Synthesizing choir singing voices using Generative Networks

Marco Guzzo, *Ecole Nationale Supérieure de l'Électronique et ses Applications*,
Sylvain Reynal, *ETIS - CY Cergy Paris Université, ENSEA, CNRS UMR 8051*

Abstract— **Choir voice synthesis stands as a captivating domain within the realm of music synthesis, offering boundless opportunities for artistic expression and exploration. In this study, we investigate the use of autoencoder architectures for synthesising choir voice signals. In particular, we focus on the significance of probing the latent space for creative endeavours. As autoencoders are renowned for their capability to capture intricate patterns and features within data, they serve as a powerful tool in this synthesis process. Beyond the artistic field, this study may also help understanding acoustic phenomena that occur in choirs and are not well understood physically so far.**

Index Terms— **Polyphonic Audio Synthesis, Generative Networks, Musical Signal Processing**

I. Introduction

CREATING voice using Deep Learning is nothing new, and neural networks are increasingly able to generate realistic voices nowadays [1, 2, 3]. Paired with deepfake algorithms, it makes it very straightforward and ever more common to propagate misinformation. Synthesising authentic singing voice is demanding because of the various effects of vibrato and movements happening in the vocal folds. It thus requires a deeper understanding of our vocal mechanisms and internal acoustic.

With respect to speech synthesis, synthesising the singing voice already represented a quantum leap due to the paramount importance of the singer's formant, a feature characterized by an increase in signal intensity between the third and fourth formants [4]. Choir synthesis based on generative AI adds up an additional level of difficulty due to the lack of data, the struggle to parametrize a rich variety of voices and the hurdle in achieving "naturalness" in the signal produced. The latter is indeed largely dependent on modulations and interactions that are still not very documented even from a purely physical perspective. Current deep learning models never consider polyphonic groups and rather focus on a single singer, *i.e.*, *a soloist*. Deep neural networks can nonetheless extract complex structures from data. Learning group choirs' dynamic and acoustic using deep learning could help understanding the physical phenomena hidden and still very obscure.

In order to utterly understand the various layers of complexity we have here, it is mandatory to know in the first place how the voice is produced and then, how it is perceived and processed by the auditive cortex. The voice is generated by the vibration of the vocal folds induced by the passage of an airflow generated by the lungs. In addition, it involves various resonators such as the larynx, the pharynx or the mouth. Singing is adjusting those resonators in different ways.

One of the main difference between a singing and a speaking voice is the relative importance of the various formants involved in the physical process. Formants - a broad spectral maxima that results from an acoustic resonance in the human vocal tract - are very important in phonetics. Most of the time, two distinct formants can be distinguished in speech processing. For a singer, there is a predominance of a third formant [5]. Other differences exist between singing voice and speech generation: the singing voice include richer emotional information, which varies with respect to singing expression and music style. Furthermore, singing voices produce longer continuous pronunciations and contain more high-frequency parts. Finally, the different statistic distributions of singing voices become broader with large variance and, on average, pitch and phoneme-level are higher in singing voice [6].

To explain why generating choir involves a completely distinct approach, we first define what we consider as a choir in this paper. A choir is a group of people singing together. Some phenomena emerge from the group dynamic such as the *Lombard effect*, *i.e.*, the involuntary tendency of speakers to increase their vocal effort when speaking in a loud noise environment so as to enhance the audibility of their own voice [7]. Another disparity is the greater force employed by the vocalist in the singer's formant area (2–3 kHz) when in soloist mode. In choral mode on the contrary, the bass and low-mid regions exhibit greater force. While in soloist mode, the vocalists' volume also depends much less on the piano accompaniment than in choral mode, where singers adjust their volume to match that of the rest of the choir [8].

A simple crowd can also be seen as a choir. Studies on this type of groups offer an understanding of how voices tune up to bring harmony out of chaos. An ordinary, *i.e*, randomly picked person, has a limited ability to sing a melody in tune and is usually a lousy vocalist. Even when each individual vocalist is out of tune, it turns out that, when a large number of singers get together to perform as a group, the crowd's combined melody is suddenly considered by an outside listener as flawlessly tuned, as if it were a choral performance. When a group of individuals start singing together on their own at a concert or any other social gathering, it is a common occurrence to witness the *wisdom of crowds* [9] effect in action. In this work, we use pitch psychoacoustic characteristics to offer a straightforward mechanical explanation of how this emergent behaviour initiates [10]. The overall tone is regarded as being

Corresponding author: marco.guzzocholet@ensea.fr
Returned: March 24, 2024
Code available on Github

