UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
COMPUTER ENGINEERING:
ARTIFICIAL INTELLIGENCE AND ROBOTICS

Sound Generation using GAN Models

Laureando: Edoardo Bastianello

Relatore: Prof. Antonio Rodà

Correlatore: Prof. Sergio Targon Canazza

ANNO ACCADEMICO 2022 - 2023

Data di laurea 27/11/2023

Abstract

In recent years, there has been a noticeable increase in the use of artificial intelligence for the generation of audio files. This master thesis presents a comprehensive approach to sound generation, incorporating the utilization of multiple GAN models and post-processing techniques to generate diverse samples of different duration. In particular, some of the GAN models used include SpecGAN, Catch-A-Waveform (CAW) and UNAGAN. The proposed pipeline consists of the following components:

- SpecGAN, employed to generate one-second inharmonic samples.
- UNAGAN, used to generate variable-length harmonic samples.
- Post-processing, applied to minimize the amount of noise in the generated samples.
- Post-processing and the utilization of CAW for concatenating and inpainting multiple samples generated with SpecGAN or UNAGAN.
- CAW, which, starting from a single reference sample, is used to generate longer and cleaner samples.

Experiments were conducted using both harmonic and inharmonic sound datasets. The results demonstrated that this pipeline allows for the generation of variable-length sounds, both harmonic and inharmonic, with quality comparable to that of real samples.

Contents

List of Figures

Chapter 1

Introduction

1.1 GAN Models

In recent years, deep learning has witnessed a growing number of applications. While tasks like object detection, classification, regression, and segmentation have been well-established, generative tasks have also gained significant popularity recently. These models include Generative Adversarial Networks (GANs), variational autoencoders, Recurrent Neural Networks (RNNs), transformers, and diffusion models. In particular, this thesis focuses on GAN models.

GAN models are a specific type of network that comprises two main components:

- Generator (G): the role of this network is to produce synthetic samples that resemble those in the dataset.
- Discriminator (D): the function of this network is to distinguish between real and fake samples.

In particular, the generator learns a mapping from low-dimensional latent vectors $z \in Z$, which are points from the distribution $\mathcal{P}_z$, to points in the space of the natural data X. While training, the learned distribution gradually approximates the distribution of real data. Consequently:

- $G : Z \to X$
- $D : X \to [0, 1]$

GAN training is adversarial and grounded in game theory, with the generator and the discriminator competing to achieve opposing outcomes. In their initial

formulation[8], the following min-max function was used for training the two networks:

$$\mathbb{E}_{x\sim P_X}[\log D(x)] + \mathbb{E}_{z\sim P_Z}[\log(1 - D(G(z))] \tag{1.1}$$

This function is minimized by the generator and maximized by the discriminator. The training process is represented in Figure 1.1.

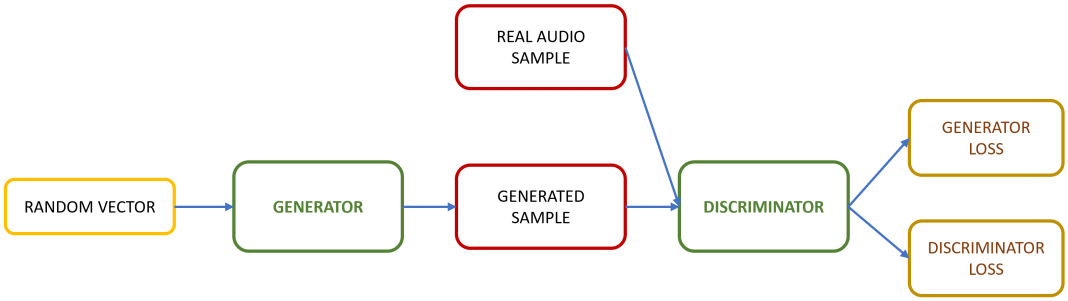

Figure 1.1: GAN adversarial training.

However, this initial formulation had several limitations, such as mode collapse[26] and training instability. As a result, to address these issues, the Wasserstein GAN (WGAN)[1] with the Wasserstein loss function led to significant improvements. The Wasserstein loss is as follows:

$$V_{\text{WGAN}}(D_w, G) = \mathbb{E}_{x\sim P_X}[D_w(x)] - \mathbb{E}_{z\sim P_Z}[D_w(G(z))] \tag{1.2}$$

This new formulation focuses on the Wasserstein distance between the real and generated data distributions rather than labeling the samples as real or fake. This change led to more stable convergence during training.

Moreover, there is also a variation of GAN models known as Conditional GAN models (cGAN), introduced for the first time in 2014[31]. While traditional GANs require no conditions, cGANs require additional input in the form of labels. This is aimed at guiding the generation process by providing more control during inference. This makes the training phase more stable and leads to faster convergence. Furthermore, the additional conditioning allows for the generation of data with specific attributes or labels. This enables cGANs to find applications in image-to-image translation, text-to-image synthesis, and style transfer.

Due to their flexibility, GAN models have found numerous applications. Some of these include generating new human faces, creating artworks, super-resolution,

face aging, converting black and white images to color images, anomaly detection, voice generation, sound generation, denoising, image restoration, simulating realistic driving environments, video generation, and video editing.

1.2 Application of GAN Models in the Audio Domain

While GANs are primarily utilized for image generation, their utilization has also expanded into the field of audio, where they have found numerous applications, including:

- Sound generation.
- Music generation.
- Audio inpainting.
- Audio upsampling.
- Music variation.
- Speech synthesis.
- Text-to-speech.
- Denoising.

Using GAN models in the audio domain provides several advantages over other models.
Autoregressive models, like WaveNet[38], are capable of modeling the local structure of audio samples but lack global structure and consistency. Moreover, they tend to be quite slow during inference.
On the other hand, GAN models can generate audio samples with both global and local consistency. Furthermore, they employ upsampling convolutions, making them significantly faster during both training and generation. In fact, unlike autoregressive models, GANs allow for parallel processing of both training and generation for the entire audio sample, resulting in a substantial increase in speed, often thousands of times faster than their competitors.

However, compared to image generation, audio generation brings more challenges. In particular, unlike images, most audio waveforms are highly periodic, which can

lead to training instability. Additionally, and more importantly, audio waveforms have a way higher dimensionality when compared to images. This high dimensionality poses a significant challenge during training, as it leads to an excessive increase in the number of parameters that must be considered.

To handle the high dimensionality of audio data, various audio representations with lower dimensionality are employed:

- Spectrograms: spectrograms provide a visual representation of how the intensity of audio files changes across different frequencies over time. Unlike the two-dimensional waveform, spectrograms are three-dimensional. To obtain a spectrogram, the Short-Time Fourier Transform is applied to the waveform. The primary advantage of using a spectrogram over a waveform is that it significantly reduces the dimensionality of the audio data. However, a notable drawback of this representation is that it's not invertible without some loss of information.

- Mel-Spectrogram: the mel-spectrogram is a variation of the spectrogram. Its main distinction is the use of the mel scale to convert the linear frequency scale found in standard spectrograms into a logarithmic scale. This adjustment places greater emphasis on the lower frequencies of the audio file, which are more relevant when considering human perception.

- MIDI: MIDI is a symbolic representation used to notate music and can also be applied in deep learning.

- Embeddings: embeddings are a representation introduced by Natural Language Processing and can be applied in the audio domain to reduce the dimensionality of audio or to generate a more interpretable representation.

Consequently, nearly every GAN model, instead of directly utilizing waveforms, opts for one of these representations, typically mel-spectrograms. The most common pipeline, reported in Figure 1.2, is as follows:

- The waveforms in the dataset are converted into mel-spectrograms.

- The model is trained to generate mel-spectrograms instead of directly generating waveforms.

- The mel-spectrograms are only inverted during the generation process.

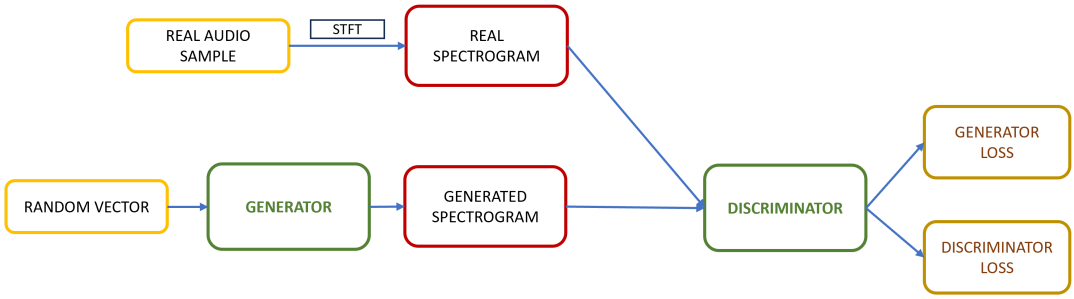

Figure 1.2: Most common GAN training process in the audio domain.

However, by generating mel-spectrograms rather than waveforms, a challenge arises in terms of inverting them back to waveforms. Mel-spectrograms, in fact, cannot be inverted without some loss of information.

The primary tools employed to invert spectrograms include:

- Griffin-Lim algorithm: this algorithm allows for an approximate inversion of spectrograms. However, it results in some information loss and may introduce artifacts.

- GAN models for spectrogram inversion: the use of GAN models specifically trained for inverting mel-spectrograms has been growing constantly. In fact, many of the most recent GAN models for unconditional generation employ models like MelGAN[22] for inverting mel-spectrograms. This approach has proven to be more effective compared to the Griffin-Lim algorithm, even though it is optimized for speech synthesis rather than music synthesis.

Evaluating a generative model is consistently a challenging task. In the audio domain, the most commonly employed evaluation metrics to assess the quality of generated samples include:

- Inception score[42]: this metric is used to evaluate the diversity of the generated samples. Specifically, it penalizes models that produce highly similar samples by employing an Inception classifier[49];

- Fréchet audio distance: this metric compares the statistical characteristics of real and synthetic samples to encourage diversity in the generated samples. To do this, it employs a pre-trained VGG-like model;

- Kernel inception distance: this metric quantifies the dissimilarity between samples drawn from real distributions and those drawn from the synthetic distribution.

Nonetheless, the most prevalent and typically more reliable metric is human evaluation. In fact, other evaluation metrics in this context are not always dependable and can sometimes yield conflicting results. Therefore, the use of surveys is the most common approach for evaluating GAN models.

1.3 The Goal of this Work

The goal of this work is to generate inharmonic sounds that match the quality of real audio files. Furthermore, for the sake of comprehensiveness, some experiments have also investigated the generation of harmonic sounds.
Additionally, this work aims to lay the foundation for a potential installation within the context of synesthesia. This installation is envisioned to immerse individuals in multisensory experiences. Consequently, this research could provide a valuable contribution to the application of GAN models in diverse sensory-enhanced and artistic settings.

Chapter 2

Related Work and State of the Art

2.1 WaveGAN and SpecGAN

In 2019, WaveGAN and SpecGAN[6] were introduced, marking the first utilization of GAN models in the field of audio. These two models can produce 1-second audio samples, yet they operate differently.

Specifically, WaveGAN directly generates waveforms, whereas SpecGAN generates spectrograms that must be then converted back into waveforms.
Both approaches have their advantages and drawbacks. In particular, due to audio signals having high temporal resolution, strategies involving waveform representation must handle high dimensionality effectively. In contrast, spectrogram representation can reduce input dimensionality, but inverting spectrograms results in information loss.

Both WaveGAN and SpecGAN are built upon DCGAN[40], an image generation model that employs transposed convolutions to progressively map low-resolution feature maps into high-resolution images.

In particular, SpecGAN uses a similar architecture to DCGAN but, instead of generating images, it generates mel-spectrograms. To convert the audio files of the dataset into spectrograms, it performs the Short-Time Fourier Transform using 16 ms windows and 8 ms strides, resulting in 128 frequency bins. Then, the magnitude of the spectrogram is logarithmically scaled to align better with human perception. Next, each frequency bin is normalized to have zero mean and unit covariance, and the spectra are clipped to 3 standard deviations and rescaled between -1 and 1.
To invert the mel-spectrograms during the generation phase, the Griffin-Lim algorithm is used with 16 iterations.

On the other hand, as WaveGAN operates in one dimension, it uses a flattened

DCGAN architecture, reported in Figures 2.1 and 2.2.

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \text{Uniform}(-1, 1)$		$(n, 100)$
Dense 1	$(100, 256d)$	$(n, 256d)$
Reshape		$(n, 16, 16d)$
ReLU		$(n, 16, 16d)$
Trans Conv1D (Stride=4)	$(25, 16d, 8d)$	$(n, 64, 8d)$
ReLU		$(n, 64, 8d)$
Trans Conv1D (Stride=4)	$(25, 8d, 4d)$	$(n, 256, 4d)$
ReLU		$(n, 256, 4d)$
Trans Conv1D (Stride=4)	$(25, 4d, 2d)$	$(n, 1024, 2d)$
ReLU		$(n, 1024, 2d)$
Trans Conv1D (Stride=4)	$(25, 2d, d)$	$(n, 4096, d)$
ReLU		$(n, 4096, d)$
Trans Conv1D (Stride=4)	$(25, d, c)$	$(n, 16384, c)$
Tanh		$(n, 16384, c)$

Figure 2.1: WaveGAN - Generator architecure. Image taken from [6].

Operation	Kernel Size	Output Shape
Input $\boldsymbol{x}$ or $G(\boldsymbol{z})$		$(n, 16384, c)$
Conv1D (Stride=4)	$(25, c, d)$	$(n, 4096, d)$
LReLU ($\alpha = 0.2$)		$(n, 4096, d)$
Phase Shuffle ($n = 2$)		$(n, 4096, d)$
Conv1D (Stride=4)	$(25, d, 2d)$	$(n, 1024, 2d)$
LReLU ($\alpha = 0.2$)		$(n, 1024, 2d)$
Phase Shuffle ($n = 2$)		$(n, 1024, 2d)$
Conv1D (Stride=4)	$(25, 2d, 4d)$	$(n, 256, 4d)$
LReLU ($\alpha = 0.2$)		$(n, 256, 4d)$
Phase Shuffle ($n = 2$)		$(n, 256, 4d)$
Conv1D (Stride=4)	$(25, 4d, 8d)$	$(n, 64, 8d)$
LReLU ($\alpha = 0.2$)		$(n, 64, 8d)$
Phase Shuffle ($n = 2$)		$(n, 64, 8d)$
Conv1D (Stride=4)	$(25, 8d, 16d)$	$(n, 16, 16d)$
LReLU ($\alpha = 0.2$)		$(n, 16, 16d)$
Reshape		$(n, 256d)$
Dense	$(256d, 1)$	$(n, 1)$

Figure 2.2: WaveGAN - Discriminator architecure. Image taken from [6].

In particular, for the generator, the 2D 5x5 filters in DCGAN are converted into 1D filters of length 25, as depicted in Figure 2.3, with the upsampling factor increased from 2 to 4 at each layer. The discriminator undergoes similar changes, using 1D filters of length 25 and an upsampling factor of 4.
Consequently, SpecGAN, WaveGAN and DCGAN have the same number of parameters and output dimensionality. Since DCGAN outputs 64x64 images, which

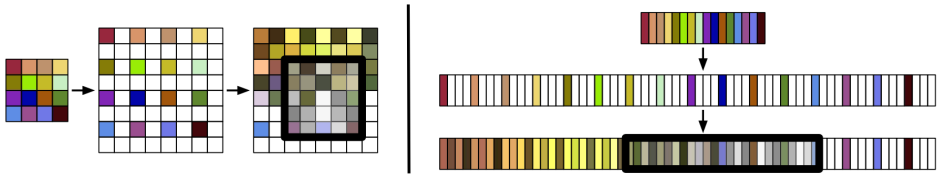

Figure 2.3: Convolution of DCGAN (left) and convolution of WaveGAN (right). Image taken from [6].

corresponds to 4096 audio samples, for WaveGAN and SpecGAN an additional layer is added to both models. This results in a total of 16384 samples, which correspond to slightly more than 1 second at a sampling rate of 16 kHz.
Both SpecGAN and WaveGAN are trained using the WGAN-GP strategy[10], which uses the Wasserstein distance for the loss function, as described in the introduction of this thesis.

However, there is one additional challenge when dealing with audio. In fact, generative image models that employ upsampling transposed convolutions, like DCGAN, produce checkerboard artifacts in the generated images. These are periodic patterns that are not very common in images, so the discriminator can learn to label images that contain them as fake and reject them. On the other hand, in the audio domain, these artifacts are perceived as pitched noise, potentially overlapping with common frequency components in the real data. The issue arises from the consistent occurrence of these artifacts at a specific phase, which can lead to the discriminator to learn to trivially reject generated samples, consequently obstructing the optimization process. To address this challenge, the researchers proposed "phase shuffling" as a solution. In particular, the phase shuffle operation is applied to the discriminator to randomly perturb the phase of each layer's activations.

To train the two models, the following datasets have been used:

- Speech Commands Zero Through Nine dataset: this is a subset of the Speech Commands Dataset[50], which contains many speaker recordings of spoken digits from "zero" to "nine".
- Drum sound effects dataset, for a total time of 0.7 hours.
- Bird vocalizations dataset, for a total of 12.2 hours.
- Piano dataset, featuring piano performances for a total of 0.3 hours.

- Large vocab speech dataset, comprising recordings from multiple speakers, with a total duration of 2.4 hours.

Both models used a batch size of 64 for the training.

Regarding the results, audio samples generated by SpecGAN achieved a higher Inception Score compared to WaveGAN (6.0 vs. 4.7) and were more accurately labeled by human evaluators.
However, WaveGAN outperformed SpecGAN in terms of sound quality and speaker diversity. These findings suggest that SpecGAN excels at capturing the variability within the training data, but the quality of its generated samples is compromised due to information loss during the inversion of the generated spectrograms using the Griffin-Lim algorithm.

In any case, both models performed better when compared with WaveNet[38] and SampleRNN[30], which failed to generate cohesive words and were notably slower during the generation phase.

In summary, both SpecGAN and WaveGAN were able to synthetize 1-second audio samples with global coherence across diverse audio domains. Additionally, they were capable of generating intelligible words when trained on a limited vocabulary dataset. Furthermore, good results were achieved even after just one hour of training.

2.2 DrumGAN

DrumGAN[35] is a model designed for unconditional generation of drum sounds and is built upon the principles of Progressive Growing of GANs[17].

More specifically, it is conditioned on the following features:

- Brightness.
- Hardness.
- Depth.
- Roughness.
- Boominess.
- Warmth.
- Sharpness.

Its architecture, reported in Figure 2.4, consists of a stack of convolutional and block up-sampling blocks, which are used to generate the signal starting from a random vector consisting of 128 components sampled from an independent Gaussian distribution.

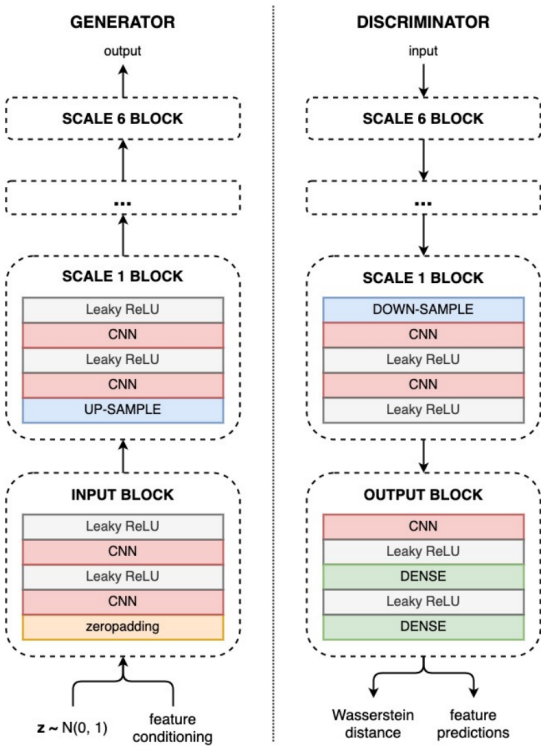

Figure 2.4: DrumGAN architecture. Image taken from [35].

For the loss function, the Wasserstein distance is utilized, in conjunction with an auxiliary Mean Squared Error term, to encourage the effective utilization of conditioning.

DrumGAN has achieved results that are comparable to the state-of-the-art work in the field of drum synthesis.

2.3 GANSynth

GANSynth[7] is built upon the Progressive GAN architecture[17] and is employed to generate individual musical notes lasting 4 seconds.

Specifically, it initiates the generation process with a single vector sampled from a spherical Gaussian distribution, which is subsequently upsampled incrementally through convolutions to produce log-magnitude spectrograms.

This model is conditioned on both pitch and timbre and has been trained on the NSynth dataset, which comprises 300,000 notes played on various instruments.

GANSynth has achieved remarkably impressive results, surpassing those obtained with WaveGAN.

2.4 TiF-GAN

TiFGAN[28] is a model designed for the unconditional generation of 1-second audio samples, and it is built upon the foundation of SpecGAN.

Its primary objective is to enhance the reconstruction of waveforms from mel-spectrograms. Specifically, it adjusts the Short-Time Fourier Transform (STFT) parameters and the dimensions of the spectrograms. It also utilizes the phase-gradient heap integration method to reconstruct the phase of the signal.

TiFGAN has achieved superior results, not only in comparison to SpecGAN but also in comparison to GANSynth. This is remarkable, especially considering its simpler and more lightweight architecture.

