# Automatic transcription using embedding latent space

Gabriel Dias Neto, Valérian Fraisse, Clément Le Moine Veillon, Guilhem Marion, Félix Rohrlich

**Abstract—**

This work adresses the problem of automatic transcription of musical audio pieces to midi sheets. We propose a method based on recent research by Dorfer et al.[1] that learns joint embedding spaces for short excerpts of musical audio and their respective counterparts in midi scores, using multimodal convolutional neural networks. The dataset is based on Piano-midi.de, a midi sheet collection of classical music played on piano and splitted by Poliner and Ellis [2006] comprising 314 midi scores from 25 composers. Finally, we define relevant measures for such projects and give propositions for model improvements.

## 1 Introduction

Automatic transcription of musical scores from a given audio file has been a difficult objective to achieve in recent decades. It's a fundamental problem in the field of Music Information Retrieval (MIR) and it's considered to be a tough problem even by expert musicians. Concretely, the problem consists in obtaining from an audio extract a symbolic musical transcription, in the form of a score, or even a midi pianoroll. Such an outcome not only plays a significant role in professional scope, but is also beneficial for amateurs to play a song they heard but which does not exist in score. Until now, the methods used to solve this problem have been largely based on audio signal processing tools such as multi-pitch detection. But more and more methods are using Deep Neural Networks to perform MIR tasks and could provide more convincing results than traditional methods. In this project, we aim to set and train a neural network that is able to perform the first step of Automatic Transcription : alignment between audio and symbolic snippets.

In more detail, we designed a new method for Automatic Music Transcription (AMT) using state-of-the-art models in music identification [1], that aims to use *Convolutional Neural Networks* (CNNs) to perform snippet identification through an common embedding space.

In this paper, we first introduce, a few state-of-the-art Automatic Transcription's methodologies as well as the functioning of *Convolutional Neural Networks* and ELU activation function, an essential bricks of our model. Then, a second part deals with the preparation of the database, which is essential for optimizing results. The next part will discuss about the architecture of the network itself. Evaluation and experimental results will then be presented, and after that a discussion and a conclusion about the project.

All our code and documentation can be found as open-source at our GitHub[1].

## 2 State of the art

### 2.1 Conventional methods for music transcription

When considering polyphonic music transcription, the main issue is to detect notes that might occur concurrently and come from several instrument sources. That's why most Automatic Music Transcription (AMT) methods are exclusively based on multi-pitch detection. Multi-pitch estimation consists in extracting notes pitches from audio features. Those multi-pitch estimation methods are divided in two categories : iterative and joint methods. Iterative methods consists in extracting the most prominent pitch at each iteration of the algorithm until no additional fundamental frequency (F0) can be estimated. The main issue is the accumulation of errors at each iteration. Joint methods consist in matching F0 combinations with the audio spectrum related to the musical piece to be transcripted. Recent developments in AMT show that the vast majority of proposed approaches now falls within the 'joint' category.

#### *2.1.1 Feature-based multi-pitch detection*

Most multi-pitch detection algorithms come directly from signal processing methods according which notes are detected through analyzing audio features. The main idea is the use of a pitch salience function which gives a criterion to detect multiple F0 in a musical piece.

1. https://github.com/GuiMarion/MultimodelEmbedding

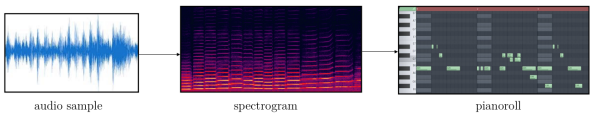


Figure 1: Multi-pitch detection and transcription process

In 2009, Yeh [7] proposed a joint pitch estimation algorithm based on a pitch candidate set score function. Given a set of pitch candidates, the overlapping partials are detected and smoothed according to the spectral smoothness principle considering that the spectral envelope of a musical tone tends to be slowly varying as a function of frequency. The pitch set score function depends on 4 features : harmonicity, mean bandwidth, spectral centroid, and "synchronicity" (synchrony).

