilon. \tag{20}$$

The transfer function is separable, therefore, it can be factorized into a product of purely temporal and purely spectral functions. The explicit and complex research of the STRF was undertaken in [21]

5 Dimensionality reduction methods

The STRF is a complete representation providing four-dimensional tensors defined by time, frequency, scale, and rate. However, its major drawback is that it generates tensors of very high dimensionality, which could hamper our future learning algorithm. Hence, one of the major way to deal with these properties in a computational setup is to rely on *dimensionality reduction* methods, such as the Principal Components Analysis (PCA), based on analyzing the variance of various dimensions.

5.1 From PCA to Multilinear-PCA

Principal components Analysis (PCA) is a dimensionality reduction procedure relying on statistical methods that analyze the variance of different dimensions. The goal is to perform a maximization of the variance of different dimensions, by finding an orthogonal projection in the subspaces of the dataset (usually after centering and normalizing the data). In other words, only the components with a rather large variation will be chosen. One of the key disadvantage of the PCA approach is that it considers every dimensions in the same way, by proceeding to a vectorization of the input tensor (leading to a matrix $N \times D$ with N elements and D dimensions). However, in this work, we will need to work with STRF tensors that have a clear structure ($N \times F \times T \times S \times R$), which would be lost with the PCA. Therefore, we will introduce a specific tensorial extension of the PCA, namely Multilinear PCA.

PCA

PCA is a well-known method of analyzing the general behavior of data components. The objective of this technique is to extract the elements with maximal variance in each of dimensions and to reconstruct a decimated representation of the original, based on these elements. The algorithm of PCA consist in the next steps. Firstly, the data need to be centralized. We calculate a mean from each of vector dimensions of data space and subtract the means from the origin values.
Secondly, we calculate the matrix of covariance. From this square matrix we can compute the eigenvectors and the eigenvalues which need to be a *unit* vectors (of the length 1). In fact, the eigenvectors represent a new basis of the space of original values, constructed with respect of its general *trend*. In other words, the eigenvectors correspond exactly to the main directions of distribution of values in space, thereby characterizing the input dataset.
Thirdly, it is necessary to define the *principal components* and to form the *feature vector*. The main idea is that the eigenvector corresponding to largest eigenvalue is a principal component of dataset. After obtaining the eigenvectors from covariance matrix we arrange them in decreasing order of corresponding eigenvalues. On this stage, in order to reduce we can ignore the lowest eigenvalues and discard corresponding eigenvectors. Some data will be lost, but, by hypothesis of the method it is non-significant information.
More generally, the feature vector E is a matrix consisting of all resting eigenvectors: $E = (e_1, e_2, ..., e_p)$, where p eigenvalues have been chosen from n founded from covariance matrix initially.

The final step is to project the feature vector on the space of the original dataset. Technically, it is a multiplication of the matrix of eigenvectors in the rows by transposed centralized data matrix:

$$X_{PCA} = FV^T \times X_c^T. \tag{21}$$

The final data X_{PCA} includes the items from original dataset in columns, and number of rows corresponds to number of dimensions.
The PCA-method classify the data elements by its significance, which is characterized by eigenvalues. It lets to separate with a help of some criterion (for example, a percent rate of ignored information) less significant elements from important items and leave them out. However, in the case of high-level dimensions this algorithm may require a powerful computational capacities.

Multilinear PCA (MPCA)