2.5 MelGAN

MelGAN[22] is a model utilized for high-quality spectrogram inversion.
Specifically, it can replace the Griffin-Lim algorithm, which, while efficient at converting spectrograms back to the time domain, introduces noticeable robotic artifacts.

MelGAN is primarily used for speech synthesis but also finds applications in music domain translation and unconditional generation.

Regarding its architecture, which is reported in Figure 2.5, the generator is a convolutional feed-forward network that takes a mel-spectrogram as input and generates a raw waveform as output. So, unlike most other GAN models, this generation does not require a global noise vector as input.
Notably, the generator architecture features residual blocks with dilations after each upsampling layer, with the receptive field that increases exponentially with the number of layers. This approach enhances long-range correlations within the generated audio samples.
Additionally, weight normalization[43] was applied, leading to significant improvements in sample quality.

Additionally, a multi-scale architecture with 3 discriminators is employed. All discriminators share the same network structure but focus on different scales of the audio. The first discriminator operates on the raw audio, while the other two focus on raw audio downsampled by a factor of 2 and 4, respectively.
This approach is highly effective, as audio exhibits structure at various levels. Notably, the first discriminator learns the high-frequency range, while the other two exclusively discriminate features in the low-frequency components.

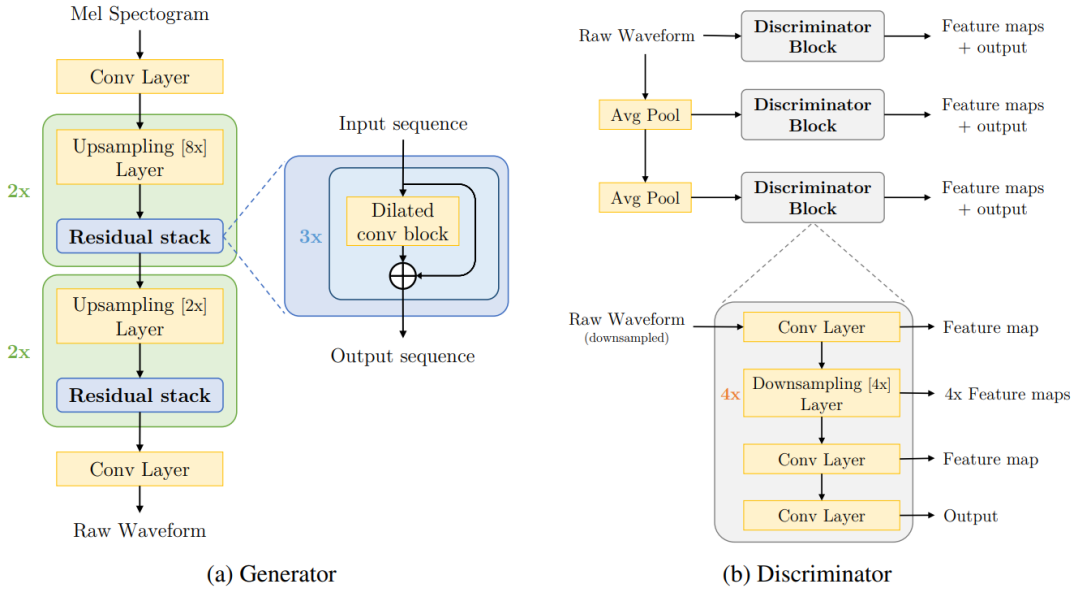

Figure 2.5: MelGAN Generator (a) and Discriminator (b) architectures. Image taken from [22].

During the training phase, the learning rate was set to 0.0001 and the batch size to 16. The hinge loss function of the GAN objective[24] was employed.

In the generation phase, MelGAN outperforms other alternatives for mel-spectrogram inversion in inference time due to its smaller number of parameters and its non-autoregressive architecture.

In speech synthesis, MelGAN achieved excellent outcomes and it's able to generalize to unseen speakers for mel-spectrogram inversion.

It's also comparable to some of the best performing models when employed as a vocoder component in a Text-To-Speech pipeline.

Additionally, it achieved decent results in music domain translation and unconditional generation, even though these were not the primary focus of the research

study.

2.6 HiFi-GAN

HiFi-GAN[21] is a model developed for synthesizing high-fidelity waveforms from mel-spectrograms, primarily in the context of speech synthesis.

It comprises the following components:

- A fully convolutional generator that takes mel-spectrograms as input and generates the corresponding waveforms.
- Two multi-scale and multi-periodic discriminators, each specializing in different periodic segments of the audio samples.

The generator and the two discriminators are trained adversarially and incorporate two additional loss functions, which serve to enhance training stability and improve the model's overall performance.

HiFi-GAN was trained on the LJ Speech Dataset, which consists of 13000 audio clips from a single speaker.

This model has achieved remarkable results, surpassing all other models in its domain.

2.7 Fre-GAN and Fre-GAN 2

Fre-GAN[20] is a model inspired by StyleGAN2[19] which is used to synthesize frequency-consistent audio on par with ground-truth audio. The model takes mel-spectrograms as input and converts them into their corresponding waveforms.

Fre-GAN employs a resolution-connected generator and two resolution-wise discriminators to learn different levels of spectral distribution across multiple frequency bands.
The generator upsamples and combines multiple waveforms at different resolutions, and each waveform is then evaluated adversarially in the corresponding resolution layers of the discriminators.

The model is trained using the least-squares GAN objective[27] on the LJ Speech dataset, which includes 13,000 audio clips from a single speaker.

The results demonstrate that Fre-GAN outperforms other vocoders, including HiFi-GAN.

Fre-GAN was further improved in its next iteration, Fre-GAN 2[23]. Notably, this new version added the inverse discrete transform to the generator, reducing the number of parameters and accelerating the audio generation speed without compromising quality.

2.8 CARGAN

CARGAN[32] is a model designed for conditional waveform synthesis using autoregression. In fact, autoregression can help prevent pitch inaccuracies and artifacts.

CARGAN consists of three main components:

- An autoregressive conditioning stack that considers a fixed number of previous samples.
- A generator network that inverts the mel-spectrogram into the corresponding waveform.
- A series of discriminators for adversarial feedback.

CARGAN was able generate samples with improved pitch accuracy and subjective quality when compared to HiFi-GAN. At the same time, it significantly decreased the time needed for training and memory consumption.

2.9 VQCPC-GAN

VQCPC-GAN[34], which is based on DrumGAN, is the pioneering GAN designed for variable-length music generation.

The primary concept behind this model involves extracting Vector-Quantized Contrastive Predictive Coding (VQ-CPC), which serves as conditional input to the architecture, offering time-dependent features.
Simultaneously, the input noise remains constant over time, ensuring the temporal consistency of global features. Consequently, the input to the generator encompasses both dynamic information, related to local frame-level context, and static information, linked to the global context.

In contrast to DrumGAN, VQCPC-GAN employs two discriminators, rather than just one, to ensure proper consideration of the conditioning. As DrumGAN, it follows the Progressive Growing of GANs training methodology[17].

VQCPC-GAN can generate variable-length sounds with controllable pitch, while achieving results comparable to previous GAN models designed for fixed-length generation.

2.10 UNAGAN

UNAGAN (Unconditional Audio Generation GAN)[25] is a model that is based on BEGAN[2] and aims to unconditionally generate audio samples.

Its generator features a hierarchical structure in which acoustic elements are generated from coarse to more refined ones, as depicted in the Figure 2.6. This hierarchical structure governs the temporal coherence of the generated samples.

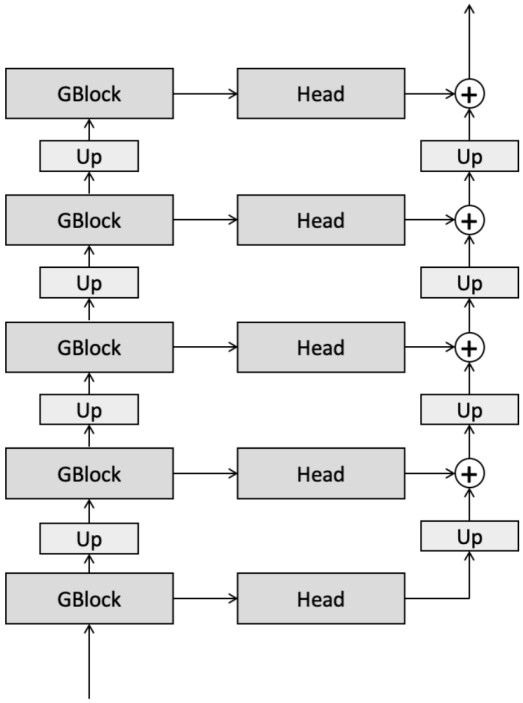

Figure 2.6: Hierarchical architecture of the generator. GBlock is a stack of convolutional layers and gate recurrent unit layers. Head is a convolutional layer and Up is an upsampling operation. Image taken from [25].

As input, the generator takes a variable-length sequence of noise vectors sampled from a Gaussian distribution with zero mean and unit variance, as opposed to just

one noise vector as the previous GAN models.
This sequence of input noise vectors is then converted into a mel-spectrogram of corresponding length, with each input vector corresponding to a certain length in the output. Consequently, this architecture can generate variable-length audio more efficiently.

Furthermore, a cycle regularization mechanism is employed to prevent mode collapse, which can lead to the generator producing insufficiently diverse sounds. Cycle regularization also enforces cycle consistency between each input noise vector and the corresponding segment in the output mel-spectrogram.

The mel-spectrograms are subsequently transformed into the corresponding audio waveforms using MelGAN.

UNAGAN was trained on the following datasets:

- Speech: LJ Speech dataset, which includes 13000 short audio clips from a single speaker.
- Singing: 17.4 hours of female voices singing Jazz.
- Piano: MAESTRO dataset[12], with 23 hours of piano performances.
- Violin: a collection of 16.7 hours of violin solo recordings.

UNAGAN has obtained very good results in speech and singing generation, and the results for piano and violin sounds are also promising, even though they were not the primary focus of the research study.

2.11 LoopTest

This research study[15] aimed to provide a performance benchmark for various GAN models when used to generate drum loops.

Three models were considered:

- StyleGAN[18].
- StyleGAN2[19].
- UNAGAN.

Both StyleGAN and StyleGAN2 generate fixed-length output, whereas UNAGAN is capable of generating variable-length output. However, for this study, all the

generated loops were standardized to have a duration of 2 seconds with a fixed tempo of 120 bpm.

To adapt the architecture of StyleGAN and StyleGAN2 for this specific task, their input tensors were modified to generate 80 x 320 mel-spectrograms through four upsampling blocks.

The mel-spectrograms generated by the three models were subsequently converted to waveforms using MelGAN.

The models were trained with the FreeSound Loop Dataset[41], which consists of 9455 drum loops.

The results indicated that UNAGAN and StyleGAN2 performed the best, yielding similar results, while StyleGAN yielded the poorest results.

2.12 PjLoop-GAN

PjLoop-GAN[53] is a model designed for the unconditional generation of one-bar drum loops and synth loops at a tempo of 120 bpm.

It employs the concept of Projective GAN, which involves the use of a pre-trained feature network to constrain the feature space of the discriminator. This constraint serves to stabilize and enhance GAN training. With this feature network, the model converges five times faster without any degradation in performance.

PjLoop-GAN is built upon the base architecture of StyleGAN2. However, specific modifications were made to the generator to optimize it for this particular task. Notably, the number of fully-connected layers was reduced to six, and a smaller noise vector was used. This adjustment addresses the issue of "intrinsic dimension"[39] within the data, where an excessively large latent space for the noise vector can introduce redundancy that obstructs the generator, leading to mode collapse. The generated mel-spectrogram are then inverted using MelGAN.

The following datasets were used:

- Youtube-100[13]: a private dataset from Google, containing 100 million YouTube videos.
- MagnaTagATune: a dataset that consists of 25,863 music clips, each lasting 29 seconds.
- Looperman dataset, consisting of 23,983 drum loops and 22,625 synth loops.

The first two datasets were utilized to train the feature network, while the last one was employed for training the loop generation model.

The results show that the performance of PjLoop-GAN is comparable to the state-of-the-art in this domain. However, the generated synth loops exhibit lower quality compared to the drum loops, suggesting that generating harmonic sound loops is more challenging than inharmonic ones.

2.13 Catch-A-Waveform

Generative models for visual data that can learn from a single image, such as SinGAN [45], have gained significant popularity. These models have been extended to various domains, including videos[11] and 3D graphics [14]. Notably, these models can be trained on a single image to generate modified versions of it, even at higher resolutions.

Catch-A-Waveform[9] (CAW) is inspired by these models and is applied in the audio domain. It can be trained using only a few seconds of audio from any domain, whether it's instrumental music or speech. A distinct benefit of this approach is that it removes the necessity for extensive datasets that include hours of audio samples.

This model can be used for multiple tasks:

- Unconditional generation.
- Inpainting.
- Denoising.
- Bandwidth extension.

CAW is a multi-scale GAN architecture capable of generating signals in their raw representation (time domain). This architecture is depicted in Figure 2.7 and consists of multiple blocks, including:

- The analysis pyramid, which uses multiple down-sampling factors to analyze the training signal at different sampling rates.
- The synthesis pyramid, which is composed of a pyramid of generators and discriminators. This pyramid sequentially generates a fake sample, with each level conditioned on the previous one. In particular, the sampling frequency is gradually increased at each level, from coarse to fine, by using different upsampling factors. While the sampling rate is increased at each scale, all generators and discriminators have the same receptive field. As a consequence, larger effective receptive fields correspond to lower sampling rates, shaping the

global structure of the signal. Gaussian noise serves as the starting point for the generation.

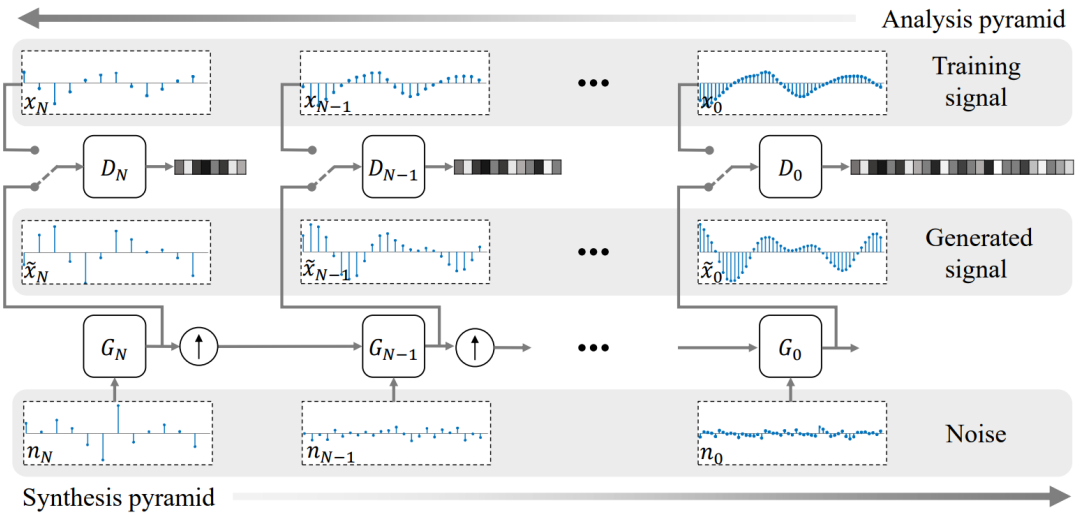

Figure 2.7: CAW multi-scale architecture. Image taken from [9].

The generators and discriminators at all scales have the same fully-convolutional architecture, with stacked blocks of 8 dilated convolutions followed by batch normalization and leaky ReLU.

The training process follows a coarse-to-fine approach as well. The loss function incorporates the Wasserstein GAN loss[10] along with a gradient penalty[1].
Additionally, a reconstruction loss is employed to ensure that the reconstructed signal at each scale closely matches the input signal at the corresponding scale.

For the different experiments, the training phase encompassed 3000 epochs, using samples at 16kHz. The learning rate was initially set to 0.0015 and reduced by a factor of 10 after 2000 epochs.

The results showed that just a few seconds of training signal is enough to achieve state-of-the-art results for all the different tasks. Furthermore, in the tasks of inpainting and bandwidth extension, CAW outperformed models trained on datasets with hours of audio samples.
The results differed significantly from a simple copy-paste approach, displaying substantial distinctions between the training signal and the generated ones.

In addition, CAW can generate samples of arbitrary duration that, thanks to its multi-scale architecture, maintain semantic similarity with the training signal, while at the same time introducing new compositions.

However, CAW has the following limitations:

- It can't learn languages or the rules of harmony since it's trained on a single sample. Consequently, generated speech samples may contain nonsensical words, and music samples may lack long-range harmonic structure.
- It is less effective at handling high-pitched signals.
- Training samples with reverb can generate high-pitched noise.

2.14 Other Audio-Related Tasks

Aside from unconditional generation and mel-spectrogram inversion, GAN models have been utilized for various tasks. These tasks include:

- Inpainting: given an audio file with gaps of silence, this task involves filling these gaps naturally in line with the rest of the audio file. Some examples of GAN models suitable for this purpose are GACELA[29] and CAW.
- Super-resolution: this task aims to increase the sampling rate of an audio file to achieve a higher quality. A GAN model that can be used for this task is Dual-CycleGAN[54].
- Video to audio generation: this task involves generating an audio file from a video source. An example of a GAN designed for this task is Dance2Music-GAN, which generates music based on dance video frames and human body motion as input[56].

Chapter 3

Datasets

3.1 One-Shot Percussive Sounds

This dataset comprises 9839 one-shot percussive sound samples, each recorded with a sampling rate of 16000 Hz and with a duration of approximately 1 second.
This dataset doesn't exhibit a wide range of sample diversity.

3.2 Good Sounds

The Good Sounds dataset offers approximately 28 hours of recordings of 12 different instruments, played by 15 musicians. These instruments include the flute, cello, clarinet, trumpet, violin, alto sax, tenor sax, baritone sax, soprano sax, oboe, piccolo, and bass. The notes are captured at a sampling rate of 48000 Hz with 32-bit precision and include a range of labels, like dynamics, location, player, and pitch.

However, when employed in a deep learning context, this dataset has the following constraints:

- The audio samples exhibit varying duration.
- The audio samples demonstrate disparities in intensity and dynamics.

3.3 NSynth

The Nsynth dataset presents 305979 samples which consists of single notes generated by 1006 instruments from commercial libraries.
These samples are recorded at a sampling rate of 16000 Hz and have a duration of

4 seconds. Notably, the dataset offers an average of 65.4 different pitches for each instrument. Additionally, each note is associated with a range of features, including information on the source, instrument, and pitch.

3.4 MAESTRO

The MAESTRO dataset[12] comprises audio samples extracted from 1,276 piano performances, accumulating to around 200 hours of data. Nevertheless, a primary limitation of this dataset is the inconsistency in the samples duration, as the sample lengths can vary.

3.5 First Custom Dataset - Inharmonic Samples

This dataset comprises samples extracted from 22 long audio files. Specifically, three different datasets were obtained based on the duration of the extracted samples:

- 11000 samples of 1 seconds.
- 11000 samples of 2 seconds.
- 10995 samples of 4 seconds.

All three datasets are derived from the same audio files, and each of them comprises 16-bit environmental and abstract inharmonic audio samples recorded at 16000 Hz. These samples exhibit a wide variety, making this dataset exceptionally valuable for assessing the performance of models trained on datasets with extensive variations and diverse audio features.

3.6 Second Custom Dataset - Harmonic Samples

This dataset comprises 6098 samples recorded at a sampling rate of 16000 Hz. Each sample has a duration of 1 second and features a single note played on an electric guitar. The samples were recorded using the following procedure:

- The guitar was directly connected to a computer through an audio interface to prevent the recording of any unwanted noise.
- A Python script was utilized to promptly record 1 second of audio as soon as a note was played.

This approach ensured the consistent capture of the note's attack while sacrificing only a portion of the release.

This dataset was created to address the limitations of available online datasets, including:

- Inconsistent balance of low and high notes, with a significant predominance of one over the other.
- Uncertainty regarding the range of notes included in the dataset.
- Variable duration of audio samples. As a consequence, trimming these audio files to obtain 1-second samples can lead to inconsistencies in the resulting samples.

In summary, this dataset offers a well-balanced representation of high and low notes spanning three octaves, with a fixed 1-second duration for each sample.

Chapter 4

Experiments

4.1 SpecGAN

As the first model to run the tests, SpecGAN was chosen. This is because SpecGAN is a light and fast model to train, which nevertheless performed very well, as reported in the corresponding paper[6].
As a result, it is an excellent model for testing different parameters and different approaches for the task of unconditional audio generation.

4.1.1 Experiment 1 - Percussive Sounds

In the initial experiment involving SpecGAN, the dataset "One-Shot Percussive Sounds" was utilized.
For the training process, the following parameters were configured:

- The learning rate of both the generator and the discriminator were set to 0.0002.
- The training process spanned 240 epochs.
- The batch size was set to 64.

During the training phase, the model checkpoints were saved every 10 epochs. This approach facilitated the comparison of audio samples generated from checkpoints at various training stages, aiding in the assessment of model convergence and providing insight into the approximate number of epochs needed for generating samples consistent with the training set and of high quality. Moreover, in this way it

was possible to estimate fairly accurately the number of epochs beyond which the model overfitted.