Therefore one of our concerns is now to know which of the audio features can be used to perform F0 estimation. In 2012, the best F0 estimation method was proposed by Dressler [8]. In his method, pitch estimation is performed by identifying spectral peaks and performing pair-wise analysis on them, resulting on ranked peaks according to harmonicity, smoothness, the appearance of intermediate peaks, and harmonic number. Finally, the system tracks tones over time using an adaptive magnitude and a harmonic magnitude threshold.

Nam et al. [10] proposed a classification based approach for piano transcription using features learned from deep beliefs networks for computing a mid-level time pitch representation. Another learning based method is the one proposed by Bittner [11] in 2017 which uses a fully convolutional neural network for learning salience representations aiming at estimating fundamental frequencies. Only a little work has been done to apply deep learning methods to fundamental frequency related tasks however it tends to show that they remain the most efficient ones.

Sigtia et al. [14] explored different Neural Networks architectures to respond to the problem of music transcription applied to polyphonic piano music. Each architecture contains an acoustic model and a language model. The acoustic model they proposed is a Convolutional Neural Network used for estimating the probabilities of pitches in a frame of audio and the language model is a Recurrent Neural Network that models the correlations between pitch combinations over time. The results show that Convolutional Neural Network acoustic models yields the best performance across all evaluation metrics.

## 2.2 Convolutional Neural Networks

The CNNs have been proven to be very effective in aligninging musical representations [1] and are therefore essential in our project.

Convolutional Neural Networks (CNNs) are *"Networks that use convolution in place of general matrix multiplication in at least one of their layers"*. CNNs are useful to process data with a known grid-like topology, for example an image wich is a 2D-grid of pixels. In audio, we can feed a CNN with a 2D-matrix such as a spectrogram. A full CNN is composed of several layers[2]:

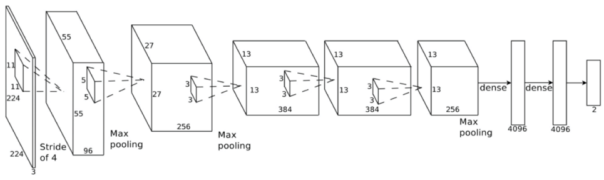


Figure 2: Famous AlexNet architecture which is a typical example of CNN [4]

According to [14], Convolutional Neural Networks are particularly adapted for our task. Indeed they can be directly applied to several frames of inputs to learn features along both, the time and the frequency axes. And using an input representation like the CQT, a Convolutional Neural Network can learn invariant features: across log-frequency and inter-harmonic spacings are constant in music signals.

### 2.2.1 Convolution Layer

A Convolutional Layer takes an input matrix and convolves $k \in \mathbb{N}$ *kernels* across the width and height of this matrix. Each kernel can be seen as a small filter. Its purpose is thus to extract features from the input matrix. The activation of one filter across every spatial position is given by the 2D activation map $h^k$:

$$h^k_{ij} = (W^k * x)_{ij} + b_k \tag{1}$$

where the weights $W_k$ and the bias $b_k$ define the filter and $*$ is the 2D discrete convolution.

After the training process, the network will be able to recognize recurring features such as edges. By stacking more layers, more and more complex features will be recognized. Thus, a Convolution Layer has several advantages: it reduces the memory requirements of the model and improves its statistical efficiency. Indeed, when processing a big image, and

because kernels are much smaller than the input image, but large enough to detect meaningful features, the information is summarized and less parameters are stored than in traditional neural networks [2].

#### 2.2.2 Non Linear activation function

The best way to describe complex features in a network is by using non linear activation functions, which will introduce some non-linearities in the training process. Such functions which has been proven very efficient are the ELU (Exponential Linear Unit) [3] defined by the following expression:

$$f(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha\ (exp(x) - 1) & if\ x < 0 \end{cases} \quad (2)$$

ELU activation functions also have proven to speed up the learning compared to RELU networks with the same architecture [3].