When objects are defined by structures of more than two dimensions (matrix), we need to rely on tensors (generalization of the matrices to three or more dimensions). This can be a useful representation of complex sound transforms or video data. The order, also called *mode*, is defined by the number of dimensions of space in which the object can be placed. For example, video is a three-order tensor, constructed by rows and columns of pixels changing in time.
By definition, tensors have a more complex structure and the treatment of these objects can be computationally expensive. Hence, if we want to use the statistical learning methods introduced previously with this type of input, we run towards the risk of largely overpassing the memory capabilities of computers. As previously argued, applying PCA could be a good idea to alleviate this problem, but in the case of multi-order tensors with high number of intercorrelated components it may be insufficient. Indeed, the PCA approach does not account for the inherent tensorial structures and potential correlations amongst them. This comes form the fact that the classical PCA requires the vectorization of input tensors. For example, a tensor of size $128 \times 216 \times 10$ is transformed to a vector of size 276480×1. In result we have a 1-order object, where the relations between different dimensions of a same mode are lost. Therefore, if we apply PCA directly to tensor ogjects, we are forced to ignore the multidimensional nature of this structure. These observations motivate the introduction of the Multilinear Principle Component Analysis method (MPCA).

MPCA framework

Detailed mathematical justifications, as well as the questions of convergence of MPCA are discussed fully in [22]. Some details and notations are taken from [9, 14]. Now we give a general algorithm of proposed method. The notations used in follows in this paragraph are taken from [22].
The next steps describe the MPCA algorithm :

1. *Centralization*
   As in the case of two-dimensional space, we need some preprocessing operations on input samples $\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m = 1, ..., M$.
   Applying to tensors, the centering will look like this: $\bar{\mathcal{X}}_m = \mathcal{X}_m - \bar{\mathcal{X}}$, where $\bar{\mathcal{X}} = \frac{1}{M} \sum_{m=1}^{M} \mathcal{X}_m$.

2. *Eigen-decomposition*
   The next step is a calculation of eigenvectors corresponding to the largest eigenvalues P_n, for $n = 1, ..., N$. As result, we obtain the eigen-decomposition of n-mode matrix full projection

$$\Phi^{(n)*} = \sum_{m=1}^{M} \tilde{\mathbf{X}}_{m(n)} * \tilde{\mathbf{X}}_{m(n)}^{T} \tag{22}$$

   and set of eigenvectors $\tilde{\mathbf{U}}^{(n)}$.

3. *Optimization*
   Then, $k = 1, ..., K$ times, where K is an arbitrary defined number of iterations, we produce by sequence the followings operations:

   - Calculate $\tilde{\mathcal{Y}}_m = \tilde{\mathcal{X}}_m \times_1 \tilde{\mathbf{U}}^{(1)^T} \times_2 \tilde{\mathbf{U}}^{(2)^T} ... \times_N \tilde{\mathbf{U}}^{(N)^T}$ for $m = 1, ..., M$;
   - Calculate $\Psi_{\mathcal{Y}_0} = \sum_{m=1}^{M} \|\tilde{\mathcal{Y}}_m\|_F^2$, where $\bar{\tilde{\mathcal{Y}}}$ is zero if $\tilde{\mathcal{X}}_m$ is centered.
   - For all $n = 1, ..., N$ calculate the new *projection* matrix $\tilde{\mathbf{U}}^{(n)}$, which is a key operation of MPCA:

$$\tilde{\mathbf{U}}^{(n)} = \underset{\tilde{\mathbf{U}}^{(1)}, \tilde{\mathbf{U}}^{(2)}, ..., \tilde{\mathbf{U}}^{(N)}}{argmax} \Psi_{\mathcal{Y}}, \tag{23}$$

     where $\Psi_{\mathcal{Y}}$ is a total tensor scatter. These matrices $\tilde{\mathbf{U}}^{(n)}$ consist in P_n eigenvectors of n-mode matrix $\Phi^{(n)}$ and correspond to biggest P_n eigenvalues.
     After that, the new projection $\tilde{\mathcal{Y}}_m$ and scatter $\Psi_{\mathcal{Y}_k}$ can be defined. The iteration continues until the scatter change sufficiently or, in other words, if η is some predefined threshold and if $\Psi_{\mathcal{Y}_k} - \Psi_{\mathcal{Y}_{k-1}} < \eta$, then go to the next step.