Once the training process was completed, a range of samples was generated using different model checkpoints to evaluate their quality at distinct training stages.
Notably, the results displayed significant differences between samples generated from different checkpoints, particularly within the initial 80 epochs. This suggests that the model had not yet converged before the 80-epoch mark.
For checkpoints between 80 and 180 epochs, on the other hand, the model tended to generate more similar sounds, indicating once again potential convergence around the 80-epoch point.

Another feature to highlight is that every checkpoint tended to generate sounds that were very similar to each other, with at most one or two more prominent variations, each of which had several less noticeable variations. This trend of reduced variety in samples generated from the same checkpoint became more pronounced as the corresponding epoch count increased.

These characteristics of the generated samples could be attributed to two primary factors:

1. Firstly, the dataset used for the training is composed of sounds that are very similar to each other, potentially contributing to the observed limited variability in the generated samples.

2. Secondly, the model might have converged around the 80-epoch mark, leading to increased similarity in generated sounds. This might be linked to the relatively limited size of the training set, potentially causing the model to display signs of overfitting early in the training process.

However, despite the limited diversity in the generated samples, their quality was already commendable after just 80 epochs. At this stage, in fact, the generated samples exhibited resemblance to real samples of the training set.

In conclusion, based solely on this experiment, it can be inferred that, for a dataset of moderate size and diversity, approximately 100 epochs are sufficient for the model to achieve good sample quality. In fact, this epoch count strikes a balance between achieving results with good quality and avoiding overfitting.
Therefore, in subsequent experiments, a key focus was examining the effects of training the model using a more diverse training set.

4.1.2 Experiment 2 - More Varied Dataset

For this second experiment with SpecGAN, the first custom dataset was used. In fact, the primary objective of this experiment was to assess how using a more diverse dataset impacts the quality of samples generated by the model.
The dataset used for this experiment was slightly larger than the one utilized in Experiment 1, containing approximately 2000 additional samples. As such, this dataset was utilized to explore the potential of using SpecGAN to generate samples with more variety.

Regarding the training process, the same parameters as those in Experiment 1 were maintained. This approach enabled a direct comparison of the dataset's influence on the model's performance without introducing other variables.
Consequently, the learning rate of both the generator and the discriminator were set to 0.0002 and the batch size was set to 64. This time, the model was trained for 220 epochs. As for Experiment 1, the checkpoints were saved every 10 epochs to evaluate the quality of the generated samples at different stages of the training process.

The generated samples started to resemble the ones of the training set already at 80 epochs, with more noticeable improvements observed around 150 epochs.
However, an important observation from the results is that the generated samples exhibited slightly more noise compared to the training set. This could be due to various factors. Specifically, the training set comprised 22 distinct classes of samples with substantial dissimilarities. This diversity might have caused the model to blend features from different sounds, resulting in noise within the generated samples. In fact, by analyzing the training set, it can be noticed that some classes of samples exhibit noisier, intense spikes, or quieter characteristics, while others are cleaner. This could be the reason why the generated samples resemble some of the cleaner samples in the training set, while preserving the noisy characteristics of other training samples.
Another major factor contributing to the noisy nature of the generated samples could reside in the characteristics of the latent space of the model from which the samples are generated. In fact, SpecGAN generates the samples starting from a 100-dimensional random noise vector. Consequently, the noisy nature of the generated samples might be due to the random characteristics of the latent space from which they are generated.

In any case, having a more varied dataset had a major impact on the variety of the generated samples. In fact, the generated samples tended to have multiple major variations as well as many less noticeable ones, while retaining some similarities between different checkpoints. This trend was particularly pronounced in checkpoints trained for less than 150 epochs.

It is important to note that, even with an extended training duration, the generated samples continued to exhibit diverse variations. In fact, unlike Experiment 1, where models after 100 epochs tended to display only one or two variations, in this case, even after 220 epochs, the generated samples maintained increased diversity. This suggests that this model required a greater number of epochs to converge, possibly due to the larger and more varied dataset used in this experiment.

Moreover, the generated samples, although partly influenced by noise, closely resemble those in the dataset. In fact, when comparing the spectrograms of one generated sample with a similar one in the dataset, it becomes apparent that they share the same features, as depicted in Figure 4.1.
These observations suggest that this model could be a great tool for generating inharmonic samples.

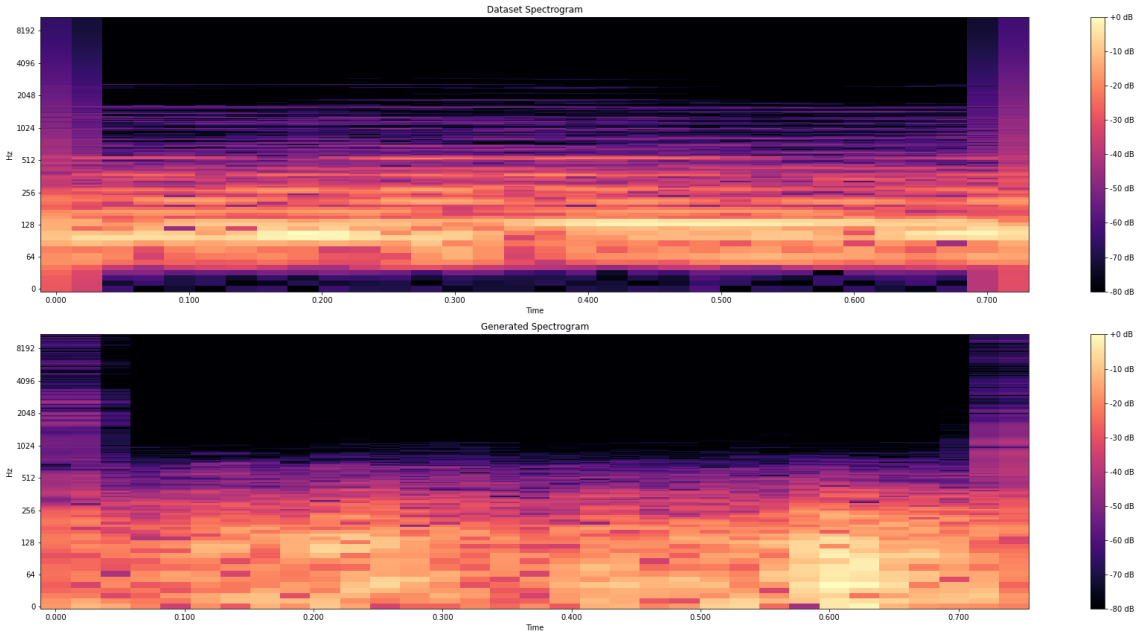

Figure 4.1: Experiment 2 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

In summary, while this experiment highlighted the influence of a varied dataset on the generated samples, it has also opened up new questions and problems that subsequent experiments aim to address. These include:

- Is it possible to generate a noise-free sample without recurring to post-processing?

- How does the learning rate of the generator affect the generated samples, considering that in this experiment the model converged fairly quickly?
- How does the model perform when training it with harmonic sounds?

4.1.3 Experiment 3 - Repeated And Longer Training

This experiment had multiple objectives:

- To determine whether conducting multiple runs of the training phase, while keeping the same parameters and the same dataset, results in the generation of similar samples.
- To assess whether training the network with a significantly larger number of epochs leads to overfitting or improvement.

To achieve the first objective, this experiment utilized the same parameters and dataset as Experiment 2.
Consequently, the learning rate of the generator and of the discriminator were both set to 0.0002 and the batch size was set to 64. In addition, the first custom dataset was utilized.
To achieve the second goal, however, the number of epochs for this experiment was set to 400, nearly double that of Experiment 2. Checkpoints were saved every 40 epochs to evaluate the quality of the generated samples at different stages of training.

The generated samples exhibited more diversity for checkpoints under 150 epochs, while samples beyond 200 epochs tended to be more similar. This observation suggests a convergence of the model around 150 epochs.
Also, as in Experiment 2, the generated samples tended to be more varied than those in Experiment 1, affirming that the diversity of training samples also translated into the generated ones.

Despite sharing the same training parameters, the samples produced in this experiment differed significantly from those in Experiment 2. In particular, among the classes of samples present in the dataset, both experiments generated samples that merge 4 or 5 classes. However, these classes differed between the two experiments.
This discrepancy probably derived from using a random seed during training from which generate samples. In fact, a variation in the seed for the first epoch could lead to considerably different training trends. Nevertheless, in both experiments, the generated samples effectively captured features from the training set. In fact, in this experiment as well, the spectrograms of a generated sample and those of a similar sample from the dataset exhibit the same characteristics, as shown in Figure 4.2.

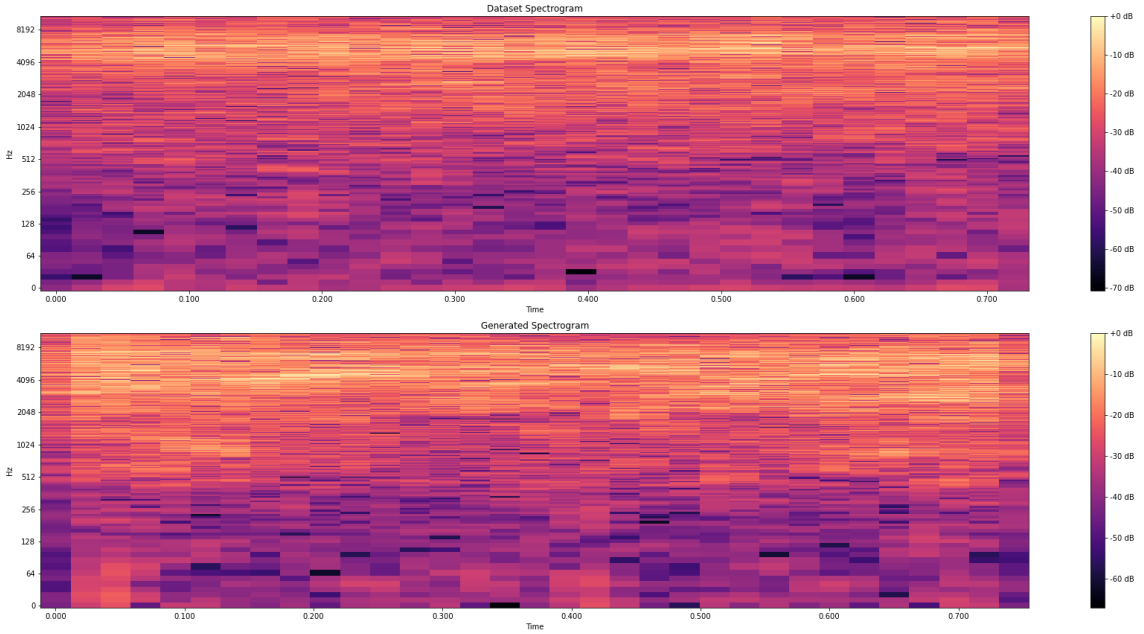

Figure 4.2: Experiment 3 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

As for sample quality, once again, they present some noise, attributable to the same factors described in Experiment 2.

To summarize:

- Repeating the training process can lead to very different generated samples, even when using the same exact parameters.
- For a dataset of this size and diversity, around 150 epochs are sufficient to achieve model convergence.

4.1.4 Experiment 4 - Lower Learning Rate of the Generator

The objective of this experiment was to determine whether reducing the learning rate of the generator would lead to different outcomes.
In fact, the trends observed in the samples generated in Experiments 2 and 3 indicated a rapid convergence of the generator. In particular, the generated samples exhibited decreased diversity after only 150 epochs.

Consequently, in this experiment, the learning rate of the generator was decreased in order to grant the model more time for capturing the features of the training data.

To exclusively assess the impact of this alteration, the same parameters as those in Experiments 2 and 3 were retained.
Therefore, the batch size was set to 64 and the learning rate of the discriminator was set to 0.0002. However, the learning rate of the generator was reduced by half to 0.0001, and the training process was conducted over 240 epochs.

As expected, the loss of the generator in this experiment was notably higher compared to the previous experiments. This clearly indicates that the alteration in the learning rate had an impact on the training process.

In this experiment as well, the generated samples closely resembles the ones in the dataset, as one can notice from Figure 4.3 and Figure 4.4. However, the generated

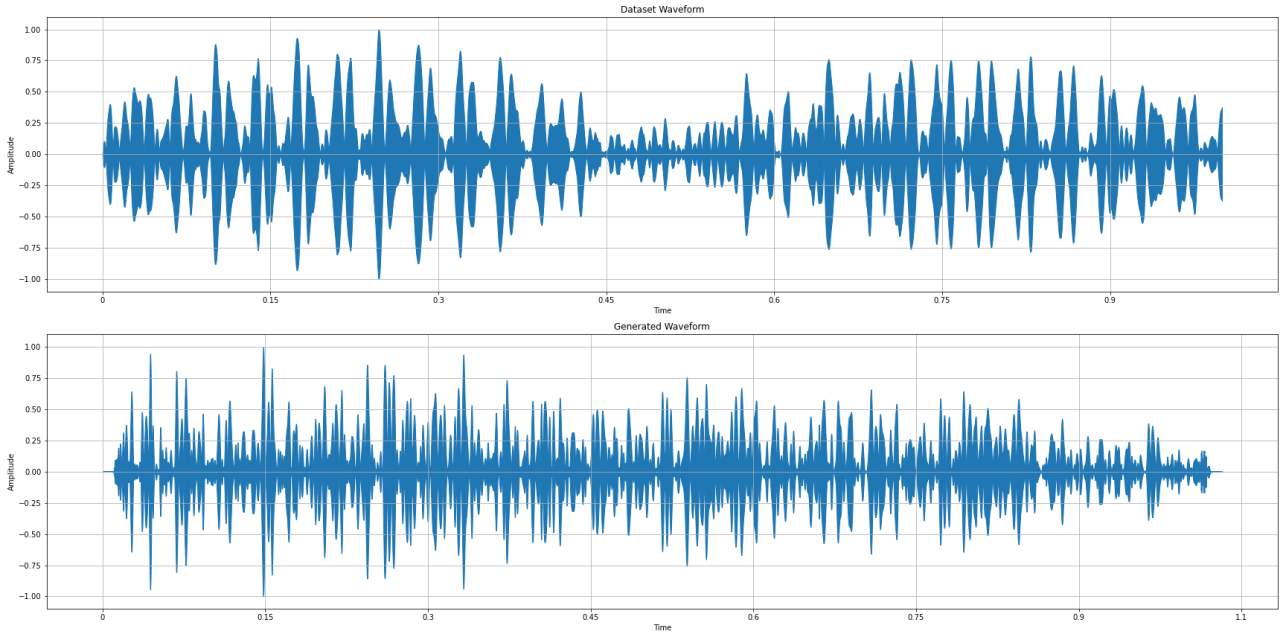

Figure 4.3: Experiment 4 - Comparison between the waveform of a sample from the dataset (top) and the one of a generated sample (bottom).

samples exhibited a similar variety and quality when compared to those from the preceding experiments, with a reduction in variation after 200 epochs.

Hence, based on the findings of this experiment, it can be inferred that the diversity in the generated samples primarily derives from the variety within the training dataset rather than being solely influenced by the learning rate.

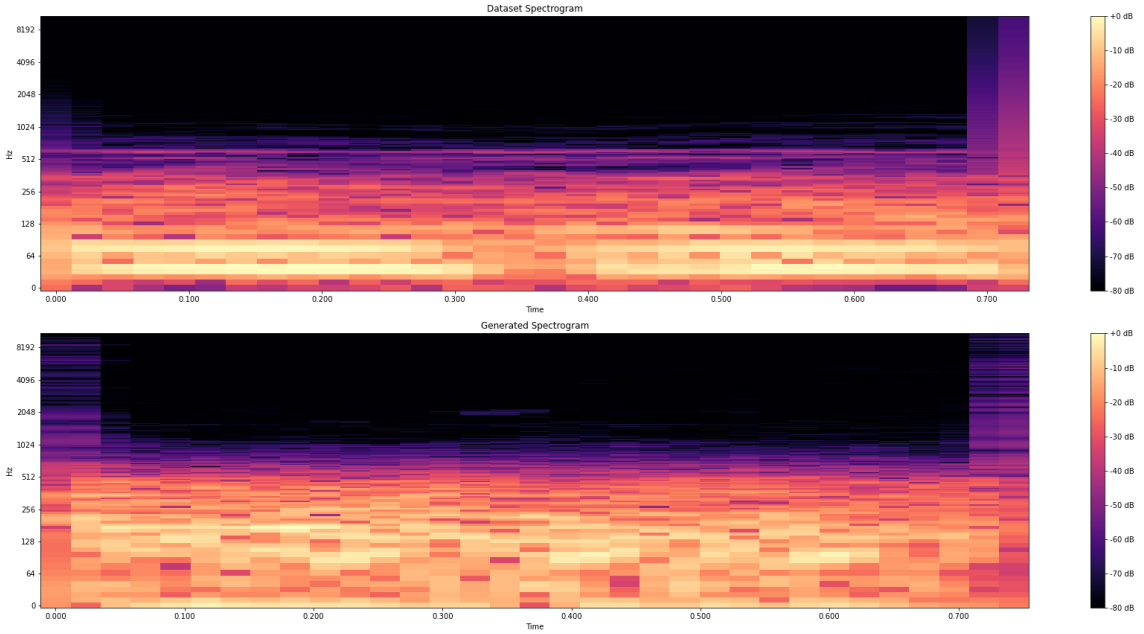

Figure 4.4: Experiment 4 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

Furthermore, the convergence appears to occur around 150 to 200 epochs for all these experiments, confirming the adequacy of this epoch range for a dataset of this size and diversity.

4.1.5 The Comb Filter

The comb filter is a filter that, given an input signal, introduces a delayed copy of that signal, resulting in interference.

Its response frequency exhibits multiple notches and peaks. The comb filter can be applied to various tasks, with its primary applications in audio signal processing being as follows:

- Noise removal.
- Eliminating unwanted recurring interference.

There are two types of comb filters, peak and notch. These types are depicted in Figure 4.5 and Figure 4.6, respectively.

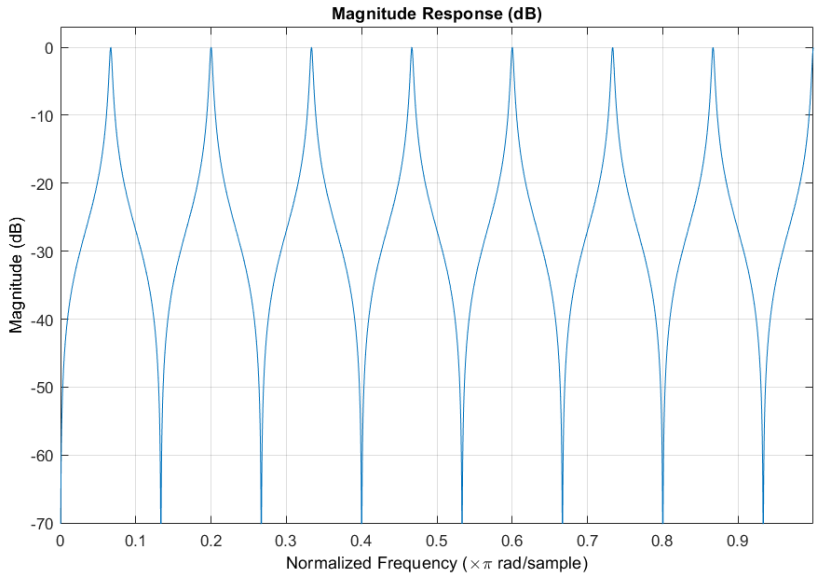

Figure 4.5: Example of peak comb filter.

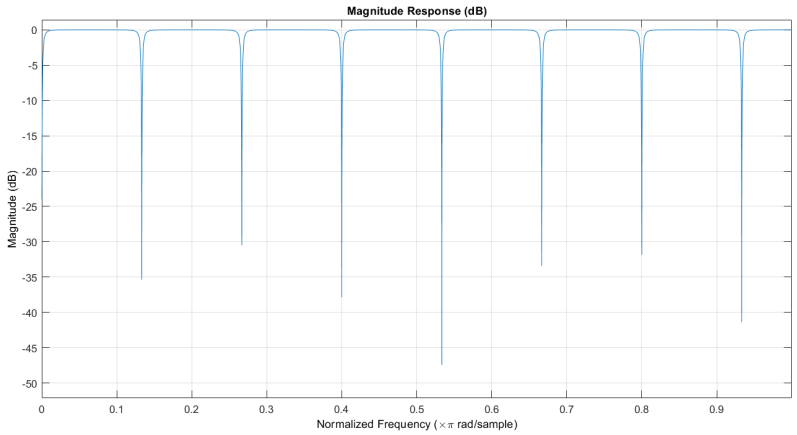

Figure 4.6: Example of notch comb filter.

Specifically, peak comb filters are employed to enhance the quality of periodic signals by eliminating noise, whereas notch comb filters are used to eliminate periodic interference. Consequently, for the upcoming experiments, the peak comb filter has been employed since the objective was to remove noise.

The primary parameters of the peak comb filter include:

- Order, which determines the number of peaks between 0 and 2Π.
- Bandwidth, which determines the steepness of the peaks and takes values between 0 and 1.