#### 2.2.3 Pooling Layer

The purpose of the Pooling Layer is to reduce the dimensionality of each feature map by summarizing the information inside. It consists in defining a spatial neighborhood (like a NxN square window) and applying some mathematical operations to the elements in that window: for example max-pooling takes the largest element, when average-pooling takes the average of all elements in the spatial neighborhood. Pooling Layer is crucial to reduce the size of the input data. It also helps to make the representation approximately invariant to small translations of the input, improving the statistical efficiency of the network.

## 3 A multimodal midi sheet music dataset

For this work we had to build a multimodal midi sheet music dataset starting from Piano-midi.de, a midi sheet collection of classical music played on piano and splited by Poliner and Ellis [2006] including 314 midi scores from 25 composers. From this database we decline both midi score information through pianoroll representations and related constant quality transforms of *midi2audio* automatically generated .wav audio files.

Before the dual set of audio samples being generated, all the midi scores are split into snippets of several notes that seem musically relevant (for example one beat or one bar). This split is processed by using a sliding window of constant size with a k-sample-step, that means we assume overlapping over the whole dataset. First, it allows us to significantly expand our dataset, besides it leads to a least specific database (partial notes at the start or end of the snippets for example).

One can also find a generator of synthetic data in our code[2] (with the exact right shape but filled with random numbers) in order to test the behavior of the program without having to spend too much computational power.

### 3.1 Data augmentation

Data augmentation ensures that the network trains on data that is likely to be representative of a large amount of information. It is carried out via two means : transposition and sound font choice.

Transposition is executed by shifting the piano rolls by 6 semitones up or down, resulting in 12 different files for each snippet. For the sound fonts, we selected 5 piano fonts to achieve a good variety of piano sounds. The fonts were chosen to be of good quality (not too light) and realistic-sounding. Considering the midi dataset that we used, we choose to use only classical piano sounds, well-assorted with the repertoire.

This last method implies that we have several matching audio files for one midi score. However, we assume not to have more than one matching audio file for one midi score in given mini batch (see part 3.3 for more details).

### 3.2 Audio snippets computation

As seen in part 2.1, spectrogram visualisation of audio data is a really convenient way to emphasize pitch structure of an excerpt. Indeed, a similar network ([1]) showed good results in multimodal identification, with logarithmic filter bands *Short Term Fourier Transforms* (STFT). However, since either musical writing and perception are logarithmic, a spectrogram representation with a *Constant Quality factor Transform* (CQT) would fit better to musical audio data. Indeed, such a representation can assure the same frequency resolution in each of the desired logarithmic intervals. Thus, this kind of representation can enhance identification since it can overcome the usual blurring that can be seen at low frequencies in conventional STFT, as it can be seen in figure 3.

2. Our code provide a fully modular way to process data, you can change the size of the window, the step, whether you want to process on binarized (with non velocity) symbolic data or note, change the sounfonts, ...

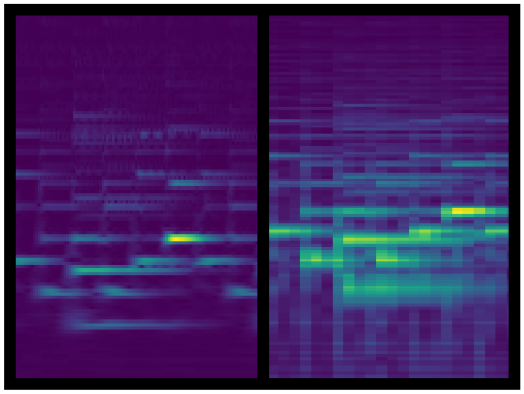

Figure 3: CQT (left) / TFCT (right, logarithmic plot) of an audio excerpt

For the dataset preparation, all the audio data were displayed into CQTs, computed with the librosa python's package [15], with 128 frequency bins in $1/16^{th}$ octave filterbands, from a minimum frequency of 30 Hertz.

