Disentangling variation factors in audio samples

Daniel BEDOYA, Pierre-Amaury GRUMIAUX, Judy NAJNUDEL, Mathis RAIBAUD, Victor WETZEL
ATIAM: UPMC, IRCAM, Télécom ParisTech

Abstract

Our project uses the generative capacities of β-VAE framework to classify and generate new samples from the latent space. It involves the creation of a toy-dataset designed from a number of parameters that we choose, such as fundamental frequency and spectral slope. The training is unsupervised. We encountered many problems regarding the representation we chose to input in the network. This led us to implement two models: a CNN and an attention RNN. We will confront our CNN with the literature using the MNIST dataset.

1 Introduction

As the algorithms, techniques and understanding of machine learning keep improving, the task of identifying latent variables within a semi-supervised approach, has seen a great advance in the last years. The results of the β-VAE framework in learning a disentangled representation in images is undoubtedly a considerable step towards discovering factors more robustly and accurately than ever before. Yet, little is known about the applicability of these methods to audio and more specifically to music. The main goal of this project is to embrace the strengths of β-VAE and at the same time explore its potential weaknesses when working with audio samples. In order to achieve this, we apply sound synthesis techniques and implement the models seizing the capabilities of a low level machine learning framework called *Pytorch*. Our results are but a first glance at the different challenges presented in this research field and thus, much improvement is needed before they are of any practical use.

2 Designing a Toy-dataset

2.1 First attempt at creating a toy-dataset

The name "*toy-dataset*" refers to the fact that the dataset contains synthesized samples from parameters we think reflects the ground truth. They are too small to accurately represent the characteristics of real sound but they allow us to exemplify the performance of our algorithms. The purpose of the database's design is to train the net to exhibit the same parameters that we designed our database with.

2.2 Parameters of the toy-dataset

The following parameters constitute the smallest collection of independent factors that the algorithm should be able to understand in any given sound generation task. We choose them due to their musical pertinence and the possibility of ultimately generalizing these dimensions to any real sound.

- **Fundamental frequency** (f_0). It is the lowest number of cycles per second in Hertz (Hz) for any given sample. We use a very restricted range of low frequencies (from 100 Hz to 1000 Hz) with an increment of at least 50 Hz.

- **Inharmonicity** (from 0 to 1). It categorizes the degree to which the overtones are not whole multiples of the fundamental f_0. It can be understood as one of the main differences of spectra ranging from a flute (harmonic) to a bell (inharmonic).
- **Spectral slope** (from -0.1 to -0.9). Represents a linear variation of the amplitude of partials at each frequency. It is a simple parameter that is related to a sound's spectral richness.
- **Distribution of harmonics** (all, even, odd). This parameter describes the presence or absence of partials in a given sound. Examples include instruments from a clarinet (only odd harmonics) to a flute (all harmonics).
- **Signal to noise ratio** (SNR). The signal to noise ratio measures a relation between the level of the signal and the background noise in decibels (dB); it is an important component of sound that generalizes to various applications as we vary the definition of noise. For instance, this parameter would eventually help us construct percussive sounds and analyze recorded samples.
- **Decay** (from 0 to 10). This parameter emulates the extinction of a sound by an exponential decrease in its amplitude. It's an early step towards the implementation of the temporal envelope parameters ADSR.

For the first tests the spectrograms are invariant in time (Decay=0) and we test the variation of only one dimension, we discuss these results starting in section 3. The next phase of the project needs to include temporal variations to render our models more realistic. However, the addition of this new dimension entails various other difficulties.

2.3 Different levels of complexity

There are certain cases in which the chosen parameters involve a greater complexity than what is addressed in this document. We specify two of the most important and how we would proceed to minimize their effect.
The first of these complexities is brought by temporal evolution. Considering that our perception of sound (i.e. pitch and timbre) depends largely on temporal cues, variations of sound parameters in time cannot be neglected. This poses a major problem that is not present when exploiting the β-VAE framework on still images. Therefore, we originally decided to use the spectrogram representation because it contains already a representation of time. However, this intuition was not correct as we will see in section 4.
In addition, modeling the extinction of acoustical energy can also potentially increase the complexity of the model. For instance, real damping coefficients are frequency dependent and this fact alone changes the evolution of the spectral slope in time.
An ideal model would take into account the combination of all these factors while analyzing the data. Furthermore, it may even be capable of choosing different time scales depending on the input.

2.4 Audio rendering with additive synthesis

Our code contains a dedicated module that manages all the parameters and functions of the Toy Dataset (see figure 1), which contains procedural functions that work together to create it:

- The core of the module is the `parameterSpace` class, which creates an array (parameter space) of all the possible permutations from the input sound parameters defined earlier and takes their cardinal product.
- The `audioEngine` class generates the sounds and their representation (spectrogram, sound and CQT) from the parameter space dictionary of parameters.
- Then, the `toyDataset` class will create an object which can be used by *Pytorch*.
- In addition, we have utility functions that are not connected to the main module but allow the exploration of different ideas used in the process, such as the 1D convolution or Hilbert curve representation.

As for the method used to generate the dataset, we chose additive synthesis because of its flexibility. Indeed, we can work with each sound component in the form of simple sinusoids, each with its own basic properties that we can control in detail.