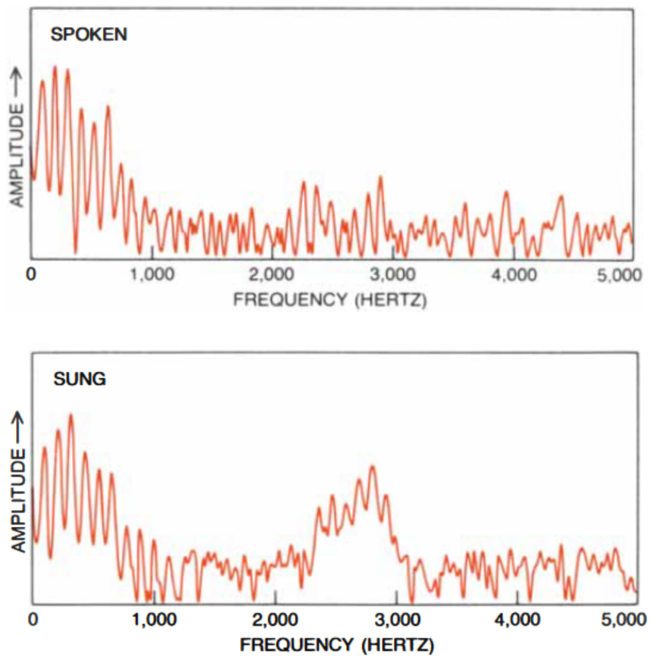

Fig. 1. Comparison between a spoken and a sung ”who’d” exposing the singing formant [4]

in tune within reasonable bounds and being independent of individual singer’s intonation. Allowing people to modify their pitch by any amount in an attempt to imperfectly mimic their neighbours, amplifies the collective effect even more. Remarkably though, this extra mechanism is not necessary for the collective effect to occur in the first place. Concisely, a crowd as a whole sings better than each individual. Thus, a form of collective harmony is generated and can be studied in the same way as collective intelligence. Nevertheless, this phenomenon has not been much studied or developed yet, be it from a physical, biological or even acoustical perspective.

II. Related Work

At Google, the Magenta team worked on a sophisticated VAE (Variational Auto-Encoder) to synthesize audio from any instrument after a long and demanding training [11]. Notwithstanding, the output of the encoder can lead to interesting discoveries about fundamental frequencies and voice envelope and else. This is the starting point of this project.

Back in the day, the most common way to estimate and represent the fundamental frequency was the pYIN algorithm (others as SWIPE or MELODIA also exists but less used). This algorithm, combination of heuristics and DSP pipelines, is that well-known and judge as efficient and accurate that it is still the one used in the LIBROSA library. Nonetheless, with the emergence of deep neural networks, other algorithms have been developed. The one used in DDSP is CREPE, the worthy heir of pYIN [12].

Pitch tracking or pitch estimation of a monophonic audio stream is a well-established area of study in audio signal processing. Pitch estimation is a crucial aspect of music signal processing, wherein monophonic pitch tracking finds use as a fundamental component of melody extraction algorithms or as a means to produce pitch annotations for multi-track datasets. In speech analysis, where prosodic elements - properties of units of speech that are defined over groups of sounds rather than single segments - like intonations may represent different parts of speech, pitch estimation is especially crucial. Pitch is not exactly a physical characteristic of the fundamental frequency; rather, it is a subjective feature of heard sounds.

To provide a pitch estimate, CREPE uses a deep 1-d convolutional neural network working directly with the time-domain audio data. In the end, we obtain a Gaussian probability density centred on the true f_0 value. It outperforms the two previous algorithms and is very accurate for different kind of instruments including voices.

The major problem with the variously designed models is that there is a great deal of variation in expressive motions and voice characteristics when singing expressively. Nevertheless, there isn’t yet a singing synthesizer that can provide realistic expressive resources like growls or harsh sounds [13]. There is a real obsession with different kind on modulations - vibrato, irregularities - that increase the sense of realism of the generated sound. Sound generated by neural networks overpass the sound quality produced by vocoders and seems more natural.

Based on the Tacotron 2 GST, Mellotron is a multispeaker voice synthesis model that can produce singing and emoting voices without the need for emotive or singing training data [14]. Mellotron can generate a wide range of speech styles, from plain speech to expressive speech, from slow drawls to rap, and from monotonous voice to singing voice, by specifically conditioning on rhythm and continuous pitch contours from an audio stream or music score. In contrast to prior approaches, we train Mellotron with read-only speech data — that is, without text-to-audio alignments. We use the LJSpeech and LibriTTS datasets to assess our models.

In another hand, XiaoiceSing uses musical scores as input [15]. Those scores contains lyrics, note pitch or note duration. Those elements are the only ones mandatory to recreate a song. This model is a modification of another model: FastSpeech [16]. A linguistic model in required to convert the lyrics into phonemes.