### 3.3 Dataset Train/Test/validation Splits

We chose to split our whole dataset in three parts : 60% for the training set, 20% for the testing set and the last 20% for the validation set. The split is performed within each composer dataset so that we remain as unbiaised as possibe.

We also chose to represent our data as a list of batch which is represented as a tuple of lists :

$$batch = (\mathbf{x}_{score}, \mathbf{x}_{wave}, l_{score}, l_{wave}, I) \qquad (3)$$

where $\mathbf{x}_{score}$ is the list of pianorolls and $\mathbf{x}_{wave}$ is the list of CQTs. Both $l_{score}$ and $l_{wave}$ are the lists of names with respect to element index in $x_{score}$ and $x_{wave}$ respectively. $I$ is a coding optimization list that contains mixed indices so that $\forall\ k \in \mathbb{N},\ 0 \leq k \leq 31$ , $x_{score}[I[k]]$ corresponds to $x_{wave}[k]$

### 3.4 Batches construction

A batch of size $k$ is constructed as following:

1) We randomly choose $k$ symbolic snippets.
2) For each snippet, we randomly choose one corresponding audio snippet.
3) We construct two $k$-long lists of tuple $x_1$ and $x_2$ such as $x_1$ is a symbolic snippet and $x_2$ is an audio snippet that are corresponding each other.
4) We then shuffle $x_2$ so that $x_1^i$ and $x_2^i$ are not corresponding anymore. We store the associated names for $x_1$ and $x_2$ and the list of the shuffled indices of $x_2$ in order to perform matching later.

This method ensure that each snippet of the batch contains one and only one corresponding snippet, this property allow a better and more uniform matching. One can test our algorithm experimentally, by using our function $testMyBatchFunction()$ which check whether a given function returns batches which dont violate the previous property.

## 4 Training an embedding space

As we want to try to map a symbolic representation to a audio representation of a music piece, we choose to embed them into a same latent space. In order to process with better results, we do not provide audio representation to the models, but a time-frequency representation (the CQT) that we described in details earlier.

For so, lets define two models: $m_1$ is a projection from a pianoroll representation into the latent space:

$$m_1 \ : \ \{0,1\}^k \ \rightarrow \ \mathbb{R}^l$$

With $k$ the size of the pianoroll input, so $k = 128 * length$, we used time window of 4 beat size with a quantization of 24, we so have $k = 3072$. And $l$ the number of dimensions of the latent space, we followed Dorfer et al.[1] and used $l = 32$.

And $m_2$ is a projection from the CQT representation into the latent space:

$$m_2 \ : \ \mathbb{R}^k \ \rightarrow \ \mathbb{R}^l$$

In order to simplify the work of the network, we choose to represent the frequency dimension with a shape of 128, as the pianoroll. The temporal length follow the one of the pianoroll (we first align both) and is about 172 samples.

The main idea is to find $m_1$ and $m_2$ such as all different projections of $m_1$ are distinct, and that

$$\forall x_1, x_2, m_1(x_1) \approx m_2(x_2) \iff score(x_2) = x1$$

So that we are able to do a proper matching. In order to create this models, we used Convolutional Neural Networks with pyTorch.

### 4.1 Model

For the network, we used the previously cited state-of-the-art model that has been proven very efficient for symbolic/waveform matching [1]. Both $m_1$ and $m_2$ follow the same architecture, only the input shapes are differing.

