**INTRODUCTION**

The project aims at developing a tool that allows a user to retrieve the key of a musical piece automatically. Our goal is to use classical methods from Machine Learning to perform an important task in the MIR (Musical Information Retrieval) domain. MIR researcher, and also musicians, are interested for a long time in this task and a lot of different methods have been tested. However, most of the research focuses on Signal Processing algorithms which involves musical and scientific expertise. In the past few years, Deep Learning methods have entered the field of MIR and some researchers have tried to apply them to Key Estimation. Surprisingly enough, we found little research with “simple” Machine Learning method that tried to perform this task end-to-end.

We programmed 2 methods : a fully connected Neural Network and a CNN. The goal is to compare the results obtained with both and to see if we could achieve an end-to-end classifier with any audio in input and a key in output.

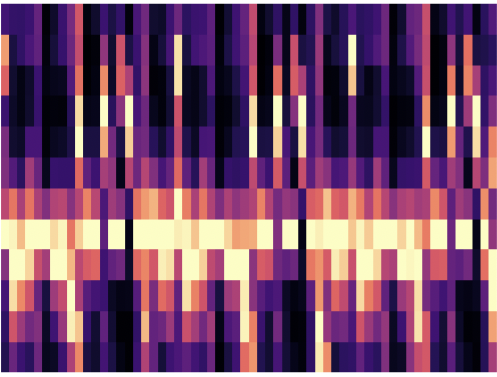
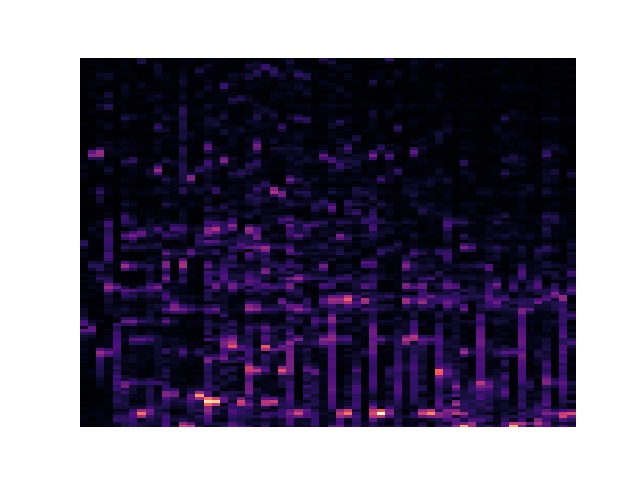
DATASETS AND PREPROCESSING

### *Datasets*

Difficulty of machine learning key-estimation is that each genre has a different relation to the key and the frequencies hit are different between genres. Estimating the key of an acoustic guitar song is different than for an EDM song. Also, classical music doesn't have a single key for the whole duration of the song. For developing purposes, we chose to focus on songs having a single key and our method will focus on EDM-genre data : Giantsteps (604 audios), Giantsteps MTG (1407 usable audios).

We decided to keep only a small part of each audio file because it is not necessary to have the whole song to detect its global key. Most of the musical information can be contained within any 10 to 30 seconds of a song which is enough to conduct our experiments. Also, keeping an exert is going to improve the speed of the training. Thus, the input is cut for a random 10sc in the Neural Network and 20sc for the CNN.

## *Features preprocessing*

To analyze the key of the songs we chose to process the raw audios before applying any Machine Learning method. Key Estimation is a pitch related problem (and pitch is a logarithmic function of the frequency) so we processed the audios to reveal their frequency related features. MIR research often cites chromagrams as being insightful features extractor as well as a light data structure. Chromagrams are matrices which represent the quantity of any pitch (from C to B in the European system) in a particular time frame. We programmed a function that performs the transformation on the exert and turns it to a chromagram. We go from a 10 seconds audio to a 12\*60 matrix of float values, 60 for the number of time frames, and 12 for the number of pitches. However, chromagrams are criticized because the low resolution undermines the precision of further processing and estimations. Because of that, we chose to compute another transformation: Q-Transform.