4. *Projection*
   The final reduced tensor is a result of a multilinear projection. It is defined by the vector

$$\mathcal{Y} = \mathcal{X}_m \times_1 \tilde{\mathbf{U}}^{(1)^T} \times_2 \tilde{\mathbf{U}}^{(2)^T} ... \times_N \tilde{\mathbf{U}}^{(N)^T}, m = 1, .., M. \tag{24}$$

This MPCA-method proposed in [22] allows to reduce the number of input dimensions, while at the same time keeping the internal mode-wise relationships [27].

Part II
Experimentations & results

6 Preprocessing

6.1 Spectral transform

In this internship, we use the Torch package developed in the Lua language that offers convenient and practical tools for learning networks. We started by developing a toolbox created to simplify the utilization of Deep Neural Networks and include useful packages of cortical representations and other possible transforms like Mel, STFT or wavelets. The Torch-toolbox also includes tools that can import, preprocess and visualize data, build models and convenient criteria of optimization, and finally construct learning structures with supervised and unsupervised options.
By using the cortical-toolbox we obtained 8 datasets corresponding to 8 2D-transforms: Mel spectrograms, Mel-Frequency Cepstral Coefficients (MFCC), Wavelet-transforms, Short-Time Fourier Transforms (STFT), Scattering transforms, Chroma, gammatone transforms and cochleograms.

6.2 Dataset

For all experiments, we used a GTZAN dataset, a public set of music samples well-known in MIR community. It consists in 1000 half-minute music examples, labeled in 10 classes representing musical classes. It is most used especially for music genre recognition task (MGR).

			last.fm	
Label	**ENMFP**	**self**	**song (no. tags)**	**artist (no. tags)**
Blues	63	100	75 (2904)	25 (2061)
Classical	63	97	12 (400)	85 (4044)
Country	54	95	81 (2475)	13 (585)
Disco	52	90	82 (5041)	8 (194)
Hip hop	64	96	90 (6289)	5 (263)
Jazz	65	80	54 (1127)	26 (2102)
Metal	65	83	73 (5579)	10 (786)
Pop	59	96	89 (7173)	7 (665)
Reggae	54	82	73 (4352)	9 (616)
Rock	67	100	99 (7227)	1 (100)
Total	60.6%	91.9%	72.8% (42567)	18.9% (11416)

Figure 6: The content of GTZAN. In columns: number of samples identified by the Echo Nest Musical Fingerprinter (ENMFP); manual searching (self); number of songs in last.fm; number of artists in last.fm (for tracks not found). Table taken from [32].

The database GITZAN contains inaccuracies of at least three types: repetitions (it has been found 50 exact repetitions), mislabelings and distortions [32]. Those inaccuracies impact the results of learning but they do not means that we cannot use effectively this dataset. One

solutions of this problem might be a "smart" repartition of samples between training, validation and testing sets. We used a manual distribution without mislabelings and repetitions in the same subset.

6.3 Analysis of audio transforms

Two bidimensional stationarity tests (trend stationarity for each of 4 vector dimensions) was used to motivate the use of STRF transforms. The *stationarity rate* was measured by scalar coefficient $\in [0; 1]$. It was compare with the rates of 8 others spectral transforms.
In order to assess the discriminative power of the models, we start by performing genre classification. We evaluate this task by relying on the 10 genres from the GTZAN set and the 10 (different) genres from the Million Song Dataset (MSD). Finally, we evaluate the task of genre classification on the GTZAN set but with models that had first been pretrained on the genre classification task for the MSD set. This will allow us to evaluate the capacity of the model to perform *feature transfer* between different datasets.

6.4 Reduction of dimensionality

The data in our works as in many other cases are represented by a set of big dimension tensors. The elements of tensors have three or more indexes (depends on dimensionality of object) and correspond to the order of the tensors. The space of spectro-temporal modulations has 4 dimensions and is naturally represented by 4-order tensors. However, high level dimensions provide more of data volume that implies obstacles for machine learning. It also raises a question of memory and calculation capacities of computing devices. STRF-representation is a complex 4-dimensional representation of sound and requires big computational capacities. This technical problem can be resolved by different methods of reduction. At the same time, we would keep the advantageous properties of this representation which is multidimensionality, and parameters of modulations in both spectral and time domains.