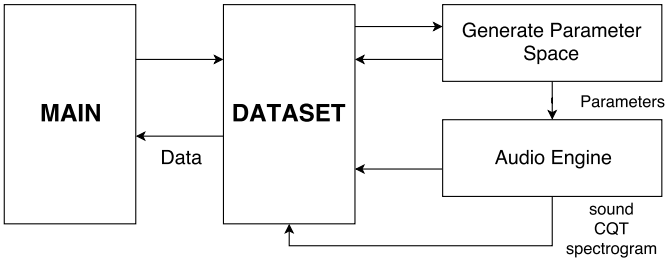


Figure 1: Flow diagram that explains the connexions of the Dataset module

Additive synthesis can be mathematically expressed as :

$$s(t) = \sum_{n=1}^{N} A_n(t) sin(2\pi f_n t + \phi_n), \tag{1}$$

where $s(t)$ represents the output signal and N, A_n f_n, ϕ_n are respectively assigned as the number of modes, the modal amplitudes, frequencies and phases. It is worth noticing that the phase at the origin is always set to zero.

In order to add some inharmonicity, we need to extend modal frequencies as :

$$f_n = n f_0 \sqrt{1 + (n^2 - 1)B} \tag{2}$$

where $B \ll 1$ is called the inharmonicity factor. Initially, A_n does not depend on time. It can be expressed as $A_n = a(n-1) + b$, where a is the spectral slope, which always behaves linearly by choice.
We can then improve the model by introducing a simple constant exponential damping coefficient α which we multiply by the signal $(s(t)e^{-\alpha t})$. The next stage is consequently to render the coefficient frequency-dependent $\alpha(t)$. Thus finally, we have at our disposal a more complete model of temporal and frequency dependency.

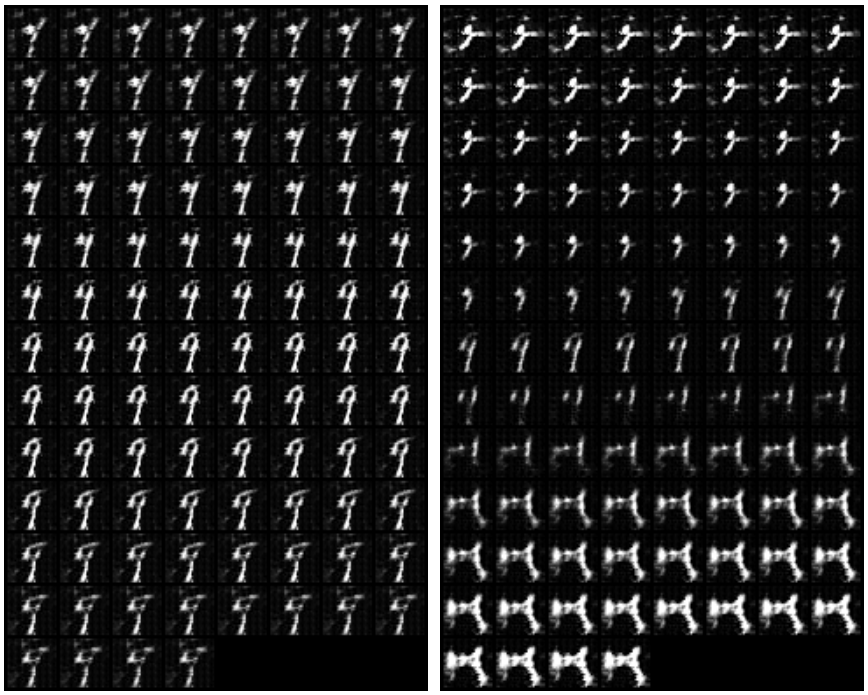
2.5 Labeling and analyzing the dataset

The labeling process could be made automatically by writing a dictionary from the parameters and their values while they are created. However, we have not labeled our dataset because the labels wouldn't be objective enough for practical purposes. We would first have to devise an rigorous way to label combined parameters for it to be applied consistently to the whole dataset. Only then, the labels would be of use to the users and the algorithms.

3 Model and expected results

Our model is a β-Variational Auto Encoder (β-VAE). VAEs have been first introduced by Kingma and Welling [2]. β-VAE, first presented by Higgins [1], are derived from it. It is a semi-supervised model that allows to had a constraint to the training process that is a compromise between: **generativity**, the capacity of the net to create some new data; the **quality of reconstruction**.
The variational lower-bound D_{KL} is modified by multiplying the KL-divergence with a β factor which represents a regularization coefficient acting on the latent factors and the learnt posterior. If $\beta = 0$, we then obtain a classic Maximum-Likelihood estimator. If $\beta = 1$, we get Bayes's solution.

This model is based on a encoder-decoder system: the input data are encoded into a latent space with latent parameters; the decoder retrieves the data from the latent space and reconstruct the data into our space. The goal is to train the framework so its latent space uses the same dimension as the parameters we designed our toy dataset with.



(a) Dimension z_0 (b) Dimension z_5

Figure 2: 20 latent dimensions, $\beta = 1$, 10 epochs

3.1 Behaviour analysis on MNIST

To understand how our model behaves, we use a well known dataset, introduced by Yann le Cun [1]. It is directly downloaded by Pytorch when you execute the script.
The VAE organizes its latent space depending on low-level features and image shape. But the reconstruction is not very good (see images in appendix, and on our GitHub site). Figure 2 shows two dimensions of the latent space that display this behaviour:

3.2 First use of the toy dataset

In the beginning, the net is fed with the spectrograms representing each sound. Each spectrogram is flattened in order to obtain a vector.
The net was not able to converge when noise is present, and thus cannot reconstruct a spectrogram that shows the presence of partials. As a result, the noise parameter (SNR) has been deleted temporarily.