*Spectrogram of an audio*

*(x-coordinate : time, y-coordinate : frequency)*

*Chromagram of an audio*

*(x-coordinate : time, y-coordinate : pitch)*

Q-Transform is a type of Fourier transform that gives a spectrum linearly spaced with respect to the pitches. We go from an audio to a 120\*60 matrix (or 120\*100 for the CNN) which keeps information on 5 octaves and a resolution of 24 points per octave which equates to 2 points for a semitone. We have now more complete information on the notes in the music but at the expense of computation time.

**METHODS**

## Dense Neural Network

We tried to estimate the key of a song with fully connected neural networks, this was the method that could lead to the best hope of generalization and was also close to the deep learning methods we’ve seen in several articles. As too many features would lead the system to have trouble training well, this is why we are working with a light structure such as a chromagram. We used sklearn to implement a Multi Layer Perceptron (MLP) to perform the classification job.

First, we automated the data preparation to create dataframes containing: the name of the audio files, the vectorized chromagram of spectrogram (vector having length of respectively 12\*60=720 or 120\*60=7200), the key of the audio (as an example "A Minor") and the key coded as an integer. The data is saved in a .csv file to be used later without having to recreate the pandas dataframe. Then, we feed the second and fourth column of the dataframe as input data to an MLP Classifier with 3 hidden layers and 24 possible outputs. The architecture of the MLP is arbitrary as no experiments with this method were known. We fixed the architecture to 3 hidden layers with respectively 100, 50, 30 neurons per layer and we use ReLU as activation function for every layer.

## Convolutional Neural Network (CNN)

The results we got with the dense network (details in RESULTS section) led us to implementing a deep learning method in order to compare with the MLP’s results and to try achieving a working version. As we did not study Deep Learning and CNNs before, we chose to implement the architecture of [1] using keras API. We chose this one and not another because it was the architecture that claimed to give the best testing results and they trained their algorithm on the same dataset than us so we could see if we can achieve at least the same results. The only difference is on the length of the input. [1] considered the whole length of an audio whereas we randomly select around 20sc of the audio. The network Is composed of 5 convolution layers with 8 5x5 filters, a dense layer which is applied on each frame of the 8\*120\*100 tensor that is the output of the convolutional layers (the dense layer is applied 100 times on each 8\*120 flattened layer), an averaging on the 100 vectors (this corresponds to a time averaging of the features, a last dense layer that output the classification. Each layer uses ReLu activation function except the last dense layer which uses Softmax.

*Architecture of the CNN*

*(output of Keras.summary())*

**RESULTS**

## *MLP qualitative results*

All preliminary experiments were made using the smallest data set (Giantsteps key). To quantify the results of our experiments, we use the categorical-cross entropy as loss and accuracy for the metric. This metric choice can be discussed (*cf* *[1] “MIREX weighted score”*), however, our goal was to evaluate if we could use this model for an end-to-end (amateur) musician friendly application. Thus, we chose to stay with this absolute metric as this type of tool would be designed to give a key to a musician who doesn’t have the knowledge to find it himself quickly. As there is little past experimentation with fully connected network for Key Estimation, a lot of the hyper-parameters are to be tuned with our direct experiments: learning rate initialization and evolution, architecture of the layers, regularization constants when we use regularization. First experiments were made with the chromagrams. However, testing different architectures, different learning rates, these preliminaries experiments led us to quickly abandon this input type as it didn’t achieve any good accuracy results even on training data : **around 25% of accuracy on training set after 400 iterations**) However, with chromagram, as the computation time is really quick (**around 7 seconds for 400 iterations on training**), we could test different architectures and fix our architecture to the one presented in the METHOD section. On preliminary experiments, Spectrograms gave more promising results. Excellent accuracy was reached on the training data but this led to hard overfitting : **95-100% accuracy on Training Data but 20% on Test Data**. What surprised us is that the training process is not too long compared to the training process with chromagrams with the same architecture. Not all the experiments done on chromagrams were done on spectrograms yet but the little done tends to show that this is a better data to work with. Excellent accuracy was reached on the training data but this led to hard overfitting (99% Training Data / 20% Test Data). We used also added some L2 regularization methods with commonly used constants to see if this overfitting could be reduced. These experiments started to look too much like random tuning and thus we decided to try preventing overfitting by systematic methods studied during class.

