Report on audio restoration

Luca Murra
1920342

February 2023

1 Introduction

My project is based on the paper [3], that aims at solving different audio inverse problems: denoising, inpainting and bandwidth extension. It uses a convolutional network based on Constant-Q Transform of the audio signal for denoising and a ”reconstruction guidance” for the generative part, following from [1].

2 The network

The network works with the CQT of the signal, and it takes as input the audio signal and a parameter based on the intrinsic noise of it. Then it transforms the audio and acts on it with convolutional layers, using a lot of ”residual blocks” (RBlock in Fig.1, expanded on the right side). In these blocks, the spectrogram is filtered by some convolutional layer (8 for each block, according to the paper) and at the end a residual connection gives the output. The fact that the network works with the Constant-Q Transform and not with a Short Time Fourier Transform is critical for the success. In fact, it generates inputs that are better for a convolutional layer due to the particular position of the harmonics, that are a crucial part for a musical signal; for a voice signal those harmonics are not so relevant, so an STFT is enough.

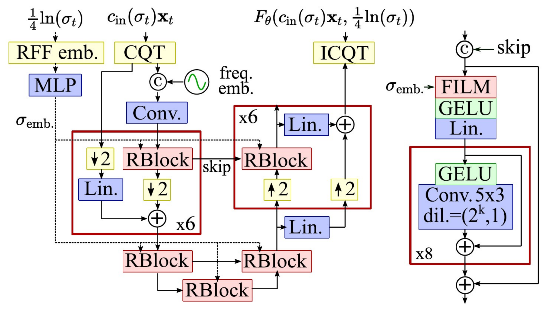

Figure 1: The CQT network

2.1 My network

Due to calculation limit, I had to shrink the network, and the residual blocks have 1 or at most 2 layers of depth, while the encoder and decoder blocks, that should be 6 copies of each, cannot be more than 2, or 3 if the RBlocks have depth of 1. In fact the combinations of encoder/decoder depth and RBlock depth that works are 1-1, 1-2, 2-2, 3-1. Since the 2-2 combination is the deepest, Colab doesn't allow the training of that for too long, so I trained the 3-1 for almost a couple of hours. The dataset that I've used is composed by the first ten sonatas of Beethoven for piano solo, played live by Daniel Barenboim. I couldn't use the dataset of the paper because of its hugeness.

3 The network output

The paper that I followed uses as target of the network a combination of audio signal and noise of it; since the input of the network is just the sum of them, the purpose of the network is to separate them. In fact, the output is:

$$F(c_{in}x_t, \sigma_{noise}) = \frac{1}{c_{out}}(y - c_{skip}(y+n))$$

where x_t is the input signal (audio and noise), y is the audio signal and n the noise; this target is taken from [2].

Then the result of the network would be

$$D(x_t, \sigma_t) = c_{skip}x_t + c_{out}F(c_{in}x_t, \sigma_{noise}) = y$$

so the signal has been denoised.

The other params that appear in the formulas are all depending on σ_t that is a random component of the noise and are defined in [2] to maintain close-to-unit variance of input and output of the network.

To visulalize the input and output of the network, we can take a look at the CQT spectri of the different signals in Fig.2.

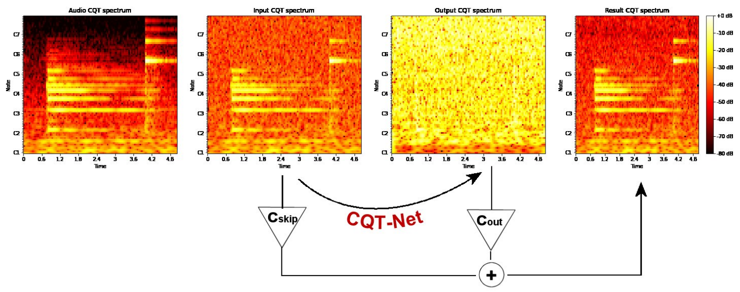

Figure 2: The network job can be seen as operations on the CQT spectrum domain. However, the operations shown are in time domain and not in frequency.

The goal of the network is to make the ”result” spectrum equal (or maybe better) than the ”audio” spectrum.

4 My results

Since I couldn’t train the whole network (for the paper results they trained the network for three days on a GPU with 80GB of RAM), I tested only the part of the denoising in a subjective manner; in fact, an objective evaluation should be done with a sort of MIDI ground truth, but for live performances of piano music it’s not so indicated. The results are far from ideal, but after a couple of hours of training there are some indications that the network does it’s job, enhancing the audio signal while reducing the noise.

References

[1] J. Ho et al. “Video diffusion models”. In: *Proc. NeurIPS* (2022).

[2] T. Karras et al. “Elucidating the design space of diffusion-based generative models”. In: *Proc. NeurIPS* (2022).

[3] E. Moliner, J. Lehtinen, and V. Välimäki. “Solving audio inverse problems with a diffusion model”. In: (2022).