The first visualization of the latent space showed that the net does not learn a disentangled representation. Figure 3 shows the reconstruction of several spectrograms drawn for the latent space. Only one dimension is varying. We see that the net displays some kind of knowledge of pitch: it begins in the low register (top of the spectrogram) and ends up in the high frequencies. However this is only an interpretation. The middle displays some kind of richness feature more than pitch. All the latent variables showed the same tendency.
We also had to ignore the distribution of harmonics parameter so that all the partials were present in every iteration.

[1] `http://yann.lecun.com/exdb/mnist/`

4 Extended model: β-VAE

We reimplemented the model to adapt it into β-VAE. This factor is a constraint pressure on the learning process.

Extended visualization To observe a structure in the latent space, we decode a set of data where only one latent variable variates. Therefore the output should display a sense of structure if the net learned something.
The results is a series of spectrograms picked from latent space. The lowest value we drew is in the top-left, whereas the highest one is in the bottom-right. Figure 3 is one of the first series we drew from the latent space.

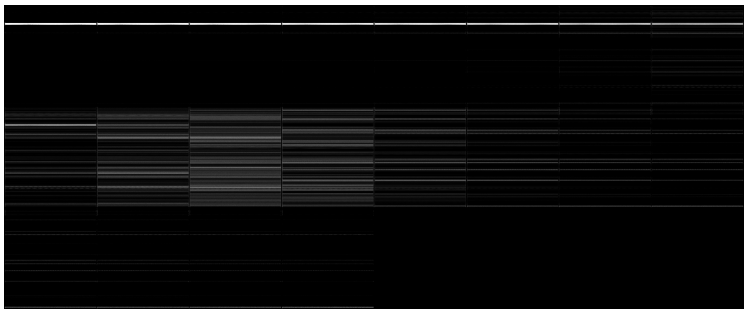


Figure 3: From top to bottom, left to right, Reconstructed spectrograms from transversal latent space: $\beta = 4$, Nfft=256, layers for the decoder: linear, ReLU, linear; decoder is linear, ReLU, linear, sigmoid

4.1 Behaviour analysis: MNIST

As β increases, the net organizes its latent space depending on the shape of the numbers, as seen on figure 4 (curved or more angled). You can see more images in the appendix and the GitHub (see `README.md`):

The best results are given by $\beta = 8$. With 20 dimensions of the latent space, some of the dimensions doesn't carry any meaning (see results on figure 5).

4.2 β-VAE behaviour on Toydataset

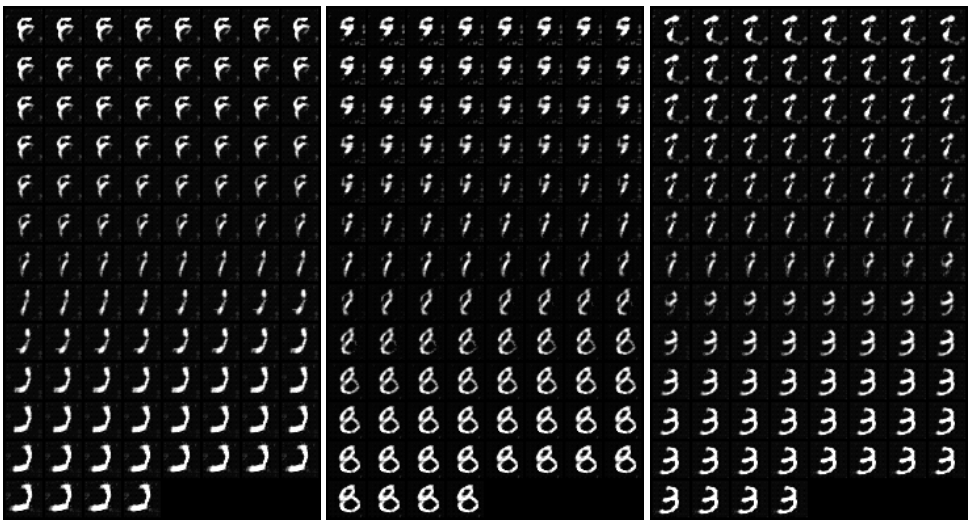
Spectrograms are not at a adapted for the algorithm. Whereas it is a great visualizing tool but it is not suited, and data needs to be more prepared than just flattened in a row vector. This transformation disrupts any kind of temporal coherence and makes it harder to evaluate the net's performance that also relies on an unspecific representation.

Solutions and ideas We use Constant Q-Transform (CQT) rather than spectrogram. The upside of this time-frequency representation is its ability to show musical features, compared to a normal spectrogram.

The net has been redesigned. We added two convolutional layers, with batch normalization, dropout, and ReLU for non-linear transformation. A linear layer has been added. The MNIST examples shown throughout the report are taken from this particular net.