Because FastSpeech has been updated since, we will compare the model with the second and last version of this text to speech. The main difference between the two version is the use of *autoregressive teacher model* and especially in its ”one to many” configuration exactly as WaveNet [17]. When among the different TTS (Text To Speech) whose are not using autoregression, FastSpeech was one of the best, it suffered from several defects such as a quite complex pipeline inducing difficulties in the training phase or several losses during the mel-spectrogram conversion. To overcome this, FastSpeech 2 uses different advanced techniques unconsidered by the first version such as wavelet transform to convert the pitch contour into a pitch spectrogram which, compared to a simple Short Time Fourier Transform, gives a different approach and improve accuracy [18].

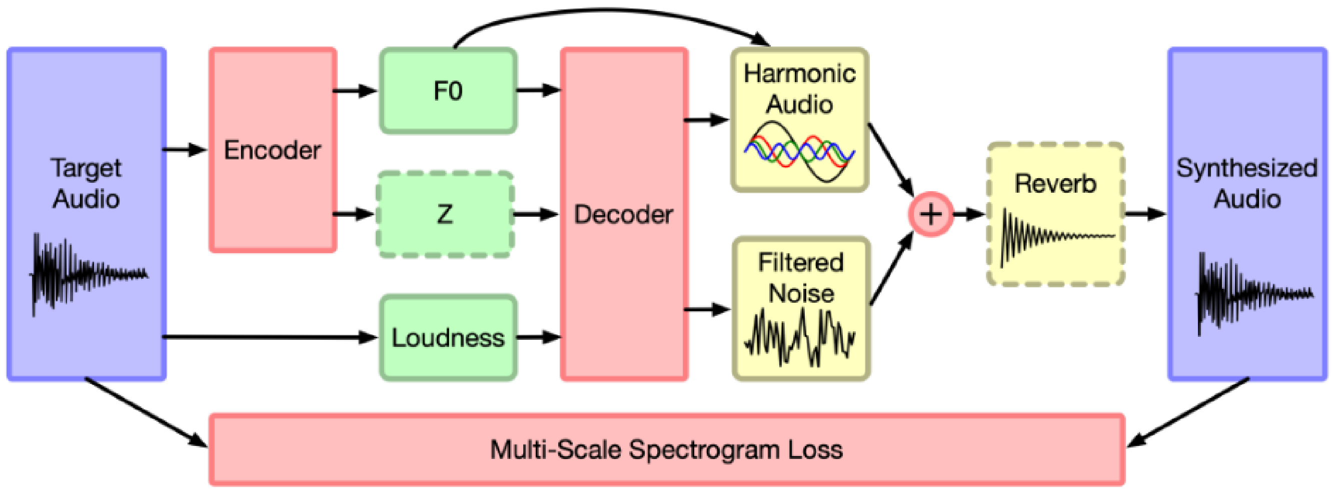

Fig. 2. DDSP process, including VAE and reconstruction [11]

III. Methodologies

A. Dataset and reconstruction method

Numerous datasets exist for a large amount of topic. Nonetheless, we have, in the music field less options and data than in the image processing field. Moreover, dealing with monophonic sources is way simpler than polyphonic ones. Some datasets exist such as ESMUC, the one we used during the first phase of this project [19], but were rejected because of poor data quality or parasite noises - several recording at the same time and reverberation of one signal on the other one on both final files. Finally, the dataset has been created using a total of 3 hours of *a capella* Gregorian choirs from copyright free sites recorded as monophonic signals with a 22050 Hz sampling rate.

The rest of this preliminary part is mainly about the choice of the preprocessing tools used for the conversion and then reconstruction of our signal. Using the documentation, the first thing coming to mind is to turn the signal into an image because capturing long-term dependencies in raw audio data would be difficult, resulting in a high computational load and slow sample generation. Nevertheless, it exists plenty of methods to do it. Which is the best choice (specifically for reconstruction) between spectrogram, mel-spectrogram or MFCC (Mel Frequency Cepstral Coefficients) ? The first argument we have to consider is do they allow good reconstruction and what could be a good metric to judge this more than just listening to the sound ?

From a pure mathematical point of view we decided to measure how far the reconstructions were from the original signals using Mean Square Error. The results can be found in the following table I.

Notwithstanding, despite its the lack of accuracy, the results are more acoustically realistic while using mel-spectrogram - because it considers the non-linearity occurring in our audio cortex, especially the cochlea. The important error in the frequency domain comes from the multiple usage of modules in the conversion processes generating an information loss on the phase.

TABLE I
Reconstruction comparison from original audio files

	MFCC	Spectrogram	Mel-spectrogram
Time domain	0.0052	0.0042	0.0070
Frequency domain	260.53	278.15	349.75

Spectrograms offer a significant advantage: they compress the time axis relative to the raw audio waveform, allowing long-term dependencies to be captured more effectively. In addition, this approach proved to be less computationally intensive.

B. Preprocessing

Before we can use the data as input for our models we consider an important preprocessing pipeline to tailor our data and turn them into usable ones for our models. To do so, we will use the Librosa Python library.