4.1.6 Experiments 5 and 6 - Comb Filter

The aim of these experiments was to address the noise present in SpecGAN-generated samples when the dataset exhibited significant sample variety.
To achieve this, a peak comb filter was used. Specifically, this filter was applied to the 100-dimensional vector from which the audio samples were generated. In this way, the random characteristic of the 100-dimensional vector were reduced during both training and generation.
Subsequently, the filtered 100-dimensional vector was normalized to maintain values within the range of 0 to 1.

To isolate the impact of the filter, the same parameters as those in Experiments 2 and 3 were employed for training. Consequently, the learning rates of both the generator and discriminator were set at 0.0002, and the batch size was set to 64. The training process spanned 160 epochs.

As for the filter, standard reference values were utilized to define its key attributes, specifically:

- The bandwidth was fixed at 0.01.
- The order was set to 5.

The visual representation of the filter can be observed in Figure 4.7.

The resulting generated samples tended to exhibit the same characteristics highlighted for Experiments 2 and 3, displaying diverse variations up to 150 epochs. However, these samples were still affected by noise.

Before progressing to subsequent experiments involving adjustments of the filter parameters, an additional experiment almost identical to the one just described was conducted. In this variation, the filtered 100-dimensional vector was not normalized. However, the generated samples showed no noticeable difference with respect to the ones generated in the previous experiment.

In conclusion, this experiment demonstrated that using these specific parameters for the filter does not substantially eliminate noise. As a result, for the upcoming experiments, adjustments were made to the filter parameters to enhance its effect.

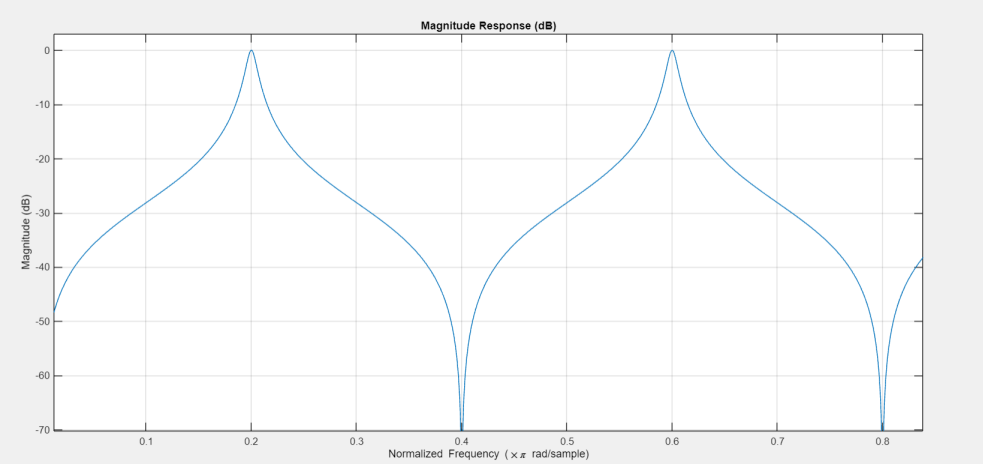

Figure 4.7: Comb filter applied in Experiments 5 and 6.

4.1.7 Experiments 7 and 8 - More Effective Comb Filter

These two experiments closely resembles Experiments 6 and 7, differing only in the adjustments made to the comb filter parameters, as represented in Figure 4.8. Specifically:

- The order was set to 40.
- The bandwidth was set to 0.01.

The reason behind these parameter changes lies in the expectation that increasing the order of the filter will reduce the noise in the generated samples.

The same dataset and parameters employed in the previous experiments were utilized.
Consequently, the learning rate of the generator and discriminator were set to 0.0002 and the batch size was maintained 64. The model in Experiment 7 underwent training for 120 epochs, while in Experiment 8 it was trained for 200 epochs.

The first of these two experiments mirrored Experiment 6, with the 100-dimensional vector undergoing normalization after the application of the comb filter. Instead, the second experiment aligned with Experiment 7, without the normalization of the vector.

The generated samples in these two Experiments exhibited greater diversity compared to earlier experiments. This is particularly apparent for Experiment 8, where, even after 200 epochs, the generated samples remain noticeably varied.
This is more significant when considering that, in the previous experiments, after

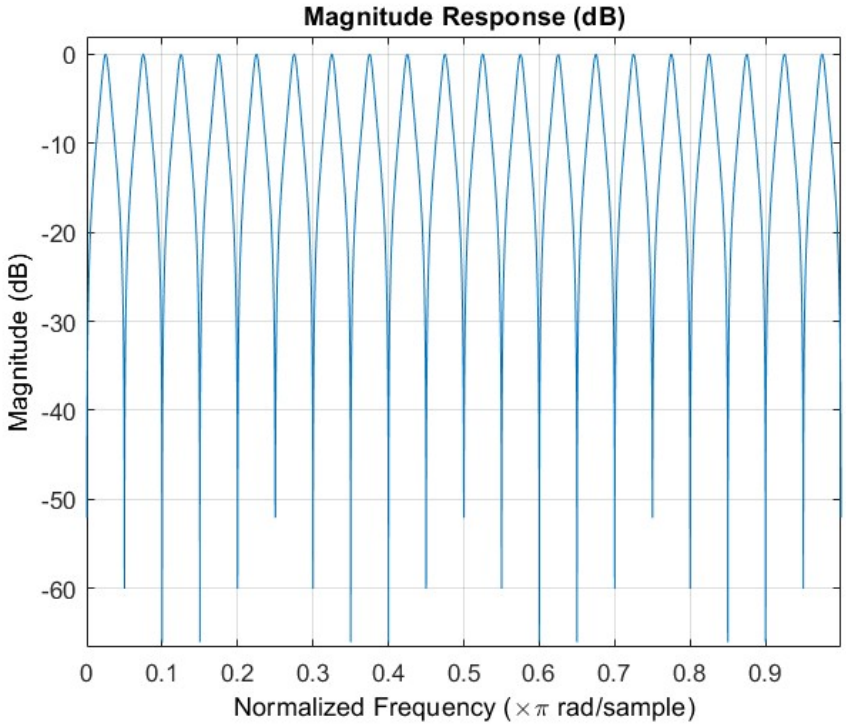

Figure 4.8: Comb filter applied in Experiments 7 and 8

150 epochs the generated samples tended to be very similar. These findings suggest that applying this specific type of comb filter to the latent space vector can enhance the variety in the generated samples, preventing early overfitting of the model.

In these two experiments, the generated samples displayed the same features as those in the dataset, as one can see in Figure 4.9.

Nonetheless, even in these two Experiments, the generated samples still present some noise.
Therefore, from the last 4 experiments, it can be inferred that applying a comb filter to the latent space vector does not lead to markedly less noisy samples.
Subsequently, the next experiments will target altering the stage at which the comb filter is applied during the training phase, aiming to assess its potential to reduce noise without having to resort to post-processing.

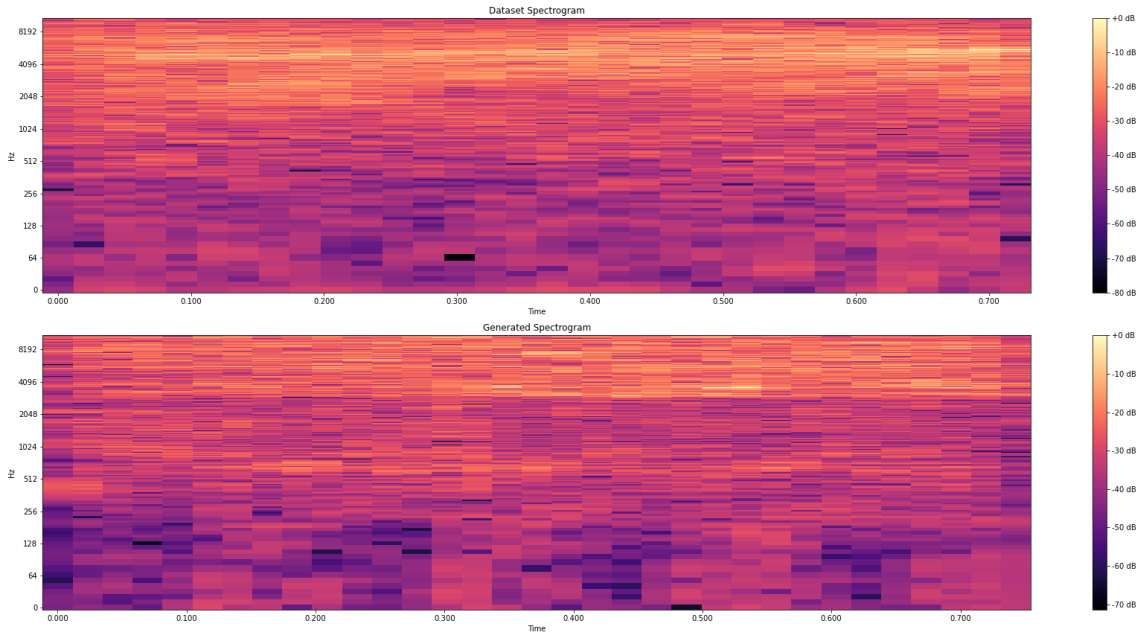

Figure 4.9: Experiment 7 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

4.1.8 Experiment 9 - Comb Filter Applied to the Waveform

The idea behind this experiment was to apply the same comb filter as Experiments 7 and 8, but at a different stage.
In fact, in previous instances, the filter was applied to the 100-dimensional random vector within the latent space. However, for this experiment, the filter was applied further along the training process.

Specifically, subsequent to generating the spectrogram and prior to computing the loss, the following steps were executed:

1. The spectrograms of the current batch underwent denormalization.

2. The denormalized spectrograms were converted into waveforms using the Griffin-Lim algorithm, iterated 20 times.

3. The comb filter was applied to the waveforms thus obtained.

4. The filtered waveforms were normalized.

5. The normalized waveforms were converted back into spectrograms so that the loss could then be computed. This approach would enable the network to learn the necessary weights for generating a spectrogram that, once inverted, filtered, and reconverted, closely resembled a real spectrogram of the training dataset.

To isolate the impact of this alteration, the training phase employed identical parameters and filters as seen in Experiments 7 and 8. Consequently, the generator and discriminator learning rates were both set at 0.0002, while the batch size remained at 64. The filter order was set at 40, with a bandwidth of 0.01.

However, a notable issue with this procedure emerged, which was its substantial computational demand. In fact, every training iteration necessitated the execution of all 5 steps explained above.

As a consequence, these operations became the new bottleneck of the training process, significantly elongating the required time. Therefore, achieving an adequate number of epochs proved unfeasible.

Hence, this experiment highlighted that, while applying the filter at this stage of training might theoretically offer greater efficiency compared to its application to the latent vector, it is ultimately overly expensive in terms of computational time.

4.1.9 Experiments 10 and 11 - Harmonic Sounds Dataset

Up to this point, all experiments conducted with SpecGAN utilized datasets with inharmonic sounds. The purpose of these two experiments, therefore, was to assess the quality of the generated samples when using a dataset containing only harmonic sounds.
For this reason, the second custom dataset containing one-second guitar notes was used. It is noteworthy that these two experiments varied in terms of the dataset size:

- Experiment 10 employed only 3068 files from the entire dataset.
- Experiment 11 utilized the complete dataset containing 6098 files.

In this way, the effect of the number of samples in the dataset was also tested.

To evaluate solely the difference in performance resulting from using harmonic sounds instead of inharmonic sounds, the same parameters as in Experiments 2 and 3 were used for training.
Consequently, the learning rates of the generator and discriminator were set to 0.0002 and the batch size to 64.
For Experiment 10, the training process extended to 520 epochs, while for Experiment 11, it was carried out for 200 epochs.

The quality of the samples obtained in both experiments was unsatisfactory. In particular, the generated samples did not resemble those from the training set for any epoch. In fact, even when comparing the waveforms and spectrograms of the generated samples with those from the real dataset, as shown in Figure 4.11 and Figure 4.10, they appear to be completely different. Specifically, it can be observed that, in the real sample, which is harmonic, the spectrogram displays the fundamental frequency and all its harmonics. However, this characteristic is not present in the spectrogram of the generated sample. Additionally, the waveforms of the two audio files are completely different as well.

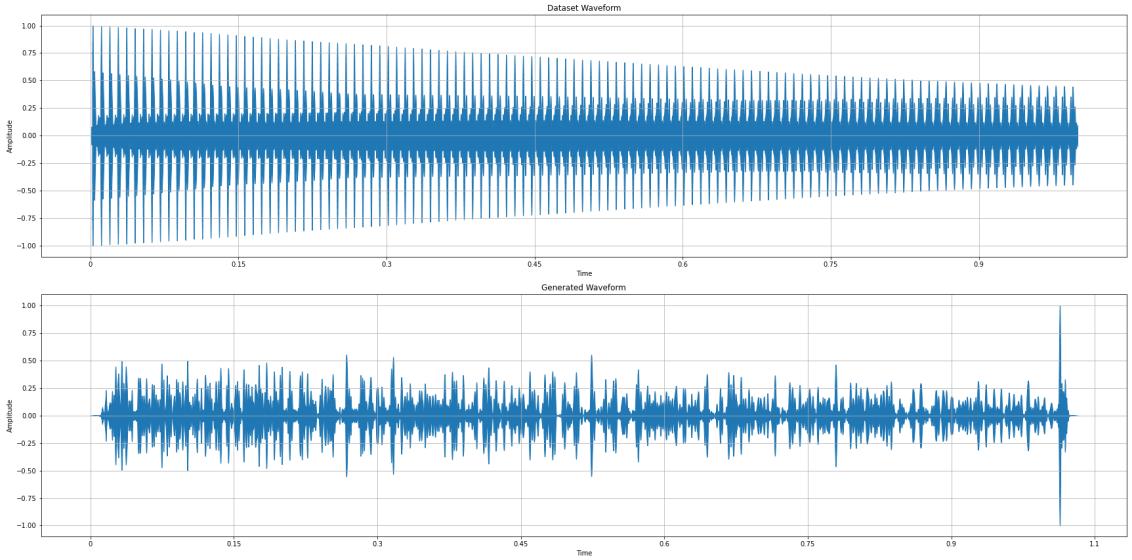

Figure 4.10: Experiment 11 - Comparison between the waveforms of a sample from the dataset (top) and the one of a generated sample (bottom).

This could be due to the following factors:

- The dataset lacked a sufficient number of samples.
- The batch size was too large for the limited size of this dataset.
- SpecGAN might not be suitable for generating harmonic sound samples.

The first two factor seemed the most probable, since this dataset contained about half the samples compared to the dataset used for Experiments 2 and 3.

Consequently, in the subsequent experiment, a smaller batch size was chosen to observe its effects on the generated samples.

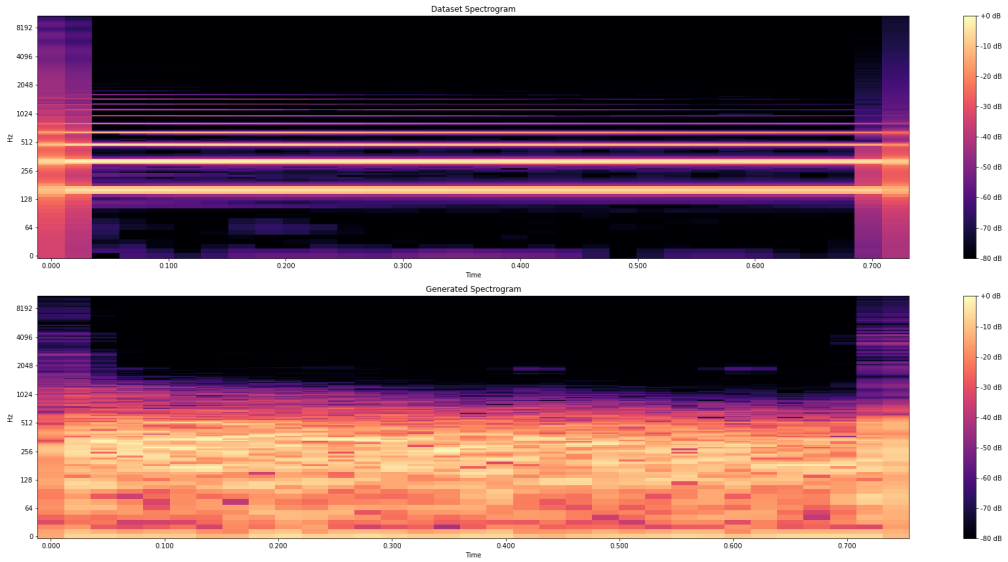

Figure 4.11: Experiment 11 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

4.1.10 Experiment 12 - Effect of Batch Size

The aim of this experiment was to assess how the batch size affects the quality of generated samples, with the objective of achieving better samples than those obtained in the previous two experiments.

To achieve this goal, the batch size for this experiment was reduced to 32, compensating for the reduced number of samples in the training set.
Additionally, compared with Experiments 10 and 11, the sounds in the dataset were normalized to ensure uniform intensity across all samples. This normalization eliminated intensity discrepancies as a factor affecting the results.
The remaining training parameters were kept consistent with those used in Experiments 10 and 11.

Regarding the generated samples, their quality improved notably, particularly around the 160th epoch. However, the results still didn't meet the desired standards. In fact, in this experiment, when comparing the waveforms and spectrograms of the samples in the dataset with those of the generated samples, they appear to be very different, as showed in Figure 4.12 and Figure 4.13.

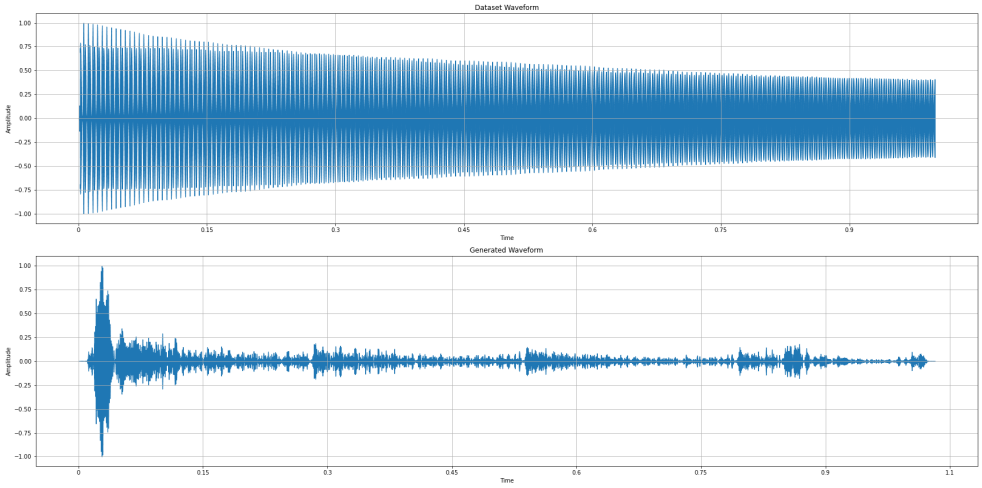

Figure 4.12: Experiment 12 - Comparison between the waveform of a sample from the dataset (top) and the one of a generated sample (bottom).

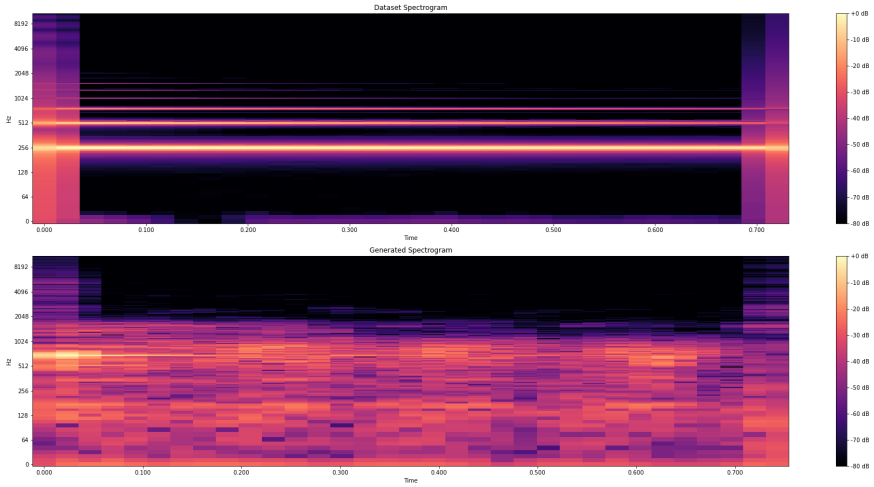

Figure 4.13: Experiment 12 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

Nevertheless, the improved sample quality compared to the previous two experiments suggested that the dataset could have an insufficient number of samples to produce high-quality results.

Consequently, in the upcoming experiment, the quality of the samples was tested using a dataset containing a greater number of harmonic sounds.

4.1.11 Experiment 13, 14 and 15 - Larger Harmonic Datasets

The aim of these three experiments was to test whether using a larger harmonic dataset would improve the performance of SpecGAN.

To achieve this goal, the following two datasets were selected:

- 14771 audio files extracted from "NSynth", which consisted of acoustic bass single notes.
- 11807 audio files extracted from "NSynth", which consisted of single notes of flute.

Since samples from the "NSynth" dataset have a duration of 4 seconds and SpecGAN uses one-second samples as input, for each sample of "NSynth" the last two seconds were discarded and the first two seconds were used as two separate samples.

Experiment 13 utilized the first dataset, while Experiments 14 and 15 both employed the second dataset.