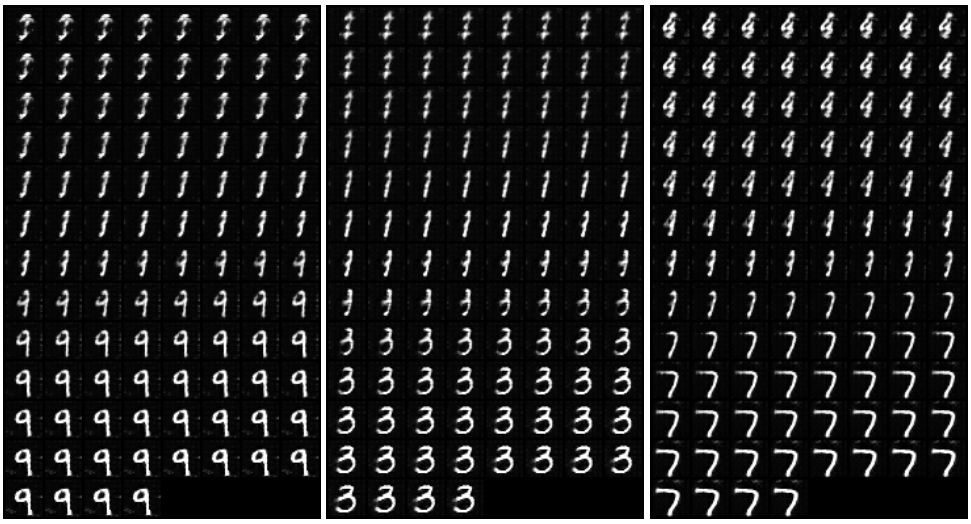
Results After fixing, the net performs very well on MNIST with very good quality reconstruction. The first epoch already shows good reconstruction. However, we observed no improvements on the toy dataset: the net reconstruct the same image all the time. We did not have time to analyse this behaviour.

We implemented an other type of net: an attention recurrent neural network that shows better results.



(a) Dimension z_7 (b) Dimension z_{10} (c) Dimension z_{12}

Figure 4: 20 latent dimensions, $\beta = 4$, 10 epochs



(a) Dimension z_9 (b) Dimension z_{16} (c) Dimension z_{17}

Figure 5: 20 latent dimensions, $\beta = 8$, 10 epochs

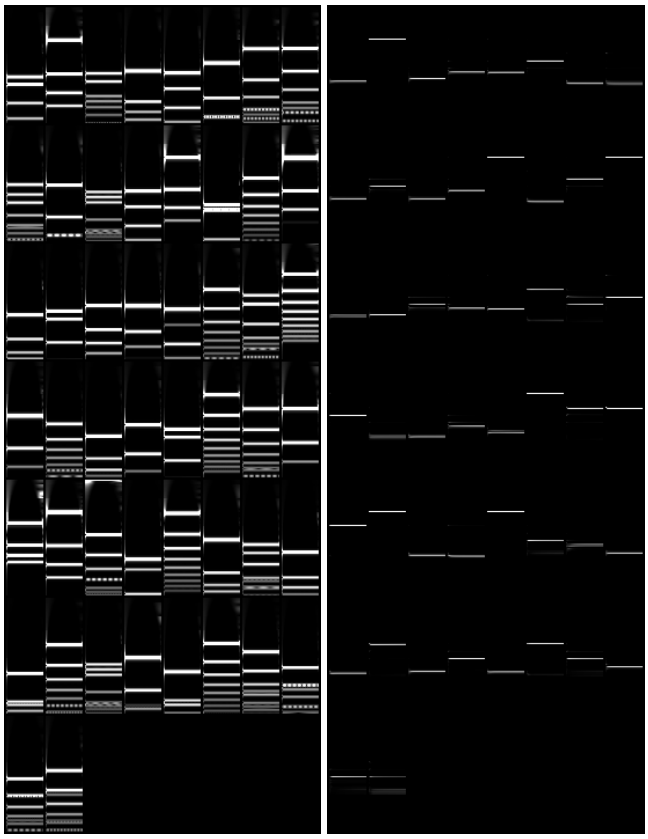
5 Recurrent Neural Network

5.1 Implementation

As sound is inherently sequential, a Recurrent Neural Network seems especially well adapted. Attention RNN in particular relies on the computation of a context vector, in order to predict the next element of a sequence from all the previous ones and highlight relevant structural patterns.
Our Pytorch version of Zafarali's model can be found in framework/modAttentiondef.py. Since data need to be arranged in a certain way to be computed by a LSTM, we also modified our main file accordingly into RNN.py.

5.2 Results

The Attention RNN is tested on CQTs. The sequence ingested by the network is thus composed of the successive windows.
Constant parameters in time (all windows are then equal), $\beta = 1$: we can see than after 5 epochs only, the fundamentals are fairly well reconstructed (fig 6) but the partials are ignored. As for the latent dimensions, we can indeed relate some of them to relevant audio factors such as spread or amplitude (fig 7) but most of them are not significant, and we did not find any dimension capturing the fundamentals.