The problem comes down to decimate data by reducing the number of points in the spectro-temporal space. In this internship, we used two main concepts: an empirical dimensions evaluation (learning channel by channel) and the dimension reduction by MPCA. Then, we evaluated and compared the results of both models. As a testing problem, we used a typical task of music genre classification.

7 Models

Parameters of models

During this internship, we used two different learning models to deal with problems of classification and dimension reduction studying.
For the classification task, the set was divided manually in three sets: train, validation and test. This division avoided problems about repetitions and mislabeling of the dataset GTZAN. The CNN model with 4 convolutional layers was used. Each layer consists in 300 neurons, the size of convolution is $64 \times 64 \times 64 \times 64$. Two first kernels are the size 11×11 and two last - 5×5 with a strides 1, 2, 2, 2 respectively. Pooling size is $3 \times 3 \times 3 \times 3$. The number of channels is 1 for channel-by-channel analysis and 198 for complex analysis of STRF. As the activation function σ, non-linear function ReLU was chosen.
In order to reduce the dimensionality of STRF-representations, we used the method of MPCA

with a variation order 0.95 for normalized data.
The both models were adapted to the size of the transforms and trained by batches 50 and 5 samples respectively.

8 Music style transfer

Particularities of audio data

The image in RGB-color model can be digitally represented as 3 channels of two-dimensional matrices. The lines and columns are equivalent in meaning, that implies that image is indifferent by rotation. It is not the case for audio signal. Sound data can be viewed as a spectrogram (built with Fourier transform or otherwise) and represents two-dimensional matrix or multidimensional tensor. But in case of typical spectrogram of short-time Fourier transform (STFT), properties of both frequency and time dimension are not similar. However, any two-dimensional sound transform TxF can be treated as an image, if it is considered as 1xT image with F channels [37].
One more problem is the sound reconstruction of the waveform from spectrogram with respect of axes. For STRF, it can be Griffin-Lim algorithm and other inverse function for given transforms.

Updating of image processing algorithms for sound processing

After applying the dimension reduction algorithm to the STRF-representations of the dataset, we can use tensors as the input in *music style transfer* algorithm. This algorithm is based on the methods proposed by [18, 19].

9 Results

We evaluated each of spectrotemporal subsets of STRF by learning on classification task. The Figure 7 demonstrate the distribution of learning rate of each subset.

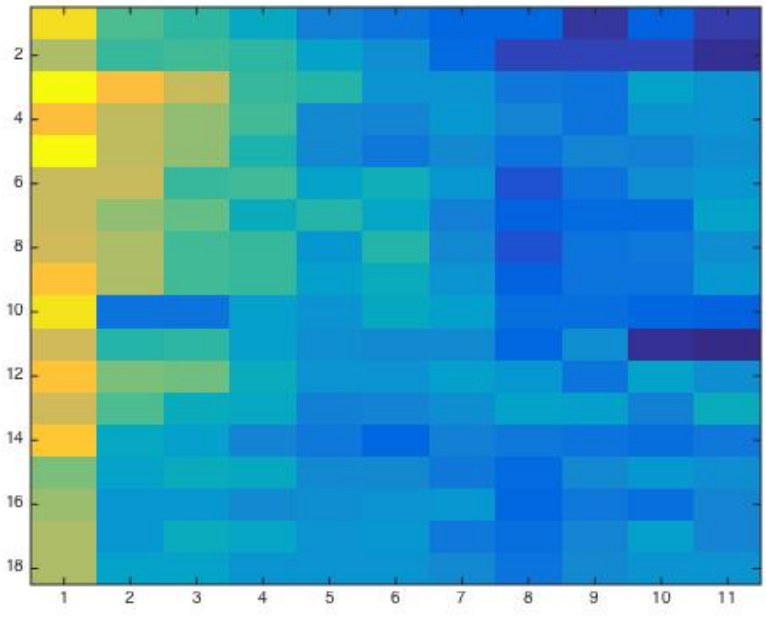

Figure 7: The most representative spectrotemporal subsets of STRF. On the axis of abscissa the score dimension is represented. On the axis of ordinates the rate dimension is represented.