For Experiment 13, the same training parameters were used as Experiments 2 and 3:

- The learning rates of the generator and discriminator were set to 0.0002.
- The batch size was set to 64.

The training process was carried out for 240 epochs.
Despite the increased dataset size, however, the results obtained with harmonic samples still exhibited insufficient quality.

Therefore, Experiments 14 and 15 were also conducted. In Experiment 14, the same parameters as in the previous experiment were retained, with the only alteration being the dataset.

Nonetheless, the results exhibited similar quality to those of the previous experiment. This might be due to the insufficient number of samples or the possibility that SpecGAN is not well-suited for generating harmonic samples.

Experiment 15 was carried out to evaluate whether reducing the batch size would enhance the sample quality. For this purpose, the same dataset and parameters from Experiment 14 were utilized, with the sole change being the batch size set to 32.

The training process spanned 120 epochs.
The results of this experiment were slightly better than those of the previous two experiments, but the quality was still not good enough. In fact, even in this case, both the spectrograms and the waveforms of the generated samples were very different compared to the ones of the real samples, as depicted in Figure 4.14 and Figure 4.15.

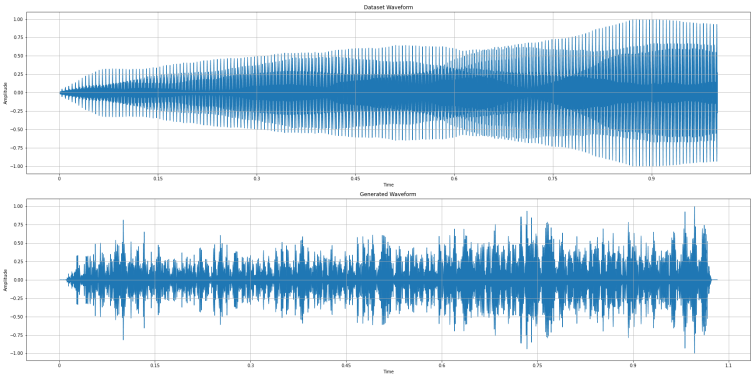

Figure 4.14: Experiment 15 - Comparison between the waveform of a sample from the dataset (top) and the one of a generated sample (bottom).

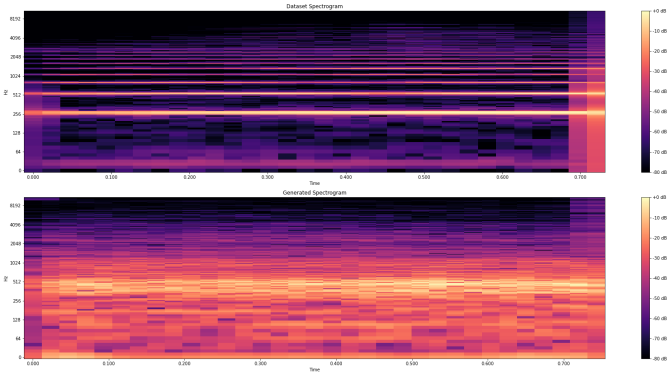

Figure 4.15: Experiment 15 - Comparison between the spectrogram of a sample from the dataset (top) and the one of a generated sample (bottom).

In summary, these last experiments led to the following potential conclusions:

- A significantly larger number of samples in the dataset is needed to generate high-quality harmonic samples compared to inharmonic samples.
- SpecGAN may not be suitable for generating harmonic samples, necessitating the use of a more complex model.

4.1.12 SpecGAN - Final Conclusions

The experiments involving SpecGAN have been summarized in Table 4.1.

EXPERIMENT	DATASET	Lr D	Lr G	EPOCHS	BATCH S.	COMB FILTER
1	One-Shot Perc. Sounds	0.0002	0.0002	240	64	No
2	First Custom	0.0002	0.0002	220	64	No
3	First Custom	0.0002	0.0002	400	64	No
4	First Custom	0.0002	0.0001	240	64	No
5	First Custom	0.0002	0.0002	160	64	Order=5 Applied to random vector Normalized
6	First Custom	0.0002	0.0002	160	64	Order=5 Applied to random vector Not normalized
7	First Custom	0.0002	0.0002	120	64	Order=40 Applied to random vector Normalized
8	First Custom	0.0002	0.0002	200	64	Order=40 Applied to random vector Not normalized
9	First Custom	0.0002	0.0002	/	64	Order=40 Applied to waveform Normalized
10	Second Custom Subset	0.0002	0.0002	520	64	No
11	Second Custom	0.0002	0.0002	200	64	No
12	Second Custom	0.0002	0.0002	200	32	No
13	NSynth bass	0.0002	0.0002	240	64	No
14	NSynth flute	0.0002	0.0002	240	64	No
15	NSynth flute	0.0002	0.0002	120	32	No

Table 4.1: Table with a summary of all the parameters and datasets used for each experiment with SpecGAN.

In conclusion, SpecGAN successfully generated inharmonic audio samples with good variety and quality, even when working with a relatively small dataset.
Moreover, its lightweight architecture and the short training time required to produce high-quality samples make SpecGAN an easily adaptable model for testing various training parameters.

However, there are instances where the generated inharmonic samples are affected by noise. Additionally, the samples are limited to a duration of only 1 second, which may restrict their practical applications.

Finally, when experimenting with harmonic sounds, SpecGAN proved to be unsuitable for generating such samples. This might be attributed to the following factors:

- The artifacts caused by the Griffin-Lim algorithm during the mel-spectrograms inversion, which become more noticeable when generating harmonic samples.
- The lightweight architecture's inability to handle harmonic data effectively.
- The limited amount of data available in the training set.

4.2 Catch-A-Waveform

The second model chosen for conducting the experiments is Catch-A-Waveform (CAW).

This choice was made because CAW differs significantly from SpecGAN in terms of both the dataset and the output it generates. Specifically, CAW requires only one file as input for the dataset, and it can generate samples of arbitrary duration as its output.

As a result, the experiments were conducted using CAW with the intention of using it independently and potentially also in combination with SpecGAN.

4.2.1 Experiment 1 - Inharmonic Sounds

The objective of this experiment was to assess the quality of the generated samples when the dataset consisted of a single inharmonic audio of short duration.
To achieve this, a sample of four seconds was chosen from the first custom dataset with a sampling rate of 16000 Hz.

For the training phase, the following parameters were set:

- The learning rate for both the generator and discriminator was set to 0.0015.

- A learning rate decay of 0.1 was applied.
- The number of epochs for each scale was set to 2000.

Due to the short duration of the input file, the training process was performed for only five scales, corresponding to the following sampling rates:

- Scale 4: 8000 Hz
- Scale 3: 10000Hz
- Scale 2: 12000 Hz
- Scale 1: 14400 Hz
- Scale 0: 16000 Hz

Each scale has a different receptive field, with larger fields associated with lower sampling frequencies. This approach allows to capture both global features that provide audio consistency and local features that are essential for generating intricate details.

The reconstruction loss values obtained for each scale were very low, suggesting that convergence has been achieved. Specifically, for each scale the values obtained were as follows:

- Scale 4: 0.0137
- Scale 3: 0.0118
- Scale 2: 0.0122
- Scale 1: 0.0139
- Scale 0: 0.0150

As for the generated samples, they exhibited a very high level of quality, comparable to the input file, although they were slightly affected by noise.
This noise was likely a result of the high frequencies present in the input file which, as described in [9], can introduce artifacts in the output file.

Samples ranging from 6 to over 30 seconds in length were generated. Increasing the duration of the files didn't result in unnatural interruptions or discontinuities. Additionally, the generated samples differed from a simple copy of the input file.
In fact, while comparing the spectrograms and waveforms of the original sample

and of a generated samples, the similarities are apparent, while still showing some differences, as depicted in Figure 4.16 and Figure 4.17.

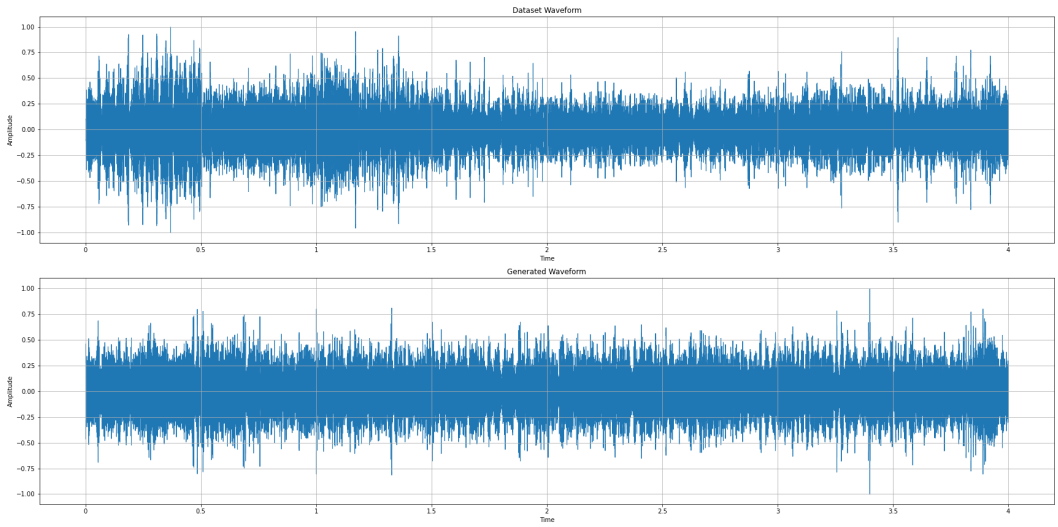

Figure 4.16: Experiment 1 - Comparison between the waveform of the sample from the dataset (top) and the one of a generated sample (bottom).

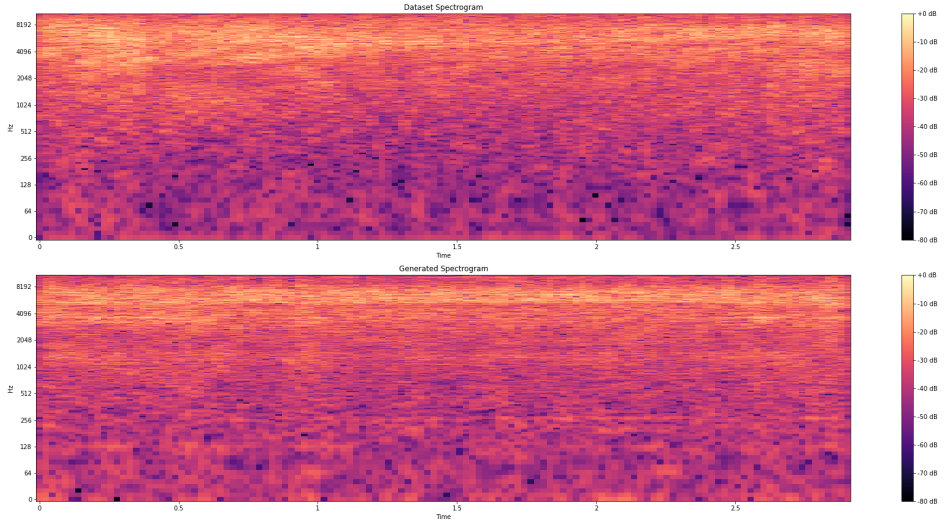

Figure 4.17: Experiment 1 - Comparison between the spectrogram of the sample from the dataset (top) and the one of a generated sample (bottom).

Moreover, the generated files displayed great diversity among themselves while still maintaining the essential characteristics of the input file.

In conclusion, this experiment demonstrates the capability of CAW to generate high-quality audio files from as little as 4 seconds of inharmonic audio.
Consequently, this network proved to be an excellent tool also for naturally extending audio files, resulting in outputs that, while sharing the same core features as the input, stand as distinct creations rather than mere duplicates.

4.2.2 Experiment 2 - Harmonic Sounds

The objective of this experiment was to evaluate the quality of the generated samples when using a single short harmonic audio file as the training data.
Consequently, this experiment was similar to the previous one but used an harmonic audio file as the dataset. To accomplish this, a sample from the "Good Sounds" dataset was selected as the input. Specifically, this sample comprises a single note produced by a cello and has a duration of approximately 6 seconds.

To ensure a fair comparison between the quality of harmonic and inharmonic sounds, the same parameters employed in the previous experiment were retained for the training process. Accordingly:

- The learning rate for both the generator and discriminator was set to 0.0015.
- A learning rate decay of 0.1 was applied.
- The number of epochs for each scale was set to 2000.

Since the training file for this experiment was longer than that of the previous one, training was conducted for 15 scales instead of 5. Consequently, the sampling rates for each scale ranged from 400 Hz for scale 14 to 16000 Hz for scale 0.

The reconstruction loss values remained consistently low across all scales, ranging from 0.0015 to 0.0071, indicating that convergence was achieved for each individual scale during the training process.

Once again, the generated samples demonstrated excellent quality, comparable to that of the input file. However, unlike the previous experiment, noise was not present, confirming that the noise observed in Experiment 1 was due to the presence of high frequencies in the training file. The waveforms and the spectrograms of the original audio file and of a generated sample are depicted in Figure 4.18 and Figure 4.19. Unlike the spectrograms of harmonic samples generated by SpecGAN, in this case, both the fundamental frequency and its harmonics are present in the generated

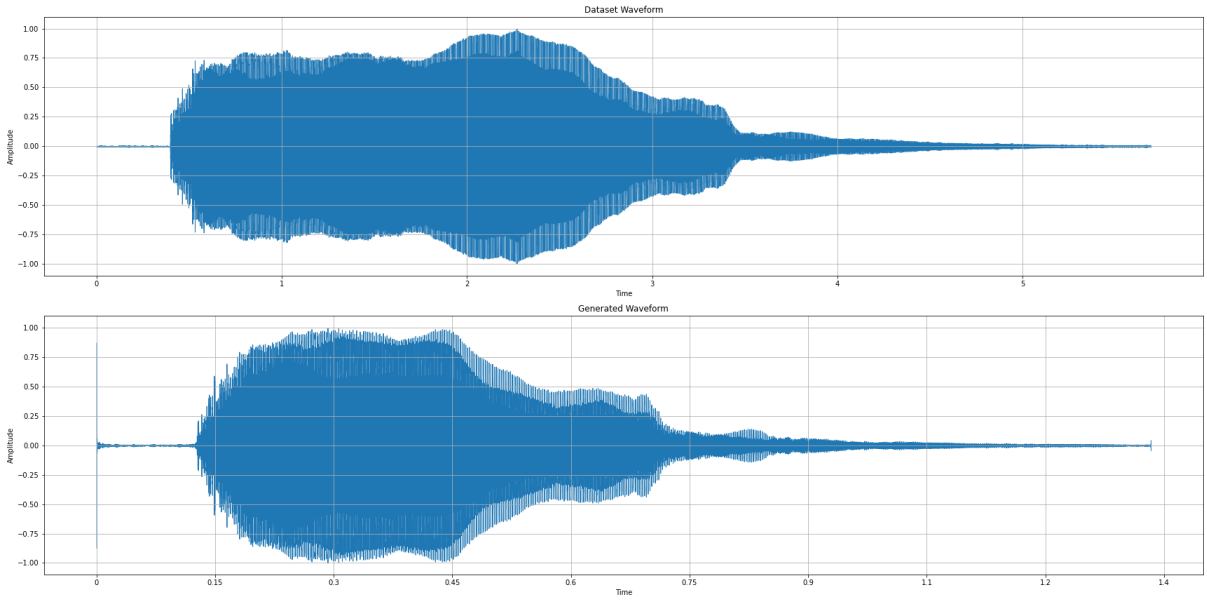

Figure 4.18: Experiment 2 - Comparison between the waveform of the sample from the dataset (top) and the one of a generated sample (bottom).

samples as well. The figures also clearly illustrate the similarities between the real and the fake samples.

The generated files can be categorized into two main types:

- Audio files in which the note from the input file was repeated at irregular intervals.
- Audio files in which the note from the input file was sustained throughout the duration of the file.

In both cases, the note maintained the same pitch as the input file. Therefore, the model successfully generated notes with characteristics similar to the input file but of varying duration.

Notes of various lengths, ranging from 2 to 30 seconds, were generated. It is worth noting that notes with a duration of 2 or 3 seconds did not meet the desired quality standards, whereas those exceeding 3 seconds closely resembled the quality of the dataset file. This suggests that, when the training file has a short duration, it is advisable to generate files of equal or longer duration than the training file.

In [9], the network was primarily trained using files of at least 20 seconds in length, which contained multiple notes. Nevertheless, this experiment showcased

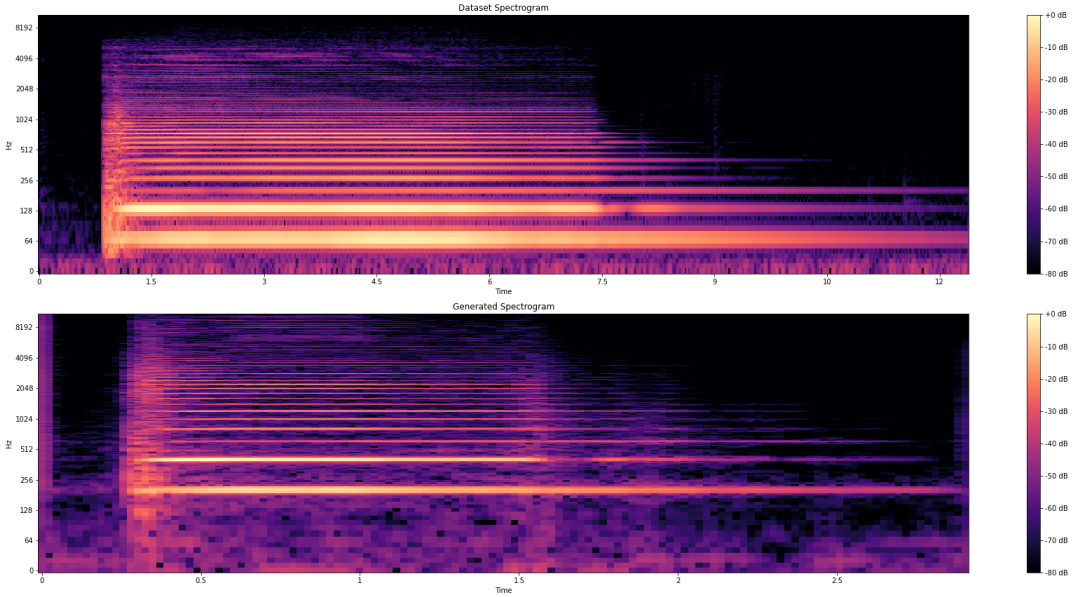

Figure 4.19: Experiment 2 - Comparison between the spectrogram of the sample from the dataset (top) and the one of a generated sample (bottom).

that the network performed exceptionally well even when trained with an harmonic sound of just 6 seconds that consisted of just a single note.

In conclusion, these initial two experiments showed how CAW can effectively generate high-quality files, both harmonic and inharmonic, when trained with a single short-duration audio file.

4.2.3 Experiment 3 - Shorter Inharmonic Sounds

This experiment had 2 main objectives:

1. To determine if the network still produces high-quality samples when further reducing the duration of the dataset file.

2. To assess if there is still noise in the output when using inharmonic audio with reduced high frequencies as input.

To achieve these objectives, a 2-second audio file from the first custom dataset, sampled at 16000 Hz and containing limited high frequencies, was used as dataset.

To determine only the effects of the different file in the dataset, the same training parameters were used as in the previous experiments. Thus:

- The learning rate for both the generator and discriminator was set to 0.0015.
- A learning rate decay of 0.1 was applied.
- The number of epochs for each scale was set to 2000.

In this case, since the input file lasted only 2 seconds, training was performed for 7 scales, spanning sampling frequencies ranging from 2500 Hz to 16000 Hz.

The reconstruction loss values obtained for the various scales are the following:

- Scale 6: 0.0093
- Scale 5: 0.0102
- Scale 4: 0.0165
- Scale 3: 0.0167
- Scale 2: 0.0242
- Scale 1: 0.0236
- Scale 0: 0.0330

Once again, low loss values indicated that convergence was achieved.

The results were excellent. In particular, samples ranging from 1 to 30 seconds in duration were generated, all exhibiting great quality, comparable to that of the training file. All generated samples had similar characteristics to those of the original file, although they were different from a mere copy.
Additionally, unlike Experiment 1 and 2, even the 1-second generated samples displayed high quality.

Moreover, the absence of noise in these samples further confirms that the noise observed in Experiment 1 was linked to the high frequencies present in the dataset file.

4.2.4 Experiment 4 - Long Harmonic Sounds

This experiment served two primary purposes:

1. Evaluating the quality of the samples generated when the network is trained using long-duration harmonic samples.

2. Assessing the quality of the generated samples when the dataset file presents not only single notes but extracts of instrumental music pieces. Additionally, it aims to test the model's robustness in the presence of pronounced reverberation.

To achieve these objectives, the first 15 seconds of a piano music piece at a 16000 Hz sampling rate were chosen as the dataset. This music piece comprises both single notes and chords with also some reverberation.

The same training parameters were retained for this experiment. Thus:

- The learning rate for both the generator and discriminator was set to 0.0015.
- A learning rate decay of 0.1 was applied.
- The number of epochs for each scale was set to 2000.

Since the input file was longer than previous experiments, training was conducted over 16 scales, with sampling frequencies ranging from 320 Hz to 16000 Hz.