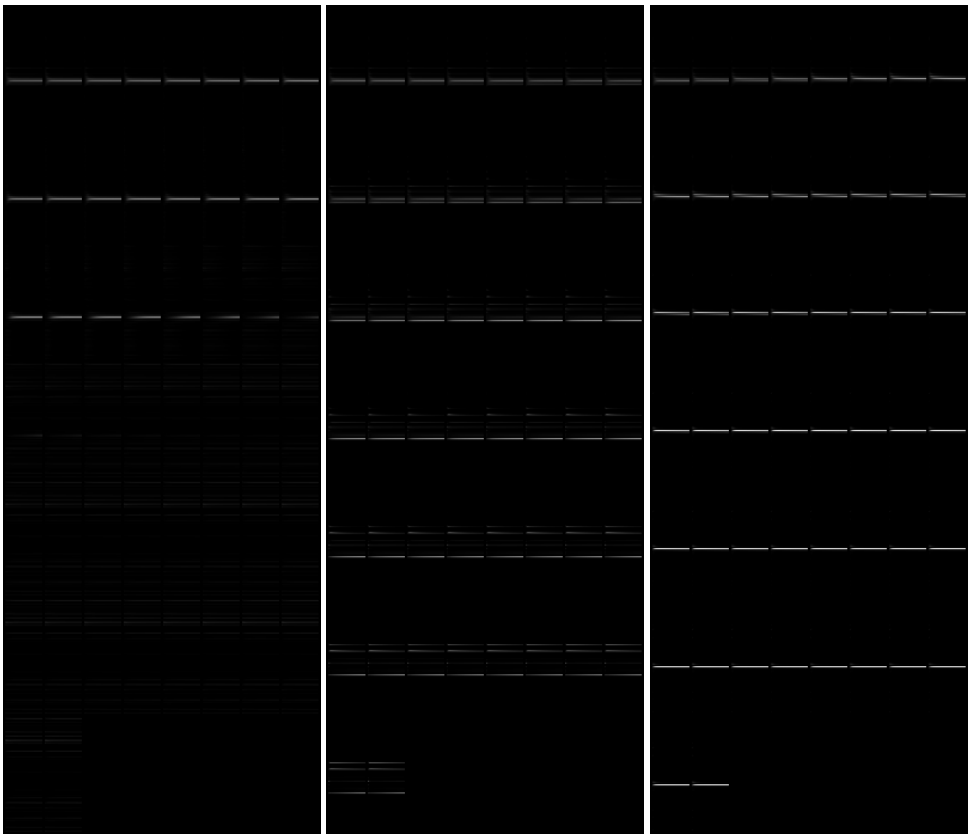


(a) original (b) reconstruction

Figure 6: 20 latent dimensions, $\beta = 1$, 5 epochs, minibatch = 50

Constant parameters in time, $\beta = 4$ and $\beta = 8$: we observe similar things, reconstruction is still acceptable for the fundamentals but few dimensions capture real audio features.

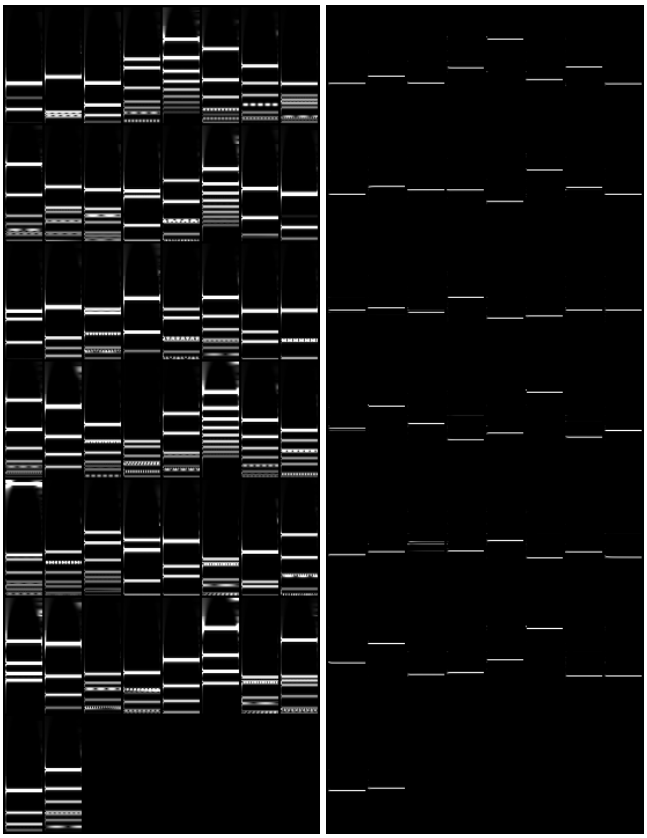
After implementing time decay in our dataset, we observe that even with a strong β (8 for instance), the reconstruction of the fundamentals is still good (fig 12). Slow decays are more easily reproduced



(a) Dimension z_{10} (b) Dimension z_{13} (c) Dimension z_{18}

Figure 7: 20 latent dimensions, $\beta = 1$, 5 epochs, minibatch = 50

than quicker ones though, it certainly has to do with the length of the windows and how quickly the context vector is computed regarding to the changes in the sequence. We still miss the partials in the reconstruction and we do not see any clear disentanglement (fig 13).
To improve our results, a lead would be the computation of the reconstruction loss. We relied on the log-likelihood of a gaussian distribution. To assess the performance of the Attention RNN further, we would have implemented other time factors in our dataset such as extinction of partials depending on their frequency or modulation. We also would have tried to implement a clockwork RNN on audio data instead of CQTs.



(a) original (b) reconstruction

Figure 8: 20 latent dimensions, $\beta = 4$, 5 epochs, minibatch = 50

Unexplored suggestions

Hilbert curve

As for flattening the 2D representation, we had the idea to use a space-filling curve: a linear representation that passes through every point in the 2-dimensional space. Hilbert's curve was a variation of the Peano curve presented in 1890. The basic idea (first order curve) is to divide the 2D plane in four quadrants and connect them using a line. As we move over to subsequent orders, we subdivide the quadrants and we map the subintervals using rotations so that we get a continuous line. This process is repeated recursively.