| Piano Roll 128 x 96 | Audio( CQT ) 128 x 172 |
|---|---|
| 2 x Conv(3, pad-1)-24 | 2 x Conv(3, pad-1) - 24 |
| BN-ELU + MP(2) | BN-ELU + MP(2) |
| 2 x Conv(3, pad-1)-48 | 2 x Conv(3, pad-1) - 48 |
| BN-ELU + MP(2) | BN-ELU + MP(2) |
| 2 x Conv(3, pad-1)-96 | 2 x Conv(3, pad-1) - 96 |
| BN-ELU + MP(2) | BN-ELU + MP(2) |
| 2 x Conv(3, pad-1)-96 | 2 x Conv(3, pad-1) - 96 |
| BN-ELU + MP(2) | BN-ELU + MP(2) |
| Conv(1, pad-0)-32-BN- LINEAR | Conv(1, pad-0) -32-BN- LINEAR |
| Global Average Pooling | Global Average Pooling |
| Embedding Layer + Ranking Loss | |

Table 1: Architecture of the model. BN: Batch Normalization, ELU: Exponential Linear Unit, MP: Max Pooling, Conv(3, pad-1)-24: 3 x 3 convolution, 24 feature maps, padding 1

### 4.2 Training

The two models has been trained together so that there were no biases about their non-independence, to ensure, for example, that $m_1$ is not converging to a state where $m_2$ is not good anymore. In order to maximize the learning speed we used an Adam optimizer with a starting learning rate of $10^{-2}$. We decided to use a batch size of 32, first because it's a power of 2 and this fits particularly well to GPU hardware, and also because a small size of batches allow to jump out from local minima and so improve generalization [13]. We noticed that 20-25 epochs are sufficient for our data.

We also used the same loss as Dorfer et al., which is a pairwise hinge loss defined for a batch $b = (x_1, x_2, l_1, l_2, I)$ supposed non shuffled where $\forall i, x_1^i$ corresponds to $x_2^i$ [3],

$$\sum_i, \sum_{j \neq i} max(0, \alpha - s(x_1^i, x_2^i) + s(x_1^i, x_2^j))$$

with $s(x_1, x_2)$ the cosine-similarity:

$$s(x_1, x_2) = \frac{\sum_i x_1^i \cdot x_2^i}{||\sum_i x_1^i|| \cdot ||\sum_i x_2^i||}$$

Note that the loss is defined as a sum over the batches, so it will be grater when your batch size is larger, and may be quite large, so a loss smaller than 5 may be quite good in some cases.

Once the models are trained, we construct a python dictionary allowing to associate every snippet of the database with its coordinate in the latent space in order to perform matching and identification.

3. We note $x_1^i$, $x_1$ at the index $i$.

### 4.3 Over-fitting

As over-fitting is a huge problem in Machine Learning, we choose to use the test dataset only to avoid over-fitting. We claim that storing the weights of the models that minimize the loss on test dataset is what we want. For that we store smallest seen loss (initialized with a maximum acceptable value, here 50, in order to avoid unnecessary disk access). And we store the weights only if the current loss is smaller than this value.

## 5 Evaluation : audio to midi transcription

In this section, we evaluate the ability of our model to retrieve the correct counterpart when given the audio representation.

Given an audio snippet $x$ we define the transcription as a search query for the corresponding symbolic snippet. For that we embed all symbolic snippet using the model $m_1$ previously defined, and we use the model $m_2$ to embed the CQT of the audio snippet. We look for $y$, the nearest neighbor of $m_2(x)$ such as $y$ correspond to a symbolic snippet. Thanks to the previously computed dictionary that make the link between all symbolic snippet $p$ and its corresponding coordinate in the latent space $m_1(p)$, we can construct the equivalent symbolic representation. In order to process nearest neighbor, we look for the symbolic snippet $p$ that minimize the cosine-similarity $s(m_1(p), m_2(x))$. For the validation part we will use several common evaluation measures. Finally, we will use two measures that are representative of our problem of audio to symbolic transcription, including one designed especially for this project.

### 5.1 Common measures

In order to validate our method, we use some common tools used in database evaluation. For this purpose, query will be asked to return not only one result, but list of likely results, sorted by assumption of validity. We use the following tests:

- R@k — Returns the meaned number of query where the expected result is contained into the $k$ most likely results.
- MR — Returns the median rank of the expected result over all queries.
- MRR — Returns the mean of $1/rank_{target}$ over all queries.