Throughout the training process, the values of the loss ranged from 0.0073 to 0.0313 for the different scales. Consequently, it is clear that the model reached the convergence.

The results once again exhibited excellent quality. Notably, the generated samples had lengths ranging from 4 to 60 seconds. In all cases, the quality closely resembled that of the training file, even for files as short as 4 seconds. The only drawback in the results was the presence of the reverberation without the corresponding note that caused it. However, this was a logical outcome since the network does not learn the musical structure of the song, but focuses solely on the features of the input file.

In summary, this network proved once again its effectiveness in generating sounds with characteristics similar to the input audio and with varying duration.

4.2.5 Experiment 5 - Long Inharmonic Sounds

The objective of this experiment was to assess the network's performance when the dataset consisted of an inharmonic long file.

To achieve this, three 4-second files from the first custom dataset were combined utilizing crossfading. Consequently, a 12-second inharmonic sample was produced and employed as the dataset.

The training parameters remained consistent with those of the previous experiments.

Given the length of the input file, training was performed for 16 scales, ranging from 320 Hz to 16000 Hz.

The resulting loss values varied from 0.0146 to 0.0647 across the different scales, indicating that the model converged.

Upon completing the training, files of various length, ranging from 8 to 60 seconds, were generated. As for previous experiments, the quality is comparable to that of the dataset file, and for the longer generated files, the audio exhibits a natural stretching effect. A comparison of the spectrograms of the original file and of a generated samples is reported in Figure 4.20. Once again, the similarities between the fake and real samples are apparent.

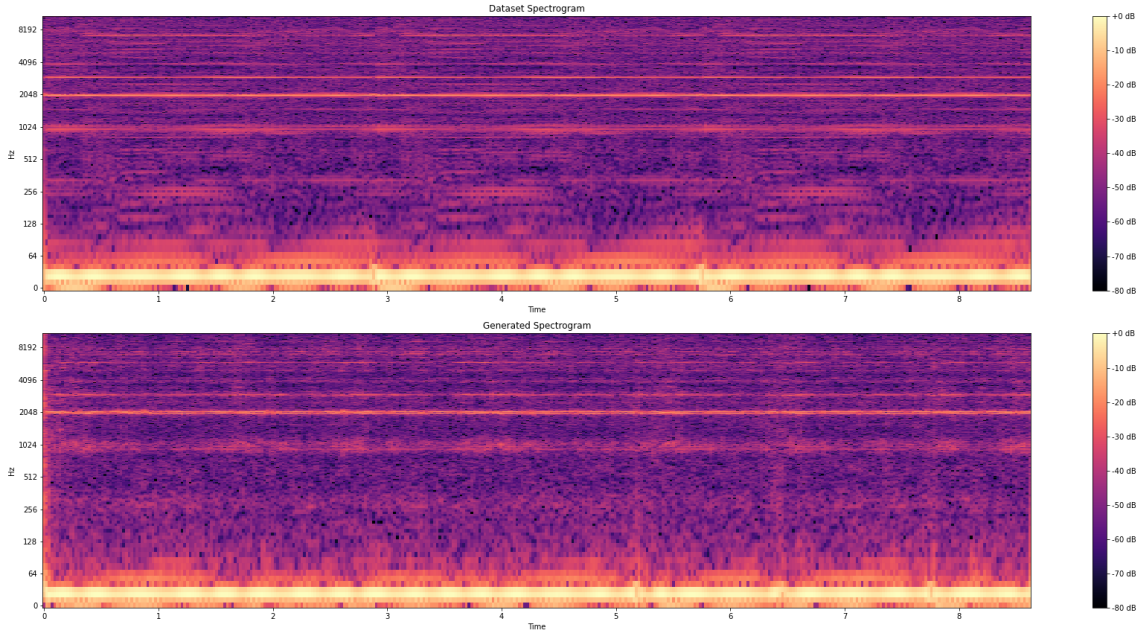

Figure 4.20: Experiment 5 - Comparison between the spectrogram of the sample from the dataset (top) and the one of a generated sample (bottom).

4.2.6 Catch-A-Waveform - Final Conclusions

The experiments involving CAW have been summarized in Table 4.2.

CAW has proven to be an excellent model for sound generation.
In summary, it offers several advantages:

- It can generate variable-length samples of exceptional quality, on par with the original.

EXPERIMENT	DATASET	Lr D	Lr G	EPOCHS	SCALES	Lr DECAY
1	4 sec. sample from First Custom	0.0015	0.0015	2000	5	0.1
2	6 sec. sample from Good Sounds	0.0015	0.0015	2000	15	0.1
3	2 sec. sample from First Custom	0.0015	0.0015	2000	7	0.1
4	15 sec. sample from a piano music piece	0.0015	0.0015	2000	16	0.1
5	12 sec. sample from First Custom	0.0015	0.0015	2000	5	0.1

Table 4.2: Table with a summary of all the parameters and datasets used for each experiment with CAW.

- It can be trained using a single sample, eliminating the need to gather hours of samples for training.
- It can be trained in a relatively short amount of time, with a training duration of less than a day for a sample lasting 25 seconds.
- It is capable of generating a great number of variations of the same sample.
- It can be used to naturally increase or decrease the duration of a sample, without abrupt changes.

However, there are still a few drawbacks to consider:

- The generated samples may exhibit less variations compared to the ones generated by a model trained with a huge dataset.
- It experiences a slight degradation in quality when trained with samples with reverb or a high pitch.

4.3 Combination of SpecGAN and Catch-A-Waveform

The previous experiments conducted with SpecGAN and CAW yielded the following conclusions:

- SpecGAN can generate samples with great variety, but it performs poorly with harmonic sounds. Moreover, it can only generate one-second samples, which restricts its potential applications.
- CAW, on the other hand, exhibits less diversity in the sounds it generates, as it is trained with only one file. However, the quality of the generated samples is excellent, even for harmonic sounds, and they can vary in length.

Subsequently, the upcoming experiments aim to explore potential strategies for creating a pipeline that combines the strengths of both models while mitigating their respective limitations.

4.3.1 Experiment 1 - Training CAW with a Single Audio Generated by SpecGAN

The main objective of this experiment was to test whether it is possible to train Catch-A-Waveform with samples generated by SpecGAN in order to extend their duration. The main problem with SpecGAN samples, in fact, is that they last only 1 second, limiting their possible applications.

In this experiment, CAW was trained using a sample generated in Experiment 3 of the SpecGAN experiments section. Specifically, a sample generated from the model after 200 epochs was selected.

The following parameters were configured for the training process of Catch-A-Waveform:

- The generator and discriminator learning rates were set to 0.0015;
- The learning rate decay was set to 0.1.
- The number of epochs for each scale was set to 2000.

Due to the short duration of the file selected for the dataset, the training process was carried out for 7 different scales, covering sampling rate values from 2500 Hz to 16000 Hz. The obtained loss values ranged from 0.0090 to 0.0178, indicating model convergence.

The obtained results exhibited quality similar to that of the input file, as one can see in Figure 4.21 and Figure 4.22.

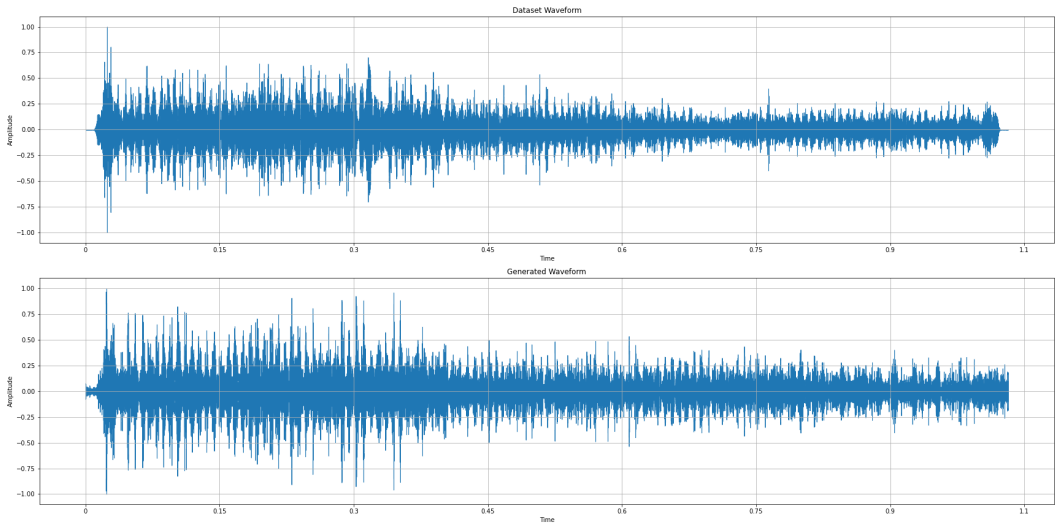

Figure 4.21: Experiment 1 - Comparison between the waveform of the sample from the dataset (top) and the one of a generated sample (bottom).

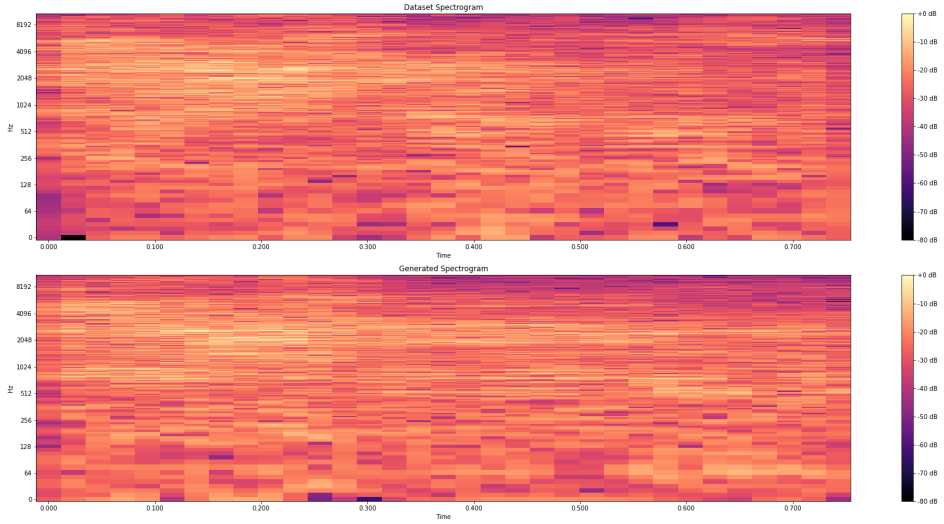

Figure 4.22: Experiment 1 - Comparison between the spectrogram of the sample from the dataset (top) and the one of a generated sample (bottom).

Notably, samples of diverse duration, ranging from 4 to 30 seconds, were generated, and in all instances, the sound was naturally extended.
However, the variations of the output were relatively limited since the model was trained with just one samples of one second.

In summary, Catch-A-Waveform demonstrated its capability to increase the duration of the files generated by SpecGAN. The next experiments will explore different methods of training Catch-A-Waveform with SpecGAN-generated files to introduce greater variation in the produced samples.

4.3.2 Experiment 2 - Inpainting and Unconditional Generation

The objective of this experiment was to establish a pipeline for addressing the following weaknesses of SpecGAN and CAW:

- The limited variety of samples generated by CAW when trained on a single sample generated by SpecGAN.
- The short duration of samples generated by SpecGAN, which severely limits their applicability.

The concept behind this experiment involved concatenating multiple samples generated by SpecGAN in a seamless manner and subsequently training CAW with the resulting concatenated sample. The full process is depicted in Figure 4.23.

For the concatenation process, three significantly different samples generated during Experiment 3 with SpecGAN were selected:

- One sample generated after 80 epochs.
- One sample generated after 200 epochs.
- One sample generated after 280 epochs.

To concatenate them in a natural way, multiple steps were followed:

1. Initially, the three samples were concatenated with 0.8 seconds of silence between them;
2. Subsequently, CAW was trained for the inpainting task using the resulting concatenated sample from the first step. This approach allowed for the filling of the 0.8-second silence intervals using CAW.

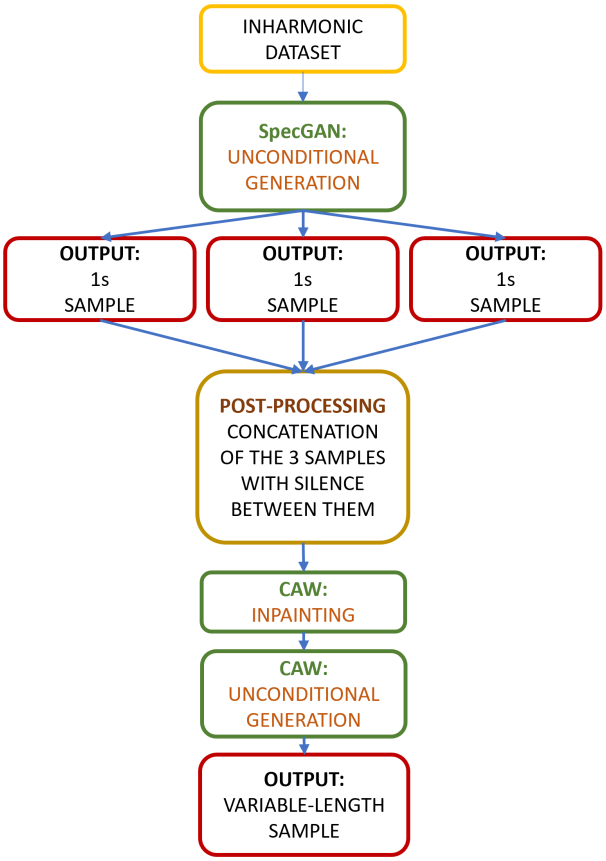

Figure 4.23: Experiment 2 - Representation of the full process.

The following training parameters were employed for the inpainting task:

- A learning rate of 0.0015.
- A learning rate decay of 0.1.
- 2000 epochs for each scale.

Eight scales were considered during training, with sampling rate values ranging from 2000 Hz to 16000 Hz.

The resulting loss values fell within the range of 0.0001 to 0.0012 for the different scales, indicating the convergence of the model.

So, at this stage, the file contains the three samples generated by SpecGAN, with two 0.8-second gaps filled by CAW.

An observation from the result of the inpainting is that the way the silence intervals were filled was unnatural due to abrupt transitions. This can be attributed to two primary reasons

- The three generated samples exhibited considerable differences, making it challenging to concatenate them seamlessly during inpainting.

- A 0.8-second silence period between the three samples is insufficient for achieving a natural transition. Hence, in subsequent experiments, longer silence intervals were tested.

At this point, therefore, a nearly 5-second sample was obtained that had far more features than the individual samples generated by SpecGAN. Consequently, this sample was used to train CAW, but this time for the task of unconditional generation. The same training parameters as the previous experiments were utilized:

- A learning rate of 0.0015.

- A learning rate decay of 0.1.

- 2000 epochs for each scale.

This time, nine scales were considered for training, with sampling rate values ranging from 1600 Hz to 16000 Hz.

Loss values obtained for this phase ranged from a minimum of 0.0054 to a maximum of 0.0274, depending on the scale.

After the training process was finished, samples of different duration, ranging from 1 to 30 seconds, were generated.
These generated files exhibited greater diversity than those obtained in Experiment 1, in which only one sample generated by SpecGAN was used. Moreover, the quality was comparable to that of the file used for the training. The only notable drawback was the presence of occasional abrupt transitions in the generated samples. However, this is due to the fact that these sudden changes were also present in the inpainted file used for training. Consequently, by refining the concatenation and inpainting processes, the final output is expected to improve as well.

In summary, this experiment demonstrated the effectiveness of combining SpecGAN and CAW in a unified pipeline to produce a final sample that exhibits both the diversity of SpecGAN-generated samples and the variable duration characteristic of CAW-generated samples.

4.3.3 Experiment 3 - Longer Silence Intervals

The goal of this experiment was to improve the pipeline proposed in Experiment 2 by extending the duration of the silence intervals during the concatenation and inpainting phases.

In particular, the duration of the silence intervals was increased from 0.8 seconds to 1.5 seconds. To assess the impact of this change exclusively, the same three samples were selected, and identical training parameters were used for the inpainting task. Additionally, training was conducted for 8 scales, consistent with Experiment 2.

The loss values obtained in this instance ranged from 0.00001 to 0.0007 across different scales, all of which are smaller than those observed in Experiment 2. Moreover, the results of inpainting exhibited marked improvement, leading to a more natural filling of the silent intervals within the samples. This improvement can indeed be attributed to the extended time available to the network for merging various samples, resulting in fewer abrupt transitions.

Subsequently, the inpainted file was used to train CAW for the unconditional generation task, using once again the same parameters as those in Experiment 2.

In this case, the loss values ranged from 0.0090 to 0.250, which are similar values to those obtained in Experiment 2.

To evaluate the performance of the network, samples of different lengths, ranging from 4 to 60 seconds, were generated. The quality of these samples exceeded that of Experiment 2, as fewer sudden transitions were present. Thus, as expected, the improvements implemented during the concatenation and inpainting phases resulted in improved output files at the end of the full process.

In conclusion, this pipeline proved to be very promising, particularly with the increased length of the silent intervals during the concatenation.

4.3.4 Combination of SpecGAN and Catch-A-Waveform - Final Conclusions

The experiments involving the combination of SpecGAN and CAW have been summarized in Table 4.3.

The combination of SpecGAN and CAW effectively merges the strengths of both models while addressing their respective weaknesses and limitations.

Specifically, the experimental pipeline tested in Experiments 2 and 3 has proven to be successful in mitigating the limitations of both SpecGAN and CAW:

- While SpecGAN is limited to generating samples of only 1 second in duration, the combination with CAW enables the generation of samples with arbitrary

EXPERIMENT	PIPELINE
1	Generate a sample with SpecGAN and train CAW with that sample
2	Generate 3 samples with SpecGAN, concatenate them with 0.8 sec. silence intervals, inpaint the result with CAW and train CAW with the inpainted sample
3	Generate 3 samples with SpecGAN, concatenate them with 1.5 sec. silence intervals, inpaint the result with CAW and train CAW with the inpainted sample

Table 4.3: Table with a summary of all the parameters and datasets used for each experiment with the combination of SpecGAN and CAW.

duration.

- While CAW may not inherently produce a wide range of variations, training it with a combination of multiple samples generated by SpecGAN results in a final output with increased variety.

4.4 UNAGAN

The last model that has been tested is UNAGAN.
So far, the only effective method for generating high-quality harmonic sounds has been through CAW, which, however, does not allow for too much variation since it's trained on a single sample.

As a result, even though the primary objective of this thesis does not revolve around generating harmonic sounds, UNAGAN was tested with the goal of generating harmonic sounds with great variety. Unlike CAW, UNAGAN needs a substantial dataset for training, so the resulting output samples are likely to exhibit more variety as well.

Furthermore, in comparison to SpecGAN, which employed a lightweight architecture and relied on the Griffin-Lim algorithm for inverting spectrograms, UNAGAN features a more intricate architecture and utilizes MelGAN for the inversion process. These attributes should promote the generation of harmonic sounds without introducing any artifacts.

4.4.1 Experiment 1 - Harmonic Samples

The aim of this experiment was to test whether UNAGAN could generate inharmonic samples of better quality than those generated by SpecGAN

A dataset comprising 7930 samples from "NSynth" was chosen. Each sample had a 4-second duration and consisted of a single piano note, with a sampling rate of 16000 Hz.

Out of these samples, 230 were designated for the validation set, while the rest was used for training.
Hierarchical training with cycle regularization was employed, setting the initial learning rate at 0.0001 and the batch size at 25.

Training was conducted for 200 epochs, and convergence was reached after 184.

To invert the resulting spectrograms, a pre-trained MelGAN checkpoint trained on piano segments extracted from the" MAESTRO" dataset was employed.

The samples obtained exhibited significantly higher quality compared to those derived from SpecGAN. Some generated samples demonstrated excellent quality, while others exhibited slightly lower quality. Regardless, the diversity of the generated samples surpassed that of SpecGAN and CAW. Notably, samples of good quality were already generated after just 70 epochs.

As a result, UNAGAN proved to be better than SpecGAN in generating harmonic samples, even when utilizing a smaller dataset.

4.4.2 UNAGAN - Final Conclusions

The experiment involving UNAGAN has been summarized in Table 4.4.

EXPERIMENT	**DATASET**	**Lr D**	**Lr G**	**EPOCHS**	**BATCH SIZE**
1	Subset of NSynth	0.0001	0.0001	200	25

Table 4.4: Table with a summary of all the parameters and datasets used for the experiment with UNAGAN.

UNAGAN has demonstrated its effectiveness in generating harmonic samples of variable lengths. However, the quality experiences a slight decrease as the length of the generated samples increases. Consequently, as done with SpecGAN, UNAGAN can also be combined with CAW if longer samples are required.

4.5 Denoising

The final experiments were focused on attempting to remove the noise from the samples generated by SpecGAN.

Various strategies were tested:

- Training CAW for noise removal.
- Applying the peak comb filter to the generated samples.
- Applying the Wiener filter.
- Applying a custom filter inspired by the Wiener filter.
- Applying the Kalman filter.

4.5.1 CAW

The initial experiment aimed at removing the noise from the samples generated by SpecGAN involved the use of Catch-A-Waveform (CAW).