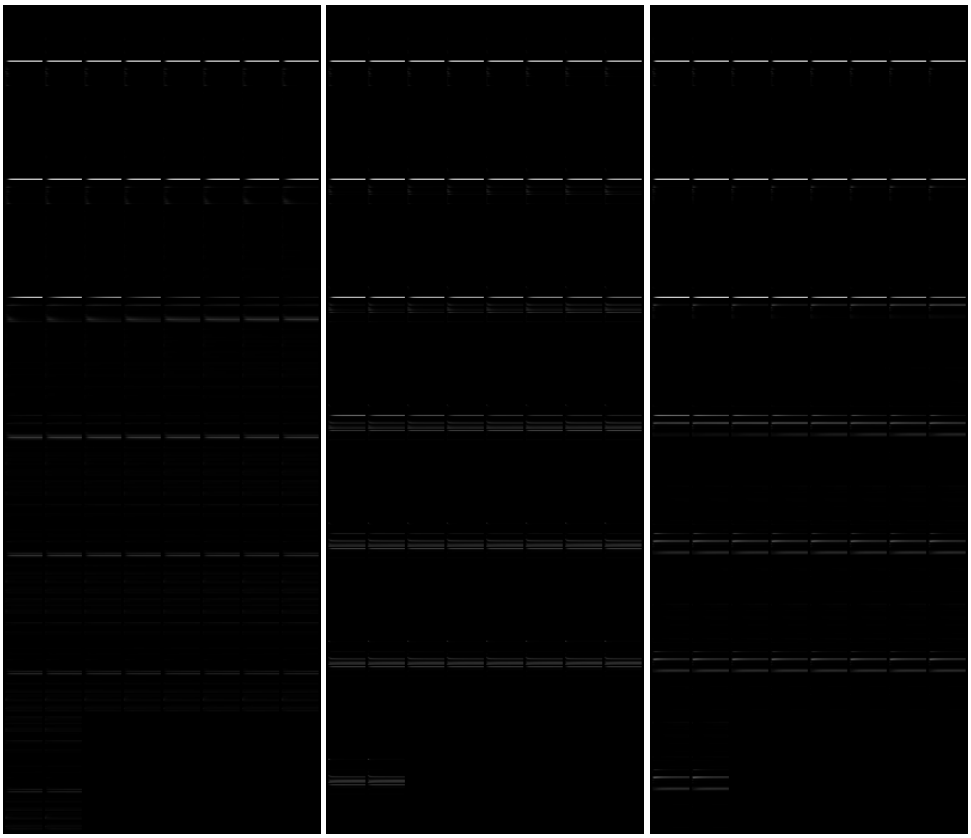
Mathematically, if we write the interval $t \in [0,1)$ in its base four (quaternary form) expansion, $t = 0_4 q_1 q_2 q_3 \ldots$, then the equation for the curve, as written by Sagan in [3], is:

$$h(t) = \sum_{j=1}^{\infty} \frac{1}{2^j} (-1)^{e_{0j}} \operatorname{sgn}(q_j) \begin{pmatrix} (1-d_j)q_j - 1 \\ 1 - d_j q_j \end{pmatrix}, \tag{3}$$

where e_{kj} denotes the number of ks preceding q_j and $d_j = e_{0j} + e_{3j} \pmod 2$.

We were interested in this approach because it allows us to maintain the spatial coherence of the data points in 2D space as they are represented by a vector. This means that we would keep the relative representation of each point in space with different sizes of matrices (see fig. 15). This in turn, could have been useful to input a more realistic representation of the data in the training process of the β-VAE algorithm, as opposed to the regular flattening using the `reshape` function, which just copies each row next to each other to create the 1D vector.

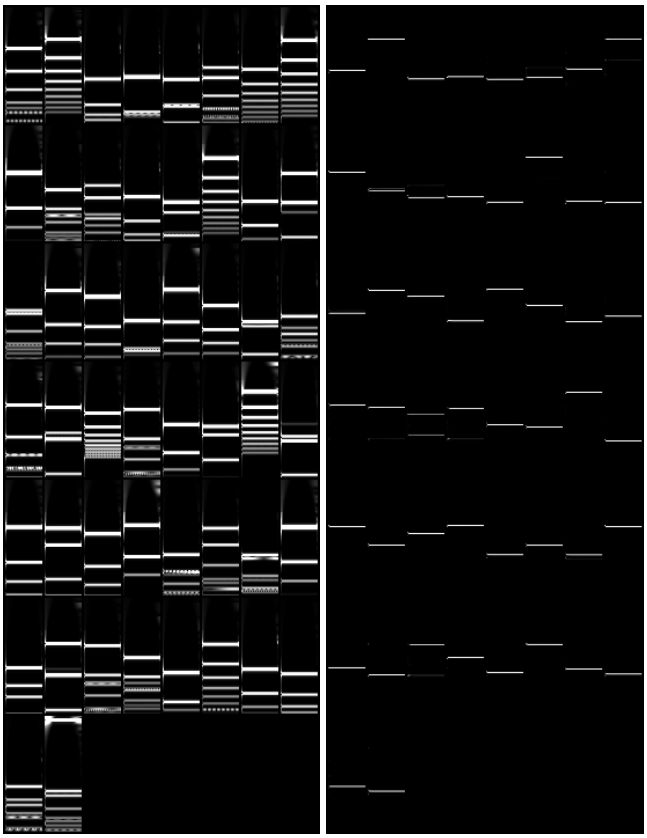
However, because of time constraints and the higher complexities involved in coding the respective functions (the original function is only defined for a $n \times n$ square plane), we weren't able to test the