All the downloaded files are firstly loaded to obtain their amplitude value tensor. They are then merged into one after verification of their respective sampling rate. To increase the dataset size we use this aggregated file to generate a noisy and a pitched version of it. The chosen pitch modulation has been arbitrary chosen as 3 semitones (corresponding to a minor third) and the noise ratio as 1% of the signal amplitude. The added noise is Standard and Gaussian.

To ease the future process, we slice the three signals - three hour long each (238.14×10^6 samples) - into five seconds sequences and convert each of these into mel-spectrograms. The number of mels is chosen to be 256 and the hop size to be 864. In the end, we have to process data of size $(256, 128, 1)$.

C. Used models

Several deep learning models exist to generate new samples or data. Nonetheless, the most important part of this job is to understand the inherent component of our musical signal. For such job, the use of an Autoencoder is interesting. Autoencoder or its variant the Variational Autoencoder allow to reconstruct data and are less complex, time and energy consuming than other models such as GAN's or Attention Learning models based on transformers. Moreover, DDSP is using those types of models [11]. To go beyond the granularity of audio signals, the models need to be composed of several hidden layers. The models have been done using the PYTORCH and TENSORFLOW Python libraries.

The first studied model is an Autoencoder.

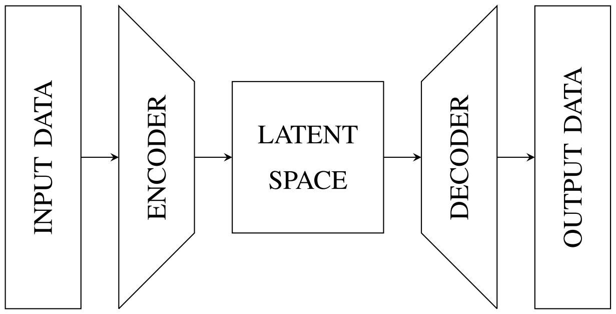

Fig. 3. Autoencoder Model

Our Encoder is composed of a succession of convolutive layers (Deep Convolutionnal Network). Each layer is composed of a CONV2D with the following parameters: square kernel with base dimension 8, stride 1, dilatation 1 and no padding ; a BATCHNORM ; a DROPOUT of parameter 0.2 and finally a RELU activation function.

The Decoder is symmetric using TRANSPOSECONV2D with same parameters and a RELU activation function.

The final model takes as input Tensors with size $(256, 128, 1)$, is composed of 5 layers for each block and is trained using a Mean Square Error (MSE) Loss between the produced image and the input image.

$$\mathcal{L}_{\text{rec}} = \text{MSE}(x, \hat{x}) = \frac{1}{256 \times 128} \sum_{\substack{1 \leq i \leq 256 \\ 1 \leq j \leq 128}} (x_{i,j} - \hat{x}_{i,j})^2$$

Nonetheless, Variational Autoencoders (VAEs) offer several significant advantages over conventional Autoencoders (AEs), particularly in addressing specific shortcomings inherent in traditional AEs.

One of the main challenges of conventional AEs is the non-symmetrical nature of their latent space around the origin. This asymmetry compromises the ability to establish a clear reference point for the generation of new samples. As a result, random sampling of points in this latent space becomes problematic, hindering the efficient transmission of these points to the decoder for accurate reconstruction.

In addition, standard AEs often exhibit disparities in the spatial representation of different labels or features. Some groupings or labels may be represented in very small areas of latent space, while others extend over much larger areas. This spatial disparity leads to an inequality in the probability of generating new samples. By sampling randomly, the probability is increased of producing points from larger areas, thus restricting the diversity of the samples generated.

Similarly, conventional AEs often suffer from noticeable discontinuities in their latent space. Between each zone or cluster of points, there are gaps, empty spaces. These gaps in the latent space create difficulties during random sampling, which can lead to the generation of poorer quality results when samples are extracted from these under-represented areas.

For such reasons we implemented a Variational Autoencoder

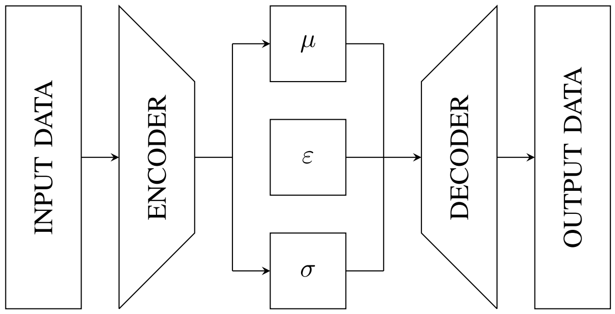

Fig. 4. Variational Autoencoder Model

For comparison purpose, the Variational Autoencoder is constructed with the same layers than the classical Autoencoder. One difference is that VAE cost function is composite. There is a double computation:

- A reconstruction part (minimisation of the error between the input data and the output data, MSE type) as for the classic auto encoder: $\mathcal{L}_{\text{recons}}$
- A measure of dispersion. This is done by calculating the divergence (Kullback-Leiber type): $\mathcal{L}_{\text{disp}}$

The cost function is therefore of the form:

$$\mathcal{L} = k_1.\mathcal{L}_{\text{recons}} + k_2.\mathcal{L}_{\text{disp}}$$

Where k_1 and k_2 are chosen so as to have a satisfactory compromise between the reconstruction and dispersion cost sub-functions.