Note that the results are highly correlated to the size of the database, if we search in a database of size

$n$ the expected value to pass R@k at random for a query is:

$$\mathbb{E}(R@k) = k/n$$

For MR,

$$\mathbb{E}(MR) = \sum_{i \in n}(i \cdot 1/n) = \frac{(n-1)}{2}$$

And for MRR,

$$\mathbb{E}(MRR) = \frac{\sum_{i \in n} 1/i}{n} = H_n/n \approx ln(n)/n$$

With $H_n$ the harmonix series.

It is clear that the result of these tests is dependant to the number of element in the database, therefore, we cannot compare results from different dataset size. In order to do so, we will always use a database with 2000 elements, this is the same number as Dorfer et al., so we will be able to compare our results.

Note also that for M@k and MR, higher the score is, better the algorithm should be, and the opposite for MRR.

### 5.2 Transcription measures

In order to suit to our specific audio to symbolic transcription goal, we add another evaluation. In the case the snippet we are looking for is not already embedded into the latent space and we want to know how far is what we return from the true result. For every element of a binarized[4] pianoroll, we define the following possibilities:

- A True Positive, is a success query where the expected value was 1.
- A True Negative, is a success query where the expected value was 0.
- A False Positive, is a failure query where the expected value was 0 (and returned value 1).
- A False Negative, is a failure query where the expected value was 1 (and returned value 0).

As our symbolic data is very sparse, we don't want to over-evaluate the similarity by taking in account a True Negative, we so propose the following score:

$$TM = \frac{\sum TP}{\sum FP + \sum FN}$$

This measure aims at measuring whether we can approximate an unknown snippet by choosing its nearest neighbor. It indicate whether:

$$\forall p_1, p_2, m_1(p_1) \approx m_1(p_2) \implies p_1 \approx p_2$$

We can also see it as the percentage of good reconstruction of a music piece.

4. Note that we can easily generalize for non-binarired, it's the case in our code.

### 5.3 Experimental Results

Here we can compare our results with the ones from Dorfer et al.

| | R@1 | R@25 | MRR | MR | TM | SG |
|---|---|---|---|---|---|---|
| Marion et al. | 0.0015 | 0.037 | 0.01 | 336 | 2% | 14%% |
| Dorfer et al. | 0.31 | 0.83 | 0.44 | 3 | ND | ND |
| Untrained | 0.0015 | 0.035 | 0.01 | 324 | 3,3% | 0% |

Table 2: Compared results

For these results we only used Mozart pieces from our database, with a window size of 1 beat and a beat-step of 2, a starting learning rate of $10^{-2}$ and a batch size of 32.

We succeed to have a loss on test dataset about 3 which seems to be pretty good, but we unexpectedly get horrible results. It's surely due to an implementation error somewhere, but we would need more time to find it.

## 6 Discussion

### 6.1 Results

Our terrible results can be explained with several arguments.

- We definitely did not have enough time to adjust our model. A bunch of parameters can be optimized, for example, we found that training with a window of size of 1 beat was improving some measures, but once the training launched, the learning rate showed to be too small for this window size (worked like a charm for 4 beats). All computation times was enormous, so that we sometimes had to wait up to several days in order to get an unsatisfying result. The final model was, for example, surely not trained completely, that should have been solved by having more time to play with the learning rate.
- As said before, the computation times were enormous, given the time limit, it was impossible to work with the full data base. Consequently, we only used the Mozart part of our data. Plus, we did not used our data augmentation part, that has been shown very positive by Dorfer et al. Using the complete database with data augmentation should drastically increase the results.
- The symbolic snippet retrieval process requires to find the nearest neighbor of the embedded symbolic snippets from a given audio snippet. However, even if this amount of snippets is huge, it is unsignificant in front of

the number of pianoroll configurations, in our case :

$$2^{128*24} \approx 10^{1000}$$

Which is way larger than the number of atoms in the universe ... It is so unreasonable to think that a given database can cover this area. In practice, that means for a given audio snippet, which does not have its associate symbolic one already embedded, we would just get the nearest without any insurance that it is actually close in terms of musical interpretation.