## *MLP quantitative results*

EVAN C TON MOMENT FRÉROT

## *CNN results*

All the results following were achieved using the biggest dataset (Giantsteps MTG key) for training, and the smallest (Giantsteps key) for testing. The results we bring about the CNN method are there in a qualitative comparison purpose. As this method goes beyond MALIS class’ content, we did not want to spend too much of our means on that. Thus, in the first implementation, we simply took the same hyper-parameters than in [1] : learning rate = 0.001, L2 regularization constant = 0.0001, batch size=64, etc. Then, we tuned the hyper-parameters in order to see qualitatively if we could have some relevant impact. The results achieved with this first set of experiment gave us reassuring results. Convergence on the training data is always achieved without having any more work to do. But then, the important number to look at is the accuracy on the validation set : **every training session led to at least one epoch with an accuracy of 50%**. This was perfect to compare to the MLP. This stays below most of the classical Signal Processing algorithm and is below [1] results. However, this meant something significant for us : the model would be predicting the correct key half the time on data it has never seen (provided data is of the correct genre). This is already better than what a lot of amateur musicians can do and even if it is not enough to hope for a real-world application, further work can give hope for better results. Secondly, this confirmed that our Pre-Processing of data was not that bad as CNN gave better results than MLP. Furthermore, one should keep in mind that we only take 20sc of the audio to build our input (similarly to [5]) and that we achieve accuracy results on testing sets comparable to [5].

**CONCLUSION AND FUTURE WORK**

We are going to multiply the experiments on the MLP to find the most suitable parameters and try to obtain a more convenient accuracy. We are also going to implement methods to avoid overfitting as seen in the courses.

After that, we would like to try convolutional neural networks (CNN) having spectrograms as input (as it produced better results). We are expecting CNNs to be more efficient than our MLP solution.

CONTRIBUTIONS

Datasets: Mirado

Data preparation (chromagram and spectrogram + automation of data preparation): Pierre and Mirado

Neural Network: Pierre

REFERENCES

[1] End-to-End Musical Key Estimation Using a Convolutional Neural Network : https://arxiv.org/pdf/1706.02921.pdf

[2] AN END-TO-END MACHINE LEARNING SYSTEM FOR HARMONIC ANALYSIS OF MUSIC : https://arxiv.org/pdf/1107.4969.pdf

[3] A FULLY CONVOLUTIONAL DEEP AUDITORY MODEL FOR MUSICAL CHORD RECOGNITION : https://arxiv.org/pdf/1612.05082.pdf

[4] FEATURE LEARNING FOR CHORD RECOGNITION: THE DEEP CHROMA EXTRACTOR : https://arxiv.org/pdf/1612.05065.pdf

[5] Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters : https://arxiv.org/pdf/1903.10839.pdf

[6] GENRE-AGNOSTIC KEY CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS : https://arxiv.org/pdf/1808.05340.pdf

IRCAM CHROMA BASED ESTIMATION OF MUSICAL KEY : http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters\_2006\_ISMIR\_KeyHPS\_p pt.pdf

Constant Q transform : https://core.ac.uk/download/pdf/144846462.pdf

Method Estimation learning rate : https://towardsdatascience.com/estimating-optimal- learning-rate-for-a-deep-neural-network-ce32f2556ce0

Fast Fourier Transform and Constant Q Explanation https://www.youtube.com/watch?v=Cl-m4X3rwac

Preprocessing audio data for Deep Learning https://www.youtube.com/watch?v=Oa\_d- zaUti8

Datasets :

https://github.com/GiantSteps/giantsteps-key-dataset

https://github.com/GiantSteps/giantsteps-mtg-key-dataset

6