D. Post processing

Our model takes an image as input and returns an image. Yet our work is about music and not images. We want to input audio and obtain audio. To do so, we need to transform the mel-spectrogram back into a time domain signal. During the conversion however, we have lost a very important data: the phase. Fortunately, there exists algorithms that regenerate the phase, *e.g.*, the Griffin-Lim Algorithm (GLA) [20] published in 1984 and updated in 2013 [21]. The latter demonstrates not only significant improvement in speed of convergence but does as well recover the signals with a smaller error than the traditional GLA.

GLA is a phase reconstruction method based on the redundancy of the Short-Time Fourier Transform (STFT). It promotes the consistency of a spectrogram by iterating two projections, where a spectrogram is said to be consistent when its inter-bin dependency owing to the redundancy of STFT is retained. GLA is based only on the consistency and does not take any prior knowledge about the target signal into account.

The basic algorithm is an iteration which takes the prior estimate of the complex spectrogram, applies the STFT backward and forward, and the reapplies the desired magnitude. In each back-and-forward step, information leaks between time-frequency bins taking the estimate closer to a consistent estimate. By a consistent estimate, we refer here to an estimate whose magnitude is equal to the desired level and whose phase is unchanged by the back-and-forward SFTF. Any deviation from the consistent estimate would spread out and get diluted in the back-and-forward step such that the iteration converges to a consistent state.

Algorithm 1 Griffin-Lim Algorithm

```
**Fix** the initial phase $\angle c_0$
**Initialise** $c_0 = s.e^{-j\angle c_0}$
**Iterate** for $n = 1, 2, \dots$
    $c_n = P_{\mathcal{C}_1}(P_{\mathcal{C}_2}(c_{n-1}))$
**Until convergence**
    $x^* = \mathbf{G}^\dagger c_n$
```

Where:

$$\mathcal{C}_1 = \left\{c \mid \exists x \in \mathbb{C}^L \quad \text{s.t.} \quad c = \mathbf{G}x\right\}$$

$$P_{\mathcal{C}_1}(c) = \mathbf{G}\mathbf{G}^\dagger c$$

$$\mathcal{C}_2 = \left\{c \mid \exists x \in \mathbb{C}^{MN} \quad \text{s.t.} \quad |c| = s\right\}$$

$$P_{\mathcal{C}_2}(c) = s.e^{-i\angle c}$$

Although GLA has been widely used due to its simplicity. It often involves many iterations until it converges to a certain spectrogram. This results in low reconstruction quality and high computation time. This is due to the cost function requiring only *consistency*, while the *characteristics* of the target signal are not taken into account. We can often observe distortion in the reconstructed signal. In addition, an important redundancy in the data is needed to precisely reconstruct the phase and this is not necessarily the case.

IV. Results

A. Autoencoder

With the aim of comparing its performance, we trained our model both on the original dataset and on the data augmented dataset. The training phase was set up to stop when the loss remained stable during at least 3 epochs. This method is crucial when one wants to prevent overtraining or overfitting. The corresponding evolution of the loss is represented in Fig. 5 and 6.

Ultimately we obtain a loss value lying around 15. This is convincing for images sizing up with the constrains intrinsic to spectrograms while not present in natural images. The version with data augmentation produces even better results, especially for the validation part, so that we decided to retain this last model for the testing part.

The synthesis of a new spectrogram and a comparison with the spectrogram from the original dataset is given in Fig. 7 and 8.

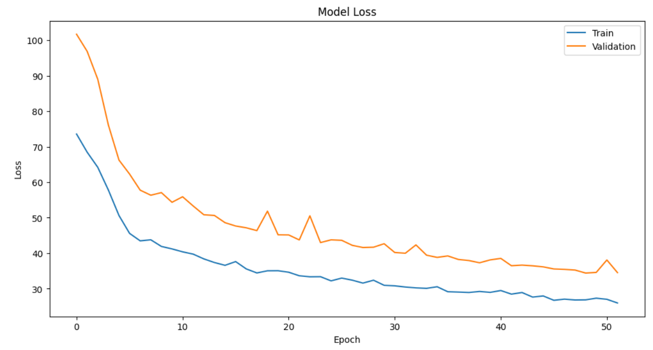

Fig. 5. Model loss evolution without data augmentation as a function of epoch (train : blue, validation : orange)