### 6.2 Improvements

In order to improve our results, we give an idea of future work. Given that, we can't cover the pianoroll configurations, we assume that we want to construct a symbolic representation from the latent space in place of choosing the nearest embedded neighbor. And this is what Variational Auto Encoder are rather good for ! A good idea might be to replace $m_1$ the embed symbolic representation into the latent space by VAE, we note $m_{1e}$ for the encoder and $m_{1d}$ for the decoder. For every couple $(p, a)$ in the database, where $p$ is the symbolic representation of the audio snippet $a$, we want that $m_2(a) \approx m_{1e}(p)$ (latent space matching) but also that $m_{1d}(m_{1e}(p)) \approx p$ (reconstruction). We suggest to train the three networks together and use a combined loss in order to guaranty these two properties. This technique can allow to get some interesting results on audio to symbolic transcription without having a huge database to match after the learning process.

## 7 Conclusion

In this work, we presented a new approach for musical audio to symbolic transcription using Convolutional Neural Network and joint embedded spaces. We also underlined a few weak points of this method, with suggestion of improvements. We designed a whole set of measures and tests in order to validate any improvement or modification. All code is open source, very modular and reachable from our GitHub, this project may so lead to interesting results in music transcription with few additional work.

## References


[1] Dorfer, M., Hajič, J., Jr., Arzt, A., Frostel, H., Widmer, G. (2018). *Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification.* Transactions of the International Society for Music Information Retrieval, 1(1), pp. 22–33. DOI: https://doi.org/10.5334/tismir.12

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

[3] Clevert et al (2015) *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs).* arxiv

[4] Alex Krizhevsky et al (2012) *ImageNet Classification with Deep Convolutional Neural Networks.* Advances in Neural Information Processing Systems 25, Curran Associates, Inc.

[5] A. P. Klapuri, *Multiple fundamental frequency estimation based on harmonicity and spectral smoothness.* IEEE Trans. Speech and Audio Proc., 11(6), 804–816, 2003.

[6] Pertusa, Antonio e, Jos. (2019). *Multiple Fondamental Frequency Estimation Using Gaussian Smoothness and Short Context.*

[7] Chunghsin Yeh. *Multiple Fundamental Frequency Estimation of Polyphonic Recordings.* Thèse de doctorat (2008).

[8] Karin Dressler. *Multiple fundamental frequency extraction for MIREX, 2012.* Fraunhofer Institute for Digital Media Technology IDMT, Ilmenau, Germany.

[9] Reis, G., Fonseca, N., de Vega, F.F., Ferreira, A.: *Hybrid genetic algorithm based on gene fragment competition for polyphonic music transcription.* In: Conf. Applications of Evolutionary Computing, pp. 305–314 (2008)

[10] Nam, Juhan et al. *A Classification-Based Polyphonic Piano Transcription Approach Using Learned Feature Representations.* ISMIR (2011).

[11] Bittner, Rachel M. et al. *Deep Salience Representations for F0 Estimation in Polyphonic Music.* ISMIR (2017).

[12] Jonathan Sleep: *Automatic Music Transcription with Convolutional Neural Networks using intuitive filter shapes.* In: Thesis. Presented to Faculty of California Polytechnic State University, 2017

[13] Yann Lecun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert Müller: *Efficient BackProp.* In: "Neural Networks: tricks of the trade", Springer, 1998.

[14] S. Sigtia, E. Benetos, and S. Dixon.: *An end-to-end neural network for polyphonic piano music transcription.* In: IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 24(5):927–939, 2016

[15] https://librosa.github.io/librosa/