The other calculations are currently being processed:
Stationarity learning;
Results of reduction of dimensionality bu MPCA;
Texture transfer algorithm.

Part III
Conclusion

10 Discussion

The goal of this internship was to analyze the different sound representations, in particular Spectro-Temporal Receptive Fields, and to produce the algorithms of music style transfer using this type of transform.
During this internship we studied neural networks, in particular Deep CNN, produced different sound transforms from the base of dataset GTZAN, analyzed stationarity properties of these transforms, studied in detail the Spectro-Temporal Receptive Fields and we tried to apply the algorithm of music style transfer inspired by [39] to the STRF. Before applying STRF to the neural network, we reduced its dimensionality by Multilinear PCA.

10.1 Advantages & shortcomings

We already noted that the algorithm shown in this work is called *artistic style transfer* . Actually, it is langage injury. *style* doesn't represent what specialists understand with this term. We suppose that everything that is not classified as content is style, but the notion *context* also consists in obscure aspects. The notion of style requires more subtle approaches.
An other imperfection is that the STRF still is a transform requiring large memory and calculation capacities. For instance, we are forced to use the methods of dimension reduction. It means that we partly loose some informations about the sound.
However, we believe that the MPCA is an efficient method of dimension reduction for the tensors of high dimensionality. It allows to keep maximum of useful informations and get rid of the stationary dimensions of the STRF-transform.
Even after the dimension number reduction, the STRF demonstrates a good score compared to the other sound transforms in the typical classification task.

10.2 Perspectives & future work

Over the last years, MIR keeps consistency to treat musical style transfer problems. However, the notion of music style remains obscure and abstract. Nevertheless, we can try to define this notion. With ears and intuition, several elements take a hand to specify music style. Firstly, it deals with instrumental composition of music. Indeed, such instruments are more typical of some centuries or compositors. Secondly, voices and the interpreters techniques are crucial to refer a composition to a particular style. Thirdly, the artwork structure often can be linked to a particular composer. Taking into account those specifications and some others in the algorithms conception could lead to a finest notion of musical style. The use of those links with other metadata (like for instance sampling frequency or the interpreter name if it is available) could help to treat classification problems. Finally, The best we can split musical styles from musical contents, the best we can improve algorithm development. It is also possible to evaluate different models based on the overall accuracy (ratio of correctly classified examples) and the number of epochs to obtain this accuracy (before overfitting occurs as assessed with the validation set).
As a possible application of algorithms studied during this internship, we see a problem

of *cover versions style recognition*. Cover versions or, more generally, remakes are popular author music compositions (song) interpreted by other artists than the original ones. Often, new versions contain the elements of the original compositions, but also with new variants of musical arrangement. In order to detect remakes, cover recognition methods can be used. The recognition requires the identification of the similar musical contents (music structure, general melody, lyrics) in the different tracks. The STRF transforms can be used as an input for future algorithms of song recognition.

References

[1] Ethem Alpaydın. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning series. The MIT Press, 3 edition, 2014.

[2] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[3] Helmstaedter M Berning M, Boergens KM. Segem: Efficient image analysis for high-resolution connectomics. *Neuron*, 87(6):1193–1206, 2015.

[4] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[5] David J. Klein Shihab A. Shamma Didier A. Depireux, Jonathan Z. Simon. Spectro-temporal response field characterization with dynamic ripples in ferret primary auditory cortex. *Journal of Neurophysiology*, 85(3):1220–1234, 2001.

[6] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture Synthesis Using Convolutional Neural Networks. *ArXiv e-prints*, May 2015.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Ju Han and Bir Bhanu. Individual recognition using gait energy image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(2):316–322, February 2006.

[10] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, 2017.