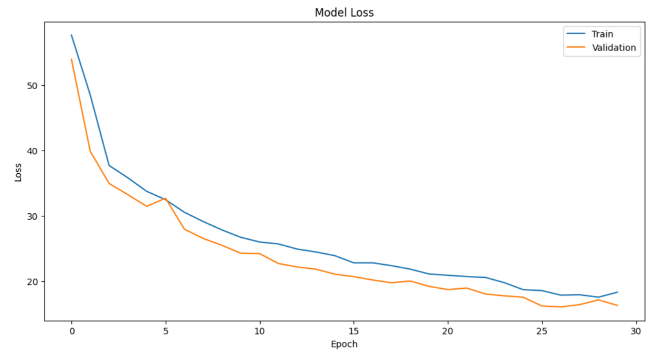

Fig. 6. Model loss evolution with data augmentation as a function of epoch (train : blue, validation : orange)

The result is close to the original with the position of the different non-zero values corresponding to the frequencies present inside the audio. The amplitudes may differ because all the audio files and input spectrograms were not normalised (different choirs or different musical segments may lead to different amplitudes).

B. Explore the latent space

If we now consider the encoding part of our model only, we can obtain - after training - the latent space of our data, *i.e.*, an abstract multi-dimensional space containing feature values that we cannot interpret directly, but which encodes a meaningful internal representation of externally observed events.

If we look back on the original paper about DDSP [11], the latent space content is composed of the fundamental frequency and residual information about the audio signal, *i.e.*, elements of the sound that are not directly related to the fundamental frequency. These residual characteristics can include various aspects such as texture, timbre and other nuances of sound, which are essential for timbre transfer and other audio manipulation applications.
Because they are using different encoders to extract distinct features and have different purposes, results may vary.

To understand what the components inside the latent space are and how they may fluctuate, we have encoded simpler signals to observe how values change. We have successively used sine, square, triangular, saw-tooth and modulated sine waves.

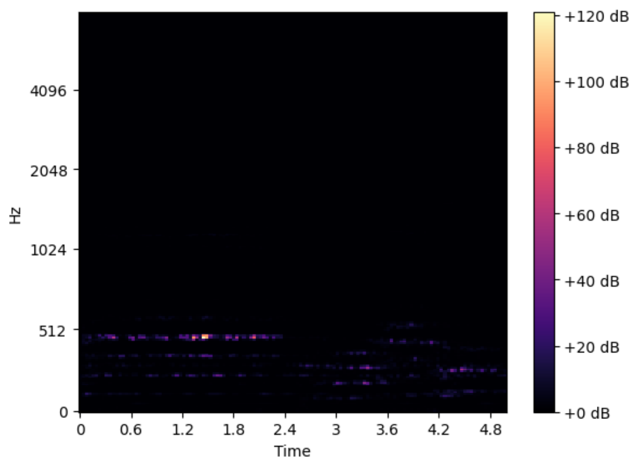

Fig. 7. Reconstructed spectrogram with AE

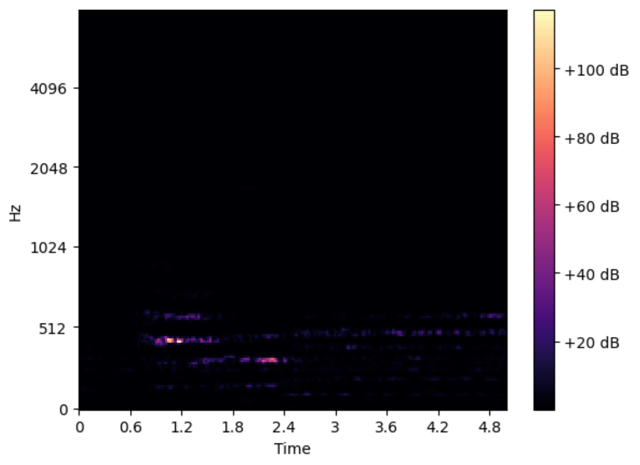

Fig. 8. Initial spectrogram

The first nodes of our model seem to learn the different coefficients of the Fourier Transform to easily understand the structure of the signal. As such, the latent space is sporadic and full of zeros when a sine wave is used. Only few elements contain non-zero values. Nonetheless, it is nearly impossible to understand what contains the different elements while using artificial signals, as some of the elements may correspond to the acoustic of the room, the reverberation process, etc...

This study allows us to visualise two parts in the latent space, one that is more dedicated to the pure signal data and one around other stuff - much like DDSP - about residual characteristics or else. Nonetheless, we are not capable to say precisely and with high accuracy what does contain this part.