Indeed, CAW can be trained on a single sample to enhance its quality and eliminate unwanted noise, in addition to its use for unconditional generation.
For this experiment, a sample generated in Experiment 3 of the SpecGAN section was selected as the input. The training was conducted over 6 scales, with each scale lasting for 2000 epochs.

However, the resulting sample closely resembled the input signal, exhibiting nearly imperceptible audible differences. This suggests that training CAW with only a one-second sample is insufficient for effective noise removal.

Even when comparing the spectrogram of the original sample with the generated one, the differences are minimal, as depicted in Figure 4.24.

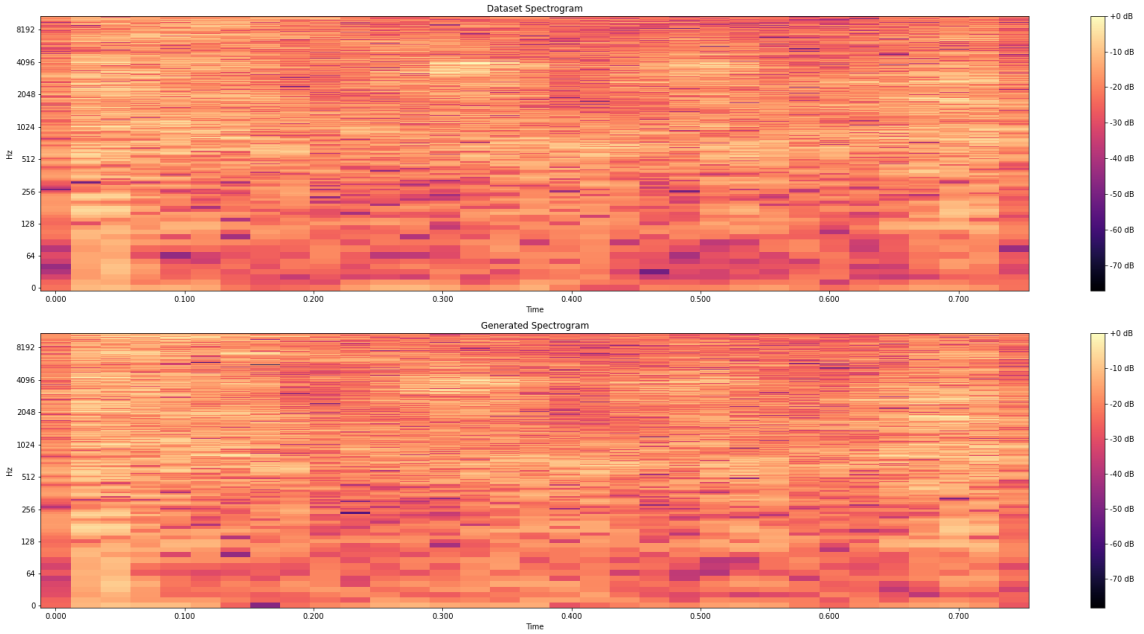

Figure 4.24: Denoise Experiment - Comparison between the original sample (top) and the denoised version (bottom) using CAW.

4.5.2 Comb Filter

The second denoising experiment involved the utilization of the peak comb filter, depicted in Figure 4.25.

The filter parameters were set as follows:

- The order was set to 40.
- The bandwidth was set to 0.01.

The input sample for denoising was the same as in the previous experiment. However, in this case, the filtered audio significantly differed from the original, indicating that it also attenuated frequencies that were not noise. This is evident from the substantial disparities in the spectrograms, as shown in Figure 4.26.
Various parameter configurations for the filter were tested but yielded similar results.

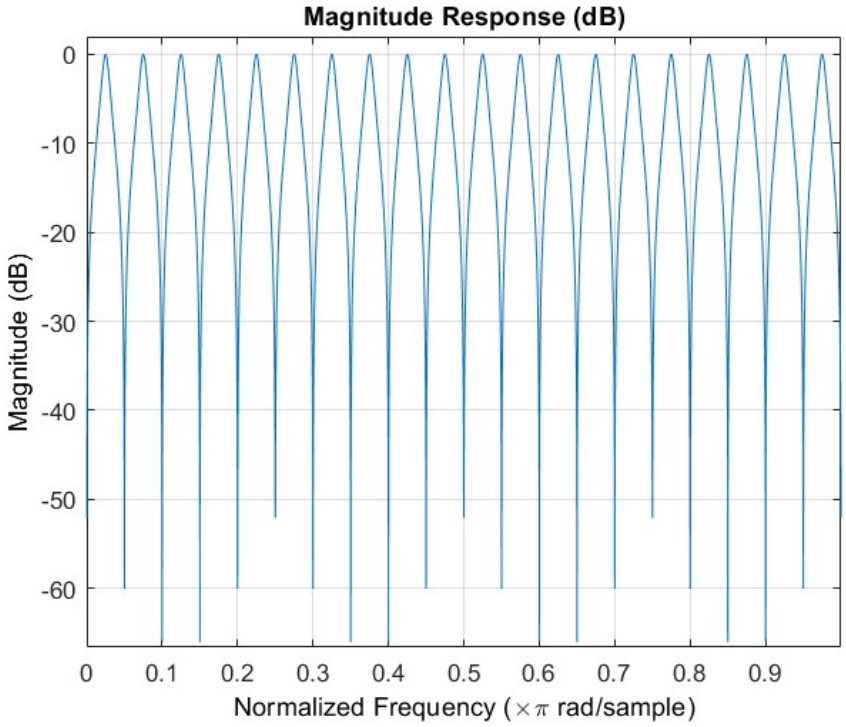

Figure 4.25: Comb filter used for denoising.

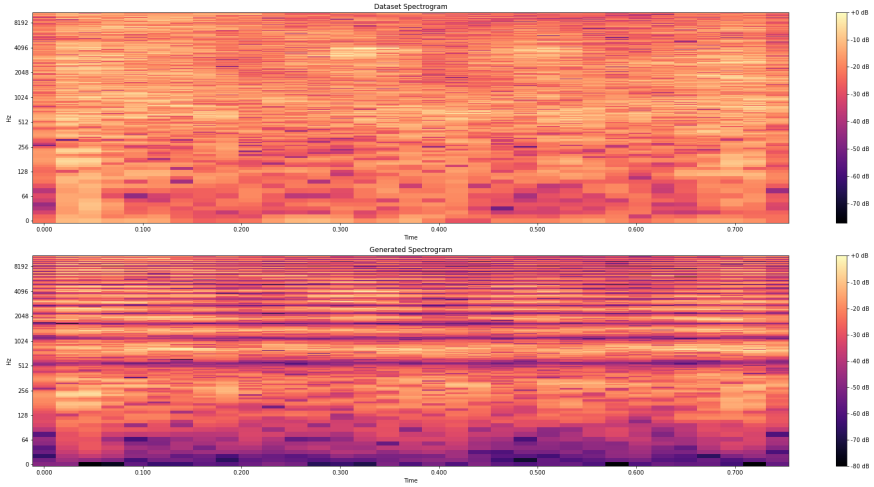

Figure 4.26: Denoise Experiment - Comparison between the original sample (top) and the denoised version (bottom) using the comb filter.

4.5.3 Wiener Filter And Custom Filter

The Wiener filter is designed to enhance the quality of a signal.

In particular, it operates by computing a statistical estimation of a specific segment of the signal, which consists solely of noise. This estimation is then used to filter the signal and produce the final output.

In the audio domain, the Wiener filter is primarily employed for tasks such as noise reduction, audio restoration, and speech enhancement.

However, when attempting to denoise an audio sample generated by SpecGAN, it did not yield satisfactory results.

As a result, an alternative custom filter, based on the same concept as the Wiener filter, was tested. To be specific, the audio file underwent a series of filtering steps:

- First, the magnitude spectrogram of the audio file was computed.
- Then, the noise profile was estimated by taking the mean of the first columns of the magnitude spectrogram. This approach assumes that the initial portion of the audio file contains isolated noise.
- Finally, the filter was applied as follows: a mask was generated by comparing the noise estimation with the original signal. Subsequently, this mask was multiplied by the original signal. In this process, magnitudes greater than the noise estimation were retained, while the rest were attenuated.

The resulting sample obtained through this procedure closely resembled the original one, while effectively reducing noise.

To assess the success of the denoising process, a third file was chosen directly from the dataset, sharing similar acoustic characteristics with the SpecGAN-generated sample. When comparing the spectrograms of the three samples, the denoised sample exhibited a significantly closer resemblance to the dataset sample, as opposed to the one generated by SpecGAN, as depicted in Figure 4.27. This observation indicates that the custom filter successfully reduced the noise while preserving the acoustic characteristics of the original sample.

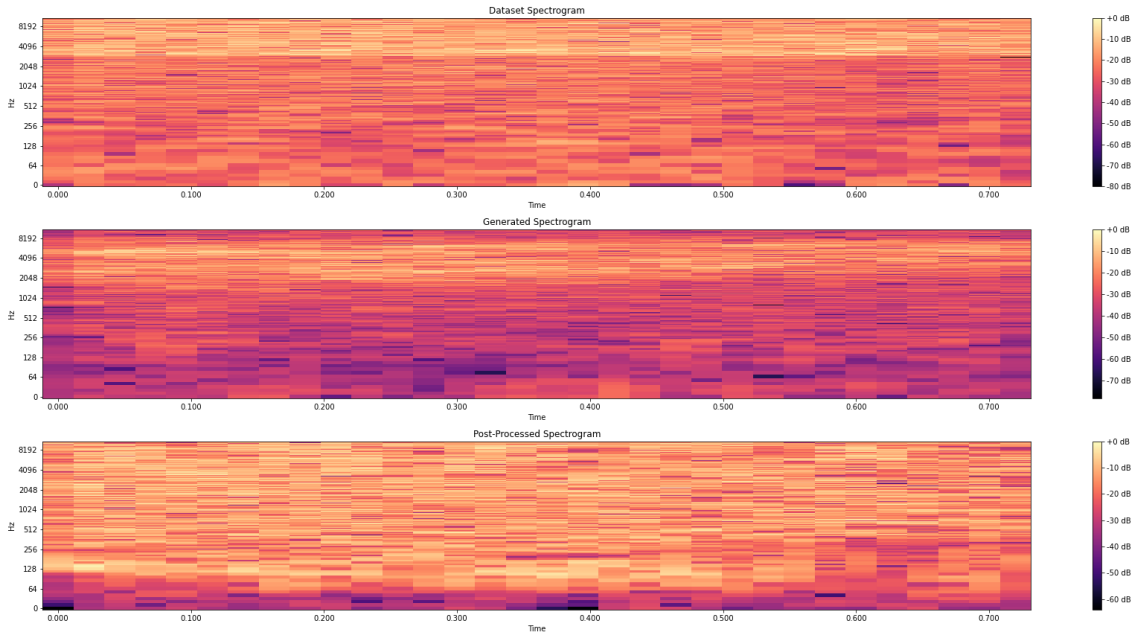

Figure 4.27: Denoise Experiment - Comparison between the sample from the dataset (top), the original generated sample (center) and the generated denoised version (bottom) using the custom Wiener filter.

4.5.4 Kalman Filter

The Kalman filter is a tool used to eliminate noise from audio samples.

It works by continuously updating an estimate of the system's state. This is achieved by combining predictions of future states with observations of the current state. Its primary advantage over the Wiener filter is that it doesn't necessitate a segment of the audio sample that contains only noise.

Its main applications in the audio domain are speech enhancement and noise reduction.

For this experiment, the following parameters have been configured:

- State vector: this represents the true audio signal at each step. In this case, it is one-dimensional and consists solely of the signal's amplitude.

- State Transition model: this matrix describes how the state changes over time. In this case, it is assumed that the audio signal doesn't change significantly over time, so it is set to 1;

- Measurement Model: this matrix defines how the state is mapped to measurements. For audio denoising, it is set to 1 since the audio signal is measured directly.

As in previous experiments, a noisy sample generated by SpecGAN has been selected for testing the filter.

The filtered audio still retains the desired acoustic features of the original sample while reducing some of the noise. Figure 4.28 displays the original noisy waveform and the filtered waveform.

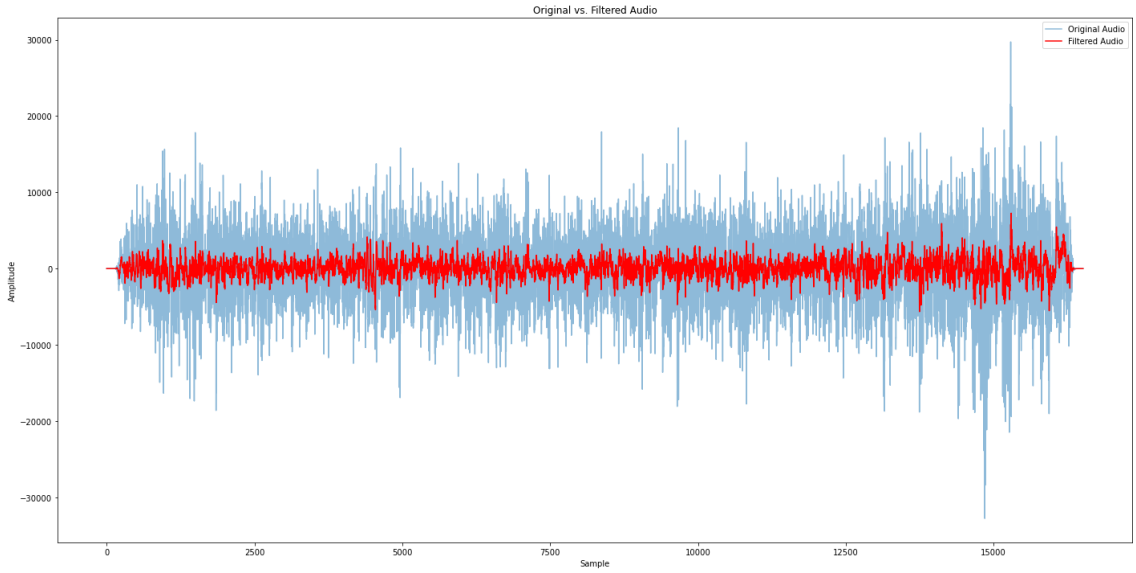

Figure 4.28: Denoise Experiment - Comparison between the original generated sample and the generated denoised version using the Kalman filter.

Chapter 5

Final Results

5.1 Final Pipeline

From the experiments that were carried out, the following results emerged:

- SpecGAN: it works well with inharmonic samples, generating samples with great variety. However:
  - It doesn't perform well with harmonic samples.
  - The generated samples may be affected by noise.
  - It's limited to generating 1-second samples.
- CAW: it works really well with both harmonic and inharmonic samples and can generate clean variable-length samples. However, since it's trained with just one sample, the variety is limited.
- UNAGAN: it works well with harmonic samples and can generate variable-length samples. However, when increasing the length of the generated samples, the quality decreases.
- Combination of CAW and SpecGAN/UNAGAN: by combining CAW with another model, it's possible to obtain all the advantages of the two models while compensating for their limitations.
- Both the custom Wiener filter and the Kalman filter obtain good results when trying to denoise the samples generated by SpecGAN.

The proposed pipeline, as depicted in Figure 5.1, combines all three models in a modular and flexible way, allowing to exploit the strengths of each model.

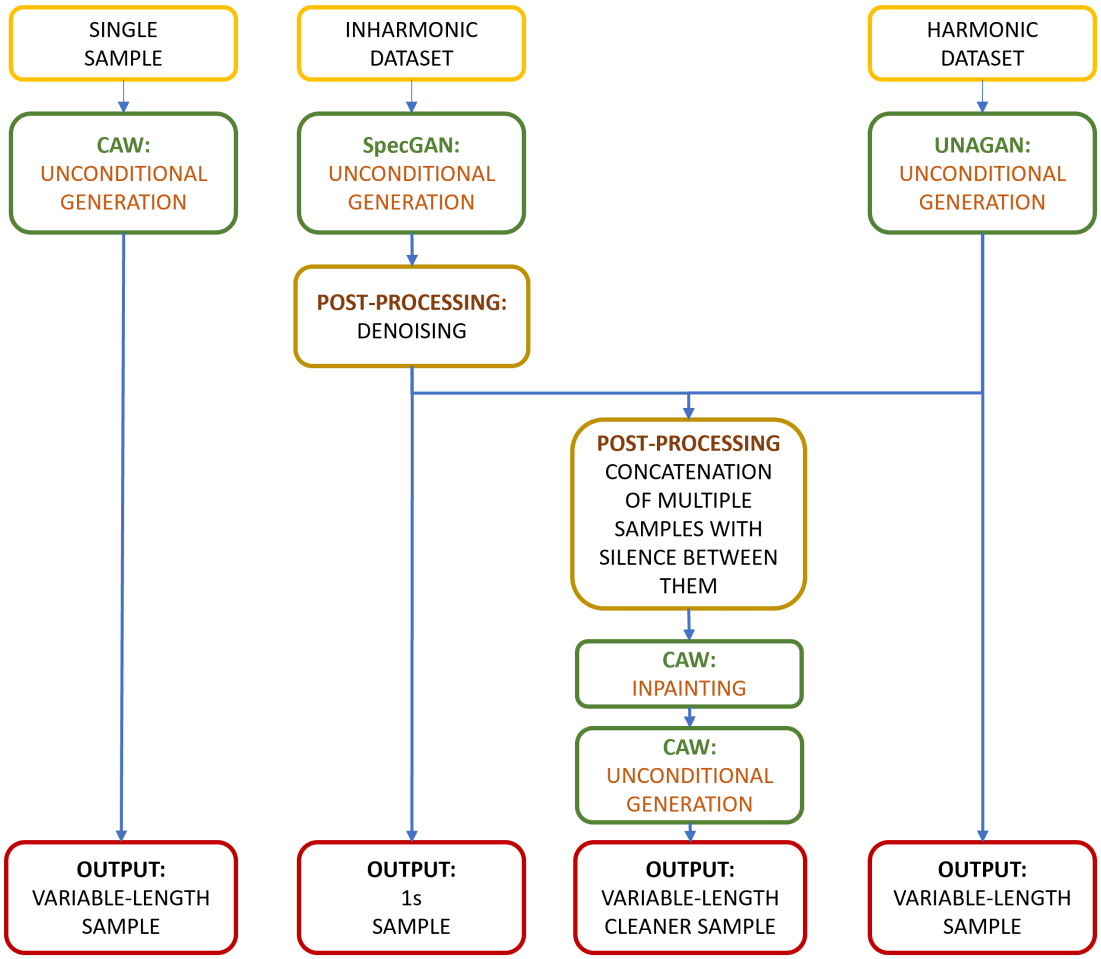

Figure 5.1: Scheme of the proposed pipeline.

The direction in the pipeline to be followed is determined by the type of the dataset and by the desired length and variety of the output samples:

- If the dataset is composed of just one single reference sample, whether harmonic or inharmonic, it's possible to train CAW directly with that sample for unconditional generation. In this way, it is possible to generate variable-length samples with quality on par with the reference sample.

- If the dataset is composed of a collection of inharmonic sounds, it can be used to train SpecGAN for unconditional generation. Post-processing (custom Wiener filter or Kalman filter) can be applied to the generated samples to remove the noise. If the goal is to generate 1-second short audio files, this step is sufficient. However, for longer files, additional steps are required, which involve a combination of SpecGAN and CAW (as described in Experiment 3 of the section dedicated to the combination of SpecGAN and CAW):
  - First, multiple samples are generated with SpecGAN.
  - Next, these samples are concatenated with 1.5 seconds of silence between them
  - Then, the concatenated file is used to train CAW for inpainting.
  - Finally, the inpainted file is used to train CAW for unconditional generation.

  In this way, it's possible to generate variable-length cleaner inharmonic samples.
- If the dataset is composed of a collection of harmonic sounds, UNAGAN can be used for unconditional generation. To generate short audio files, this is sufficient. However, for longer files, UNAGAN can be combined with CAW, following the same approach as described for SpecGAN.

This pipeline offers several advantages:

- Modularity: since this pipeline is composed of independent blocks, it can be improved by enhancing a single block. For example, if a new model can generate cleaner inharmonic sounds, it can replace SpecGAN while maintaining the rest of the pipeline as it is.
- Flexibility: this pipeline allows the generation of a wide variety of sounds, both harmonic and inharmonic, with fixed-short length or variable-length.

However, its main drawback is that, if long inharmonic sounds are needed, three different training processes must be performed:

- SpecGAN for unconditional generation.
- CAW for inpainting.
- CAW for unconditional generation.

Nevertheless, both SpecGAN and CAW have very short training times, so the total required time needed for training is still less than state-of-the-art models for audio generation, such as auto-regressive models.

5.2 Survey Results

To evaluate the performance of the tested models, a survey was prepared. This survey presented a first section with the following samples:

- 2 inharmonic samples extracted from the First Custom Dataset:
  - Dataset sample 1.
  - Dataset sample 2.
- 3 samples generated with SpecGAN:
  - SpecGAN sample 1, generated in Experiment 3 after 320 epochs.
  - SpecGAN sample 2, generated in Experiment 6 after 160 epochs.
  - SpecGAN sample 3, generated in Experiment 8 after 200 epochs.
- 4 samples generated with CAW:
  - CAW sample 1, generated in Experiment 3.
  - CAW sample 2, generated in Experiment 2.
  - CAW sample 3, generated in Experiment 5.
  - CAW sample 4, generated in Experiment 4.
- 2 samples generated with UNAGAN:
  - UNAGAN sample 1.
  - UNAGAN sample 2.