[11] Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures: New directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456, 2015.

[13] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. M. Smeulders. Structured Receptive Fields in CNNs. *ArXiv e-prints*, May 2016.

[14] A. Kale, A. Sundaresan, A. N. Rajagopalan, N. P. Cuntoor, A. K. Roy-Chowdhury, V. Kruger, and R. Chellappa. Identification of humans using gait. *IEEE Transactions on Image Processing*, 13(9):1163–1173, Sept 2004.

[15] D Kerr, TM McGinnity, S Coleman, and M Clogenson. A biologically inspired spiking model of visual processing for image feature detection. *Neurocomputing*, 158:268–280, 2015.

[16] Mark Sandler Keunwoo Choi, Gy¨orgy Fazekas. In *Explaining Deep Convolutional Neural Networks on music classification*, 2016.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[18] Matthias Bethge Leon A. Gatys, Alexander S. Ecker. In *Image Style Transfer Using Convolutional Neural Networks*, pages 2414–2423. IEEE Computer Vision Foundation, 2016.

[19] Matthias Bethge Leon A. Gatys, Alexander S. Ecker. A neural algorithm of artistic style. *arXiv:1508.06576*, August, 2015.

[20] Lihua Li. Audio musical genre classification using convolutional neural networks and pitch and tempo transformations. 2010.

[21] Li Zhaoping Lingyun Zhao. Understanding auditory spectro-temporal receptive fields and their changes with input statistics by efficient coding principles. *PLoS Comput Biol*, 7(8):e1002123, August 18, 2011.

[22] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Mpca: Multilinear principal component analysis of tensor objects. *IEEE Transactions on Neural Networks*, 19(1):18–39, Jan 2008.

[23] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4898–4906. Curran Associates, Inc., 2016.

[24] J.G. Makin. Backpropagation. 2006.

[25] Mario Baroni Mariateresa Storino, Rossana Dalmonte. An investigation on the perception of musical style. *Music Perception: An Interdisciplinary Journal*, 24(5):417–432, 2007.

[26] K. O'Shea and R. Nash. An Introduction to Convolutional Neural Networks. *ArXiv e-prints*, November 2015.

[27] S. Ravi, D. P. Mankame, and S. Nayeem. Face recognition using pca and lda: Analysis and comparison. In *Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013)*, pages 6–16, Sept 2013.

[28] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[29] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[30] Shamma Shihab. Spatial and temporal processing in central auditory networks. *Methods in neuronal modeling*, pages 247–289, 1989.

[31] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. *arXiv preprint arXiv:1504.04658*, 2015.

[32] Bob L. Sturm. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *CoRR*, abs/1306.1461, 2013.

[33] Powen Ru Taishih Chi and Shihab A. Shammac. Multiresolution spectrotemporal analysis of complex sounds. *Acoustical Society of America*, 118(2):887–906, 2005.

[34] Steven K. Tjoa and K. J. Ray Liu. Musical instrument recognition using biologically inspired filtering of temporal dictionary atoms. In *ISMIR*, 2010.

[35] George Tzanetakis and Perry Cook. Multiscal multirate spectro-temporal auditory model. *IEEE Transactions on speech and audio*, 10(5):293–302, 2002.

[36] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio*, 10(5):293–302, 2002.

[37] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. *ArXiv e-prints*, March 2016.

[38] I. Ustyuzhaninov, W. Brendel, L. A. Gatys, and M. Bethge. Texture Synthesis Using Shallow Convolutional Networks with Random Filters. *ArXiv e-prints*, May 2016.

[39] Manjunath Kudlur Vincent Dumoulin, Jonathon Shlens. A learned representation for artistic style. 2017.

[40] Abhinav Gupta Xiaolong Wang. Generative image modeling using style and structure adversarial networks. 2016.

[41] Angela D. Friedericia Yun Nana, Thomas R. Knöschea. The perception of musical phrase structure: A cross-cultural erp study. *Brain Research*, (1094):179—-191, 2006.