C. Generate new sound

As we mention in Section III-D, our initial aim was to generate sounds based on our model. But as we said above, the algorithm involved to perform this task is not perfect and this is even more blatant with data that have been highly compressed. In the end, we obtain a sound that is fairly noisy and yet contains distinguishable choir features in the background. Some filtering process have been tried out but none of them has given convincing results. The process removed some of the noise but induced a fair loss on the sound authenticity.

Here are the most plausible assumptions about the cause of that problem. Because of the high level of compression of the data, part of the signal redundancy gets removed - and as a result of the varying importance of elements in the signal, this does not occur evenly. Now, redundancy is also the core point that allows the algorithm to work flawlessly. When tried on the original signal, the algorithm works really well - only a small reverb shows up but this is fairly irrelevant at first sight - but with reconstructed data, the noise partly hides the signal (see Fig. 9 ; 10).

Moreover, because of windows overlapping, both signals do not have the same length.

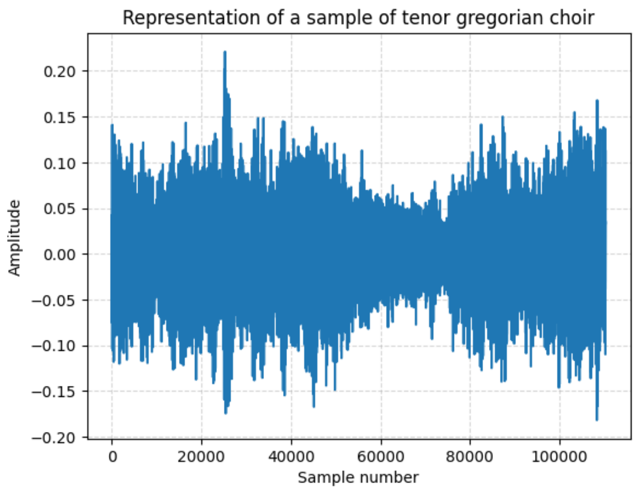

Fig. 9. Original signal

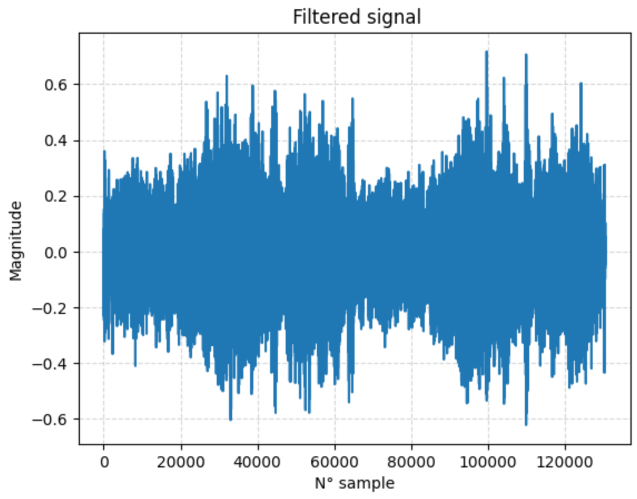

Fig. 10. Filtered output signal

V. Discussion

In this paper we tried to figure out what parameter is making a choir sound particularly harmonious and realistic when using deep-learning methods. We implemented an autoencoder to understand the inherent parts of this specific and poorly investigated audio source.

Our model is clearly efficient enough to produce spectrograms. And yet, further work should focus on inverting these spectrograms back to time-domain audio signals, particularly for data exhibiting compressed features.

A closer look at the content of the latent space when inputting several types of signals - that is, not only artificial ones, but also soloist voices whose recording follows a consistent protocol (recording in the same conditions, knowledge of the acoustics of the room, elimination of the various possible interferences) - can help understanding how the various parameters can be adjusted to produce the desired sound. It is mandatory to know the whole limit of the latent space to narrow the parameters and assure that the generated sound still have common characteristics with the original type of signal (here deep and low-pitched *a capella* Gregorian choir).

Having an artistic purpose in mind is deeply rooted in this work. Our results can prove an interesting tool for music composers. For instance, we can imagine generating new musics following the style of an author or even performing *genre-shifting*: given an audio signal we return a Gregorian choir version of it. However, we underline that ethical rules may still apply, especially regarding the dataset elements that were used to train or use the model and thus, indirectly create new music.

VI. Future work

We have made a step in this field with simple models. More time could thus be assigned to the creation of more complex and up-to-date models, *e.g.*, using transformers or attention based models. One very efficient improvement could involve removing the image-based pre- and post-processing steps, allowing us to deal directly with real audio signals.

Further experimentation could also be directed at working with soloists. We could thus compare the results of both models and extract information about how harmony emerges from the group.

VII. Acknowledgement

I thank Sylvain REYNAL for all the assistance, methodology and feedback he gave me and the ETIS lab for giving access to their offices.
I thank Victor LEVESY for his unfailing support and his invaluable help throughout this project.