(a) Dimension z_{10} (b) Dimension z_{12} (c) Dimension z_{17}

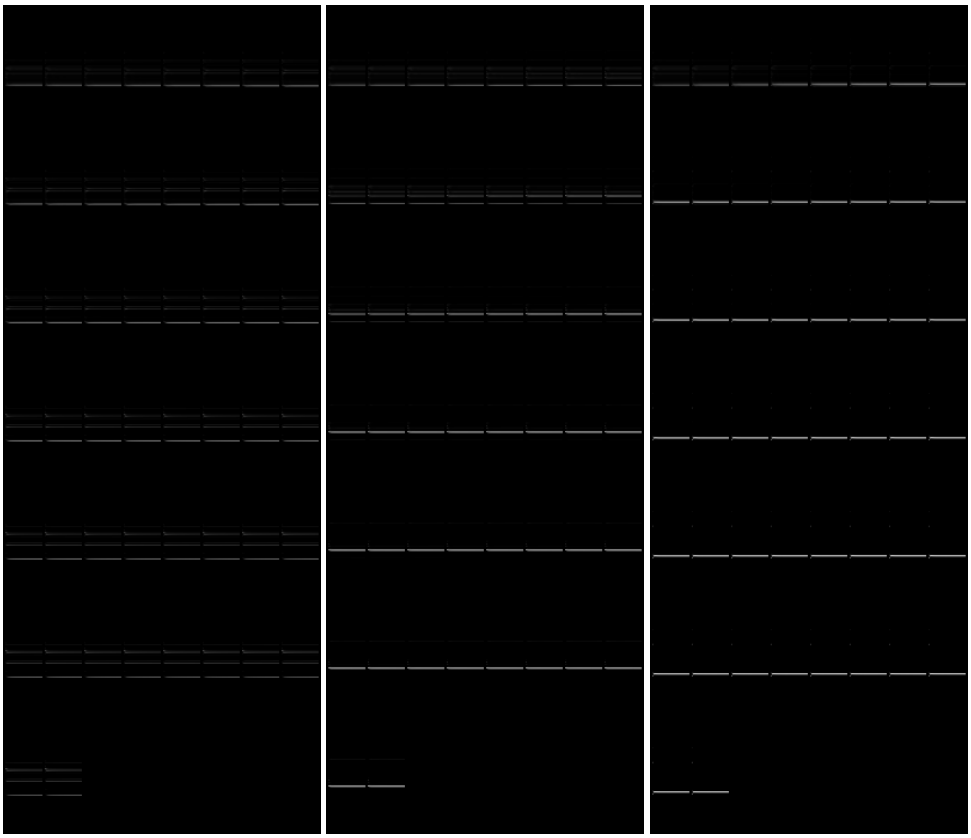
Figure 9: 20 latent dimensions, $\beta = 4$, 3 epochs, minibatch = 50

Hilbert curve in the real audio data spectrograms. A future study could include the generalization algorithm for this approach, which would allow the user to use arbitrary proportions for the 2D space by interleaving the intermediate points, thus accepting any $(m \times n)$ STFT matrix size.