For each of these samples, participants were required to evaluate its quality and the emotions it conveyed. Specifically:

- For quality assessment, participants were asked to assign a score between 0 and 10.
- For the evaluation of the emotions, participants were asked to assign a value from 0 to 10 for each of the following emotions conveyed by the sound:
  - Happiness.
  - Sadness.

- Fear.
- Anger.

Following this, a second section included the following questions:

1. Participants were presented with two samples generated using the combination of SpecGAN and CAW as described in Experiments 2, and the one described in Experiment 3. They were asked to select the sample with more naturalness, referring to a sound in which the various parts are connected seamlessly. The objective of this question was to determine whether the use of 1.5-second silence intervals during the concatenation in Experiment 3 resulted in generated samples with fewer abrupt changes compared to those obtained using 0.8-second silence intervals, as seen in Experiment 2.

2. Participants were presented with three samples and tasked with ranking them based on their quality. The samples included:

   - A noisy sample generated with SpecGAN.
   - The same sample filtered with the custom Wiener filter.
   - The same sample filtered with the Kalman filter.

   The aim of this question was to assess whether the two filters were capable of denoising the sample.

Finally, the last section of the survey included a few questions about age, gender, hearing problems and participants' idea of the survey's purpose. For online data collection, the software PsyToolkit was used[47][48]. The survey was completed by 29 participants. Of these, 31.03% were female, and 68.97% were male. The average age was 29.55, with a standard deviation of 11.

The results concerning the quality of the samples in the initial section of the survey are depicted in Figure 5.2. This figure illustrates the average perceived quality of each sample. Additionally, Figure 5.3 showcases the mean quality of all samples generated by each model.

The results indicate that the real samples extracted from the dataset achieved the lowest average quality compared to all other samples.
In contrast, CAW emerged as the model with the best results, followed by UNAGAN and SpecGAN.

Regarding SpecGAN's results, it is noteworthy that the first sample, generated with the reference training parameters, achieved the highest average score. Instead,

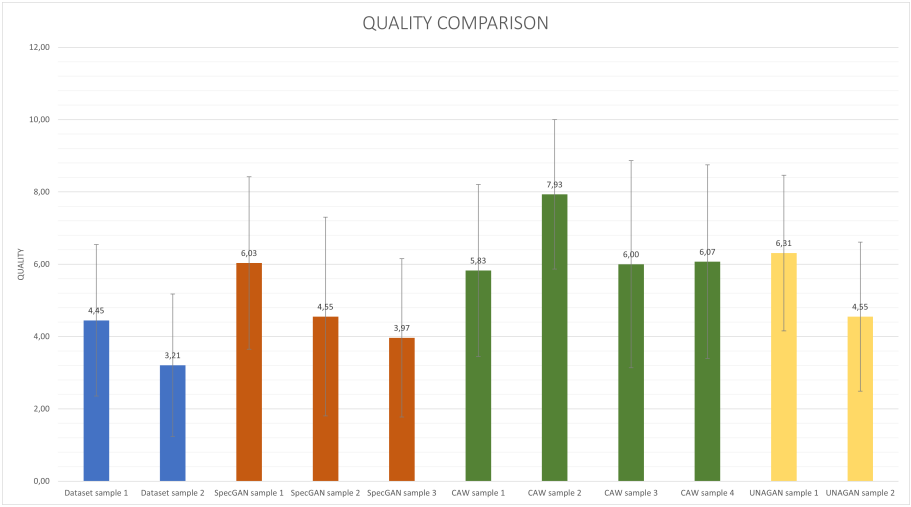

Figure 5.2: Comparison of the average perceived quality of each sample. The vertical bars represent the standard deviation.

the other two samples, which were both generated with models integrating the comb filter in the training phase, obtained lower scores. This suggests that, while incorporating the comb filter during the training phase may enhance sample variety, as discussed in the experiments section, it also leads to a perceived decrease in quality.

Concerning CAW's results, samples 2 and 4, both of which were harmonic, achieved the best scores. Notably, sample 2, featuring a single cello note, obtained the overall highest average quality. Instead, sample 4, which consisted in a segment of a piano music piece with reverberation, yielded lower results. This suggests that the presence of reverberation led to a perceived decrease in quality.
Instead, samples 1 and 3, both inharmonic, obtained very similar scores, which were lower than those of the harmonic samples.

Finally, concerning UNAGAN's results, it can be observed that, while both samples achieved scores generally higher than those of SpecGAN and of the real samples, their quality exhibited significant inconsistency, with an almost 2-point difference in their average scores.

The results concerning the evaluation of the emotions conveyed by the sounds are depicted in Figure 5.4. Notably, the results reveal that fear was the predominant emotion conveyed by the samples. However, there were a few exceptions, particularly in the case of the two harmonic samples generated by CAW (sample 2 and sample 4). Specifically, CAW sample 2 predominantly conveyed sadness, while CAW sample 4 predominantly conveyed happiness. In general, while fear emerged as the most

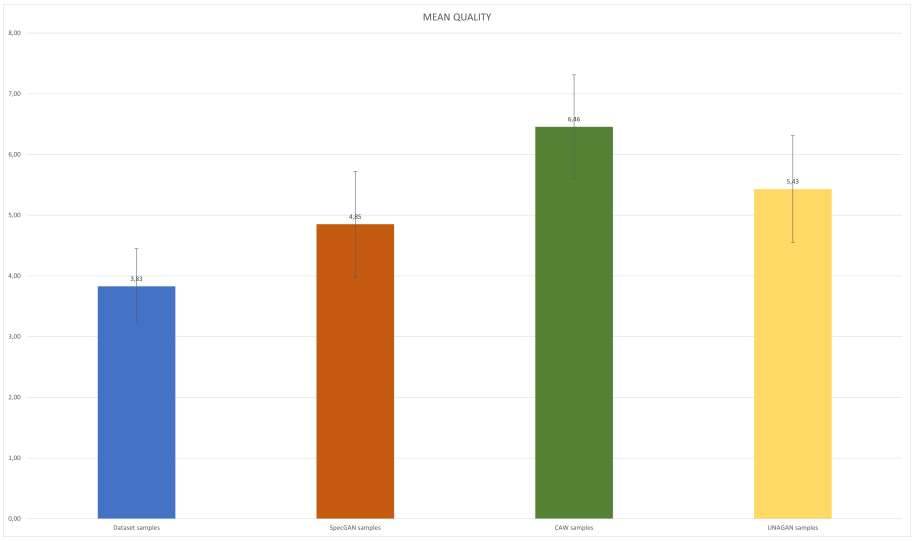

Figure 5.3: Comparison of the mean quality of each model. The vertical bars represent the standard deviation.

frequently conveyed emotion in inharmonic sounds, harmonic sounds, such as those generated by UNAGAN, commonly conveyed sadness as well.

The results regarding the preference for the different duration of the silence intervals used during the concatenation and the inpainting are reported in Figure 5.5. The results indicate that longer silence intervals during concatenation resulted in generated samples with smoother transitions and fewer abrupt changes. In fact, nearly three-fourths of the participants preferred samples that utilized longer silence intervals.

Finally, the denoising results are illustrated in Figure 5.6. Notably, the custom Wiener filter was ranked first most frequently. Additionally, when evaluating the overall score, as depicted in Figure 5.7, the custom Wiener filter achieved the highest performance, with the Kalman filter following closely behind.

Examples of generated samples and the samples selected for the survey can be found in the following repository:
https://github.com/EdoardoBasti/Sound-Generation-using-GAN-Models.git

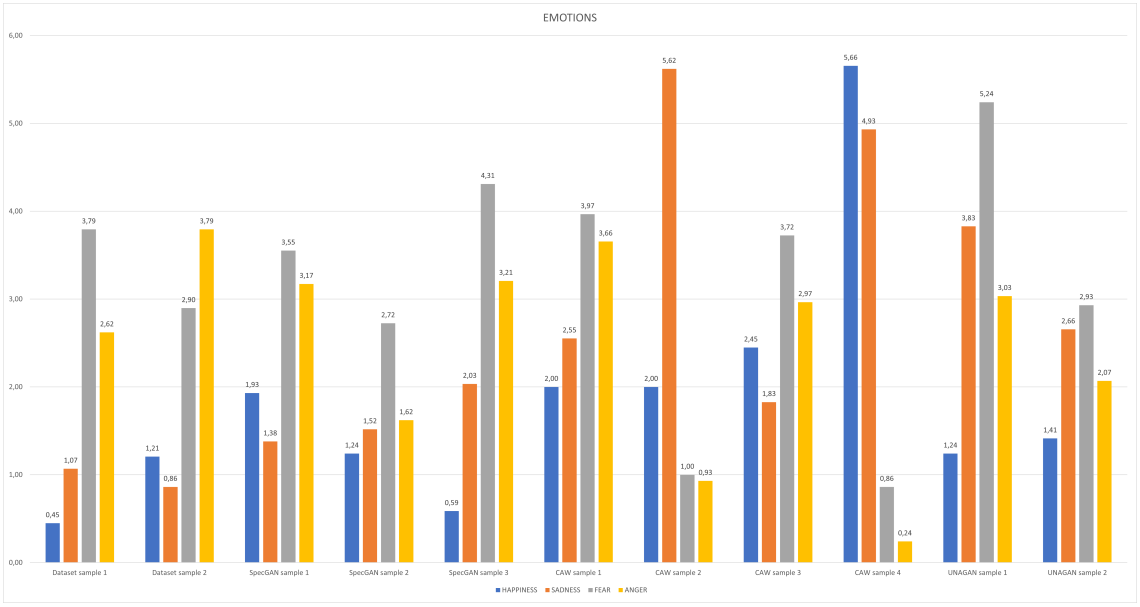

Figure 5.4: Comparison of the emotions conveyed by each sample.

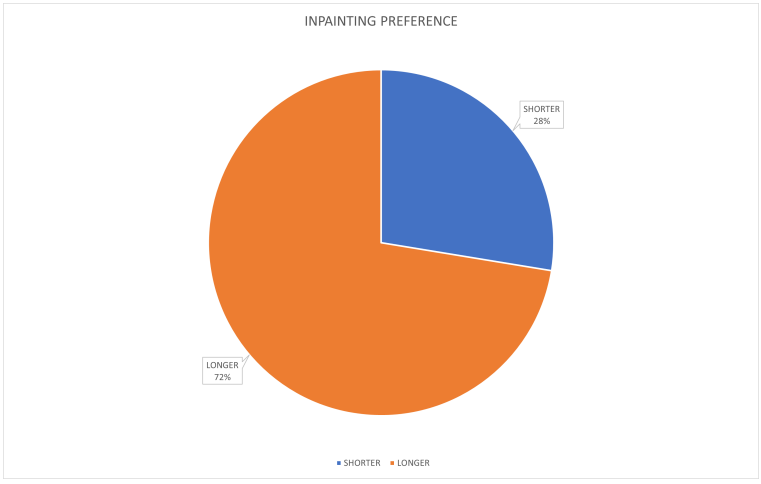

Figure 5.5: Preference of the different length of silence for concatenation and inpainting.

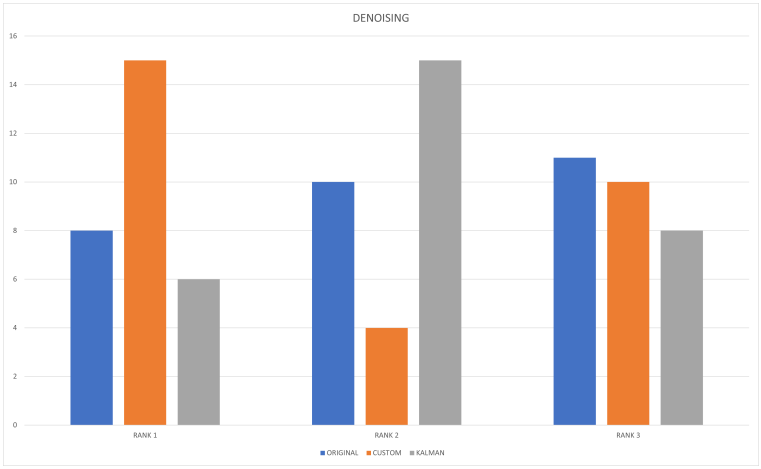

Figure 5.6: Ranking of the original noisy sample, the sample denoised with the custom Wiener filter, and the sample denoised with the Kalman filter. Each bar represents the number of times that sample has been ranked in that position.

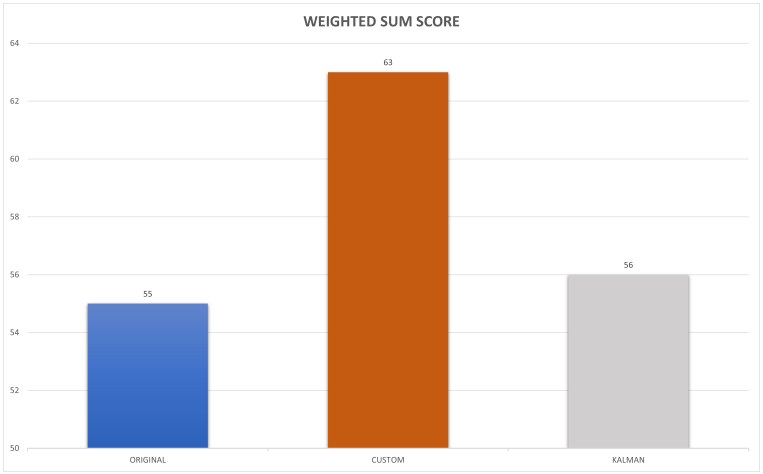

Figure 5.7: Weighted sum score of the original noisy sample, the sample denoised with the custom Wiener filter, and the sample denoised with the Kalman filter. The weighted sum is calculated by multiplying the number of times a sample has been ranked first by 3, adding the product of the times it was ranked second by 2, and finally, adding the times it was ranked last.

Chapter 6

Conclusions

This master thesis proposes an effective pipeline for unconditional sound generation, comprising multiple GAN models and post-processing techniques.
Specifically, SpecGAN is utilized to generate 1-second inharmonic samples with significant variety, Catch-A-Waveform is employed to generate both harmonic and inharmonic variable-length samples with limited variety, and UNAGAN is utilized for generating short harmonic samples with considerable diversity.
These three models can be combined to create a flexible and modular system, taking advantage of the strengths of each model while addressing their weaknesses. This enables the generation of variable-length harmonic and inharmonic samples.
The samples generated with this pipeline were evaluated by 29 participants through a survey. The results indicated that the generated samples exhibited a higher perceived quality compared to the real samples.

6.1 Future Improvements

There are two primary potential improvements for the proposed pipeline.
The first involves adding conditioning to SpecGAN and UNAGAN. This modification would enable the generation of samples based on specified textual descriptions, significantly expanding the potential applications of the proposed pipeline.
The second improvement consists in training UNAGAN with a larger harmonic dataset, as the limited available data likely constrained the performance of this model.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[4] Gemma Calvert, Charles Spence, and Barry E Stein. *The handbook of multisensory processes*. MIT press, 2004.

[5] Mark Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10(4):14–27, 1986.

[6] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

[7] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[9] Gal Greshler, Tamar Shaham, and Tomer Michaeli. Catch-a-waveform: Learning to generate audio from a single short example. *Advances in Neural Information Processing Systems*, 34:20916–20928, 2021.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[11] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample. *Advances in Neural Information Processing Systems*, 33:16761–16772, 2020.

[12] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[13] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[14] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *arXiv preprint arXiv:2007.00074*, 2020.

[15] Tun-Min Hung, Bo-Yu Chen, Yen-Tung Yeh, and Yi-Hsuan Yang. A benchmarking initiative for audio-domain music generation using the freesound loop dataset. *arXiv preprint arXiv:2108.01576*, 2021.

[16] Shulei Ji, Jing Luo, and Xinyu Yang. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *arXiv preprint arXiv:2011.06801*, 2020.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[20] Ji-Hoon Kim, Sang-Hoon Lee, Ji-Hyun Lee, and Seong-Whan Lee. Fre-gan: Adversarial frequency-consistent audio synthesis. *arXiv preprint arXiv:2106.02297*, 2021.

[21] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

[22] Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

[23] Sang-Hoon Lee, Ji-Hoon Kim, Kang-Eun Lee, and Seong-Whan Lee. Fre-gan 2: Fast and efficient frequency-consistent audio synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6192–6196. IEEE, 2022.

[24] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017.

[25] Jen-Yu Liu, Yu-Hua Chen, Yin-Cheng Yeh, and Yi-Hsuan Yang. Unconditional audio generation with generative adversarial networks and cycle regularization. *arXiv preprint arXiv:2005.08526*, 2020.

[26] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1429–1437, 2019.

[27] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[28] Andrés Marafioti, Nicki Holighaus, Nathanaël Perraudin, and Piotr Majdak. Adversarial generation of time-frequency features. *arXiv:1902.04072v2*, 2019.

[29] Andres Marafioti, Piotr Majdak, Nicki Holighaus, and Nathanaël Perraudin. Gacela: A generative adversarial context encoder for long audio inpainting of music. *IEEE Journal of Selected Topics in Signal Processing*, 15(1):120–131, 2020.

[30] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[31] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[32] Max Morrison, Rithesh Kumar, Kundan Kumar, Prem Seetharaman, Aaron Courville, and Yoshua Bengio. Chunked autoregressive gan for conditional waveform synthesis. *arXiv preprint arXiv:2110.10139*, 2021.

[33] Anastasia Natsiou and Seán O'Leary. Audio representations for deep learning in sound synthesis: A review. In *2021 IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2021.

[34] Javier Nistal, Cyran Aouameur, Stefan Lattner, and Gael Richard. Vqcpc-gan: variable-length adversarial audio synthesis using vector-quantized contrastive predictive coding. In *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 116–120. IEEE, 2021.

[35] Javier Nistal, Stefan Lattner, and Gael Richard. Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks. *arXiv preprint arXiv:2008.12073*, 2020.

[36] Javier Nistal, Stefan Lattner, and Gael Richard. Comparing representations for audio synthesis using generative adversarial networks. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 161–165. IEEE, 2021.

[37] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[38] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[39] Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. *arXiv preprint arXiv:2104.08894*, 2021.

[40] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[41] António Ramires, Frederic Font, Dmitry Bogdanov, Jordan BL Smith, Yi-Hsuan Yang, Joann Ching, Bo-Yu Chen, Yueh-Kao Wu, Hsu Wei-Han, and Xavier Serra. The freesound loop dataset and annotation tool. *arXiv preprint arXiv:2008.11507*, 2020.

[42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[43] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[44] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

[45] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4570–4580, 2019.

[46] Charles Spence and Nicola Di Stefano. Coloured hearing, colour music, colour organs, and the search for perceptually meaningful correspondences between colour and sound. *i-Perception*, 13(3):20416695221092802, 2022.

[47] Gijsbert Stoet. Psytoolkit: A software package for programming psychological experiments using linux. *Behavior research methods*, 42:1096–1104, 2010.

[48] Gijsbert Stoet. Psytoolkit: A novel web-based method for running online questionnaires and reaction-time experiments. *Teaching of Psychology*, 44(1):24–31, 2017.

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[50] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[51] John Woodruff, Yipeng Li, and DeLiang Wang. Resolving overlapping harmonics for monaural musical sound separation using pitch and common amplitude modulation. In *Proc. ISMIR*, pages 538–543. Citeseer, 2008.

[52] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020.

[53] Yen-Tung Yeh, Bo-Yu Chen, and Yi-Hsuan Yang. Exploiting pre-trained feature networks for generative adversarial networks in audio-domain loop generation. *arXiv preprint arXiv:2209.01751*, 2022.

[54] Reo Yoneyama, Ryuichi Yamamoto, and Kentaro Tachibana. Nonparallel high-quality audio super resolution with domain adaptation and resampling cycle-gans. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[55] Massimiliano Zampini and Charles Spence. The role of auditory cues in modulating the perceived crispness and staleness of potato chips. *Journal of sensory studies*, 19(5):347–363, 2004.

[56] Ye Zhu, Kyle Olszewski, Yu Wu, Panos Achlioptas, Menglei Chai, Yan Yan, and Sergey Tulyakov. Quantized gan for complex music generation from dance videos. In *European Conference on Computer Vision*, pages 182–199. Springer, 2022.

Ringraziamenti

Un sentito ringraziamento va a tutte le persone che hanno contribuito alla realizzazione di questa tesi magistrale offrendomi il loro sostegno.
Vorrei ringraziare la mia famiglia, per avermi sempre supportato e per avermi permesso di intraprendere questo percorso di studi.
Dedico un ringraziamento particolare al mio relatore, Antonio Rodà, per essere sempre stato disponibile e per avermi guidato nello svolgimento di questa tesi. Grazie ai Suoi consigli, è stata un'esperienza stimolante e ricca di soddisfazioni.
Un ringraziamento speciale va al Centro di Sonologia Computazionale, dove ho svolto il periodo di Research Training. In particolare, ringrazio il mio correlatore Sergio Targon Canazza, e i ricercatori Alessandro Fiordelmondo e Filippo Carnovalini, per avermi assistito nel corso di questa ricerca.
Infine, ringrazio i miei colleghi che mi hanno accompagnato in questi anni di studi all'Università degli Studi di Padova, che hanno reso gradevoli anche i momenti più difficili e impegnativi.

Padova, 27 Novembre 2023
Edoardo Bastianello