References

[1] Hélène Combis. *Petite histoire de la synthèse vocale*. https://www.radiofrance.fr/franceculture/petite-histoire-de-la-synthese-vocale-6339644. Aug. 2014. (Visited on 12/28/2023).

[2] "Vocoder". In: *Wikipedia* (Dec. 2023). (Visited on 12/28/2023).

[3] Pritish Chandna et al. "A Vocoder Based Method for Singing Voice Extraction". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019, pp. 990–994. DOI: 10.1109/ICASSP.2019.8683323. (Visited on 12/28/2023).

[4] M.W. Macon et al. "A Singing Voice Synthesis System Based on Sinusoidal Modeling". In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. Munich, Germany: IEEE Comput. Soc. Press, 1997, pp. 435–438. ISBN: 978-0-8186-7919-3. DOI: 10.1109/ICASSP.1997.599668. (Visited on 10/31/2023).

[5] Johan Sundberg. "The Acoustics of the Singing Voice". In: *Scientific American* 236.3 (Mar. 1977), pp. 82–91. ISSN: 0036-8733. DOI: 10.1038/scientificamerican0377-82. (Visited on 10/31/2023).

[6] Rongjie Huang et al. *Multi-Singer: Fast Multi-Singer Singing Voice Vocoder With A Large-Scale Corpus*. Dec. 2021. arXiv: 2112.10358 [cs, eess]. (Visited on 10/31/2023).

[7] Steven Tonkinson. "The Lombard Effect in Choral Singing". In: *Journal of Voice* 8.1 (Mar. 1994), pp. 24–29. ISSN: 0892-1997. DOI: 10.1016/S0892-1997(05)80316-9. (Visited on 12/28/2023).

[8] S Ternström. "Choir Acoustics - an Overview of Scientific Research Published to Date". In: (). (Visited on 12/13/2023).

[9] Francis Galton. "Vox Populi". In: *Nature* 75 (1907), pp. 450–1. DOI: 10 . 1038 / 075450a0. (Visited on 01/15/2023).

[10] Lucas Lacasa. "Emergence of Collective Intonation in the Musical Performance of Crowds". In: *EPL (Europhysics Letters)* 115.6 (Sept. 2016), p. 68004. ISSN: 0295-5075, 1286-4854. DOI: 10 . 1209 / 0295 - 5075 / 115 / 68004. arXiv: 1608 . 01943 [nlin, physics:physics]. (Visited on 12/19/2023).

[11] Jesse Engel et al. *DDSP: Differentiable Digital Signal Processing*. Jan. 2020. arXiv: 2001.04643 [cs, eess, stat]. (Visited on 10/31/2023).

[12] Jong Wook Kim et al. *CREPE: A Convolutional Representation for Pitch Estimation*. Feb. 2018. arXiv: 1802 . 06182 [cs, eess, stat]. (Visited on 10/31/2023).

[13] Emilia Gómez et al. *Deep Learning for Singing Processing: Achievements, Challenges and Impact on Singers and Listeners*. July 2018. arXiv: 1807.03046 [cs, eess, stat]. (Visited on 10/31/2023).

[14] Rafael Valle et al. *Mellotron: Multispeaker Expressive Voice Synthesis by Conditioning on Rhythm, Pitch and Global Style Tokens*. Oct. 2019. arXiv: 1910.11997 [cs, eess]. (Visited on 10/31/2023).

[15] Peiling Lu et al. *XiaoiceSing: A High-Quality and Integrated Singing Voice Synthesis System*. June 2020. arXiv: 2006 . 06261 [cs, eess]. (Visited on 10/31/2023).

[16] Yi Ren et al. *FastSpeech: Fast, Robust and Controllable Text to Speech*. Nov. 2019. DOI: 10.48550/arXiv.1905.

09263. arXiv: 1905.09263 `[cs, eess]`. (Visited on 12/20/2023).

[17] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. Sept. 2016. arXiv: 1609.03499 `[cs]`. (Visited on 12/23/2023).

[18] Yi Ren et al. *FastSpeech 2: Fast and High-Quality End-to-End Text to Speech*. Aug. 2022. DOI: 10.48550/arXiv.2006.04558. arXiv: 2006.04558 `[cs, eess]`. (Visited on 12/17/2023).

[19] Helena Cuesta and Emilia Gómez. *ESMUC Choir Dataset*. Jan. 2022. DOI: 10.5281/zenodo.5848990.

[20] D. Griffin and Jae Lim. "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243. DOI: 10.1109/TASSP.1984.1164317. (Visited on 02/23/2024).

[21] Nathanaël Perraudin, Peter Balazs, and Peter L. Søndergaard. "A fast Griffin-Lim algorithm". In: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2013, pp. 1–4. DOI: 10.1109/WASPAA.2013.6701851. (Visited on 03/05/2024).