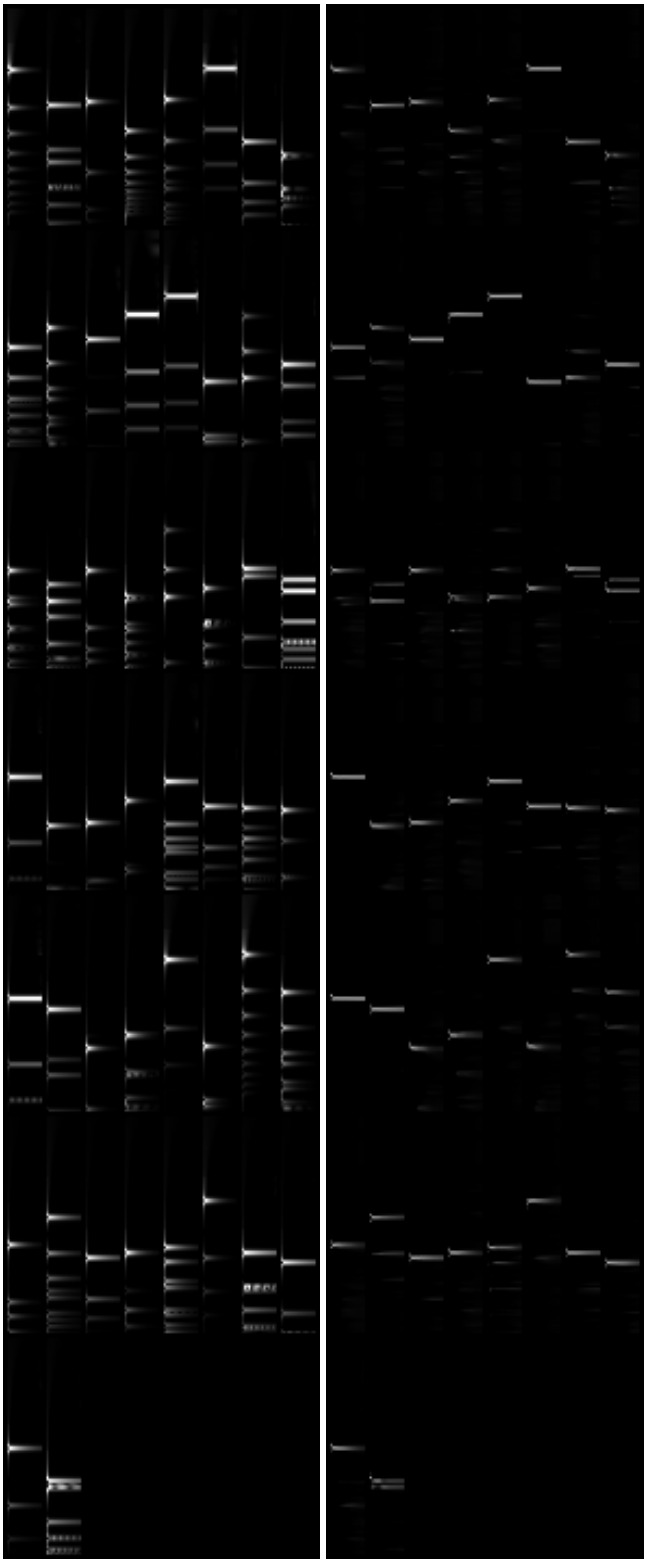
(a) original (b) reconstruction

Figure 10: 20 latent dimensions, $\beta = 8$, 10 epochs, minibatch = 50



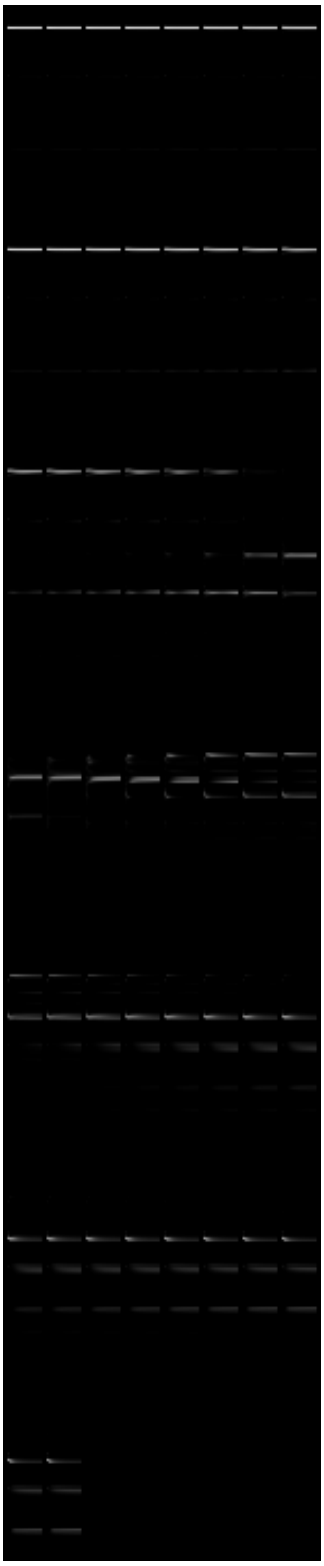
(a) Dimension z_2 (b) Dimension z_5 (c) Dimension z_{14}

Figure 11: 20 latent dimensions, $\beta = 8$, 10 epochs, minibatch = 50



(a) original (b) reconstruction

Figure 12: 20 latent dimensions, $\beta = 8$, 6 epochs, minibatch = 50



(a) Dimension z_{12}

Figure 13: 20 latent dimensions, $\beta = 8$, 6 epochs, minibatch = 50

1D convolution
Generative models based on the variational autoencoder's approach have been implemented in the first times for images. Interesting results have encouraged researchers to transfer this technique to audio files. For that reason, our preliminary intuitions were to work on 2D audio signal's representations, such as spectrograms. However, in spite of our previous results, exploring machine learning techniques on raw audio seems to be an interesting field, which is not often used in scientific papers. We also purpose to apply an 1D convolution before the β-VAE in order to give it a clearer representation of the data (in the sense that data are compressed during the convolution's operation), and to deconvolve after (fig. 14). This process is realized by the Pytorch's functions `conv1d` and `conv1dtranspose`.

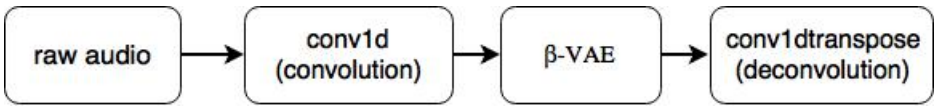


Figure 14: *convolution/deconvolution's process*

Unfortunately, the framework must build its own sound's representation, and "pure" sounds, like those we generate, lack informations. To counter this difficulty, we need to give it tools in order to build a good representation. In this way, two tools could be treated in a future work : convolution with input stride (also sometimes known as convolution with "holes") and WaveNet.

6 Conclusion

The use of spectrograms to represent the variation of a signal's frequency and amplitude over time may not be the easiest abstraction for musical machine learning. It seemed like an obvious first choice since it works instantly to inform human beings of its content but even so, the user has to be trained in order to read it correctly.
Finally, the RNN applied to the CQT representation yields the best results. It works consistently for various β values, and what is more, it is able to learn when the time decay constant is present. Even if further research is needed to fully exploit this technique, we have already looked at its potential.

APPENDIX

Hilbert Curve

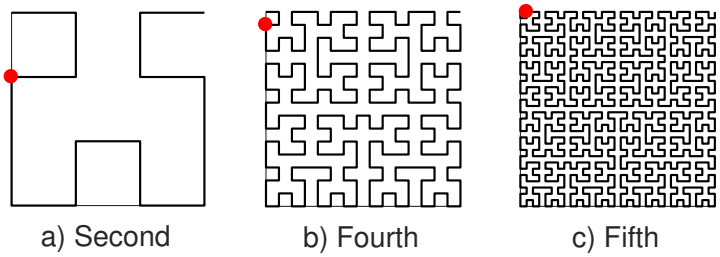


Figure 15: Three orders of the Hilbert curve. We can see that the red dot keeps its relative position in each iteration and will eventually approach a specific point in space

ACKNOWLEDGMENTS

We thank A. Bitton and P. Esling for providing assistance, jokes and beers.

References

[1] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[2] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[3] Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.