Generative Adversarial Networks for Synthesis and Control of Drum Sounds

Antoine Lavault

Sorbonne Université

Ecole Doctorale Informatique, Télécommunications et Électronique

Laboratoire Sciences & Technologies de la Musique et du Son (UMR 9912)

Équipe Analyse/Synthèse

Generative Adversarial Networks for Synthesis and Control of Drum Sounds

Par Antoine Lavault

Thèse de doctorat de Sciences et Technologies de l'Information et de la Communication

Dirigée par Axel Roebel

Présentée et soutenue publiquement le 8 Décembre 2023

Devant un jury composé de :

Rapporteur, Pr. Philippe Depalle, McGill University (Canada)

Rapporteur, Pr. Vesa Välimäki, Aalto University (Finlande)

Examinateur, Pr. Slim Essid, LTCI - Télécom Paris - Institut Polytechnique de Paris

Examinateur, Dr. Sølvi Ystad, Laboratoire Prism, Université Aix-Marseille[1]

Examinateur, Dr. Stefan Lattner, Sony Computer Science Laboratories, Paris

Directeur de thèse, Dr. Axel Roebel, Ircam / UMR9912 / Sorbonne Université Paris

[1]Présidente du Jury

Contents

List of Figures

List of Tables

Summary

Audio synthesizers are electronic systems capable of generating artificial sounds under parameters depending on their architecture. Even though multiple evolutions have transformed synthesizers from simple sonic curiosities in the 1960s and earlier to the main instruments in modern musical productions, two major challenges remain; the development of a system of sound synthesis with a parameter set coherent with its perception by a human and the design of a universal synthesis method, able to model any source and provide new original sounds.

This thesis studies using and enhancing Generative Adversarial Networks (GAN) to build a system answering the previously-mentioned problems. The main objective is to propose a neural synthesizer capable of generating realistic drum sounds controllable by predefined timbre parameters and hit velocity.

The first step in the project was to propose an approach based on the latest technological advances at the time of its conception to generate realistic drum sounds. We added timbre control capabilities to this method by exploring a different way from existing solutions, i.e., differentiable descriptors. To give experimental guarantees to our work, we performed evaluation experiments via objective metrics based on statistics and subjective and psychophysical evaluations on perceived quality and perception of control errors. In subsequent experiments we added control to the timbral control.

Furthermore, with the idea of pursuing the realization of a versatile synthesizer with universal control, we have created a dataset ex-nihilo composed of drum sounds to create an exhaustive database of sounds accessible in the vast majority of conditions encountered in the context of music production. From this dataset, we present experimental results related to the control of dynamics, one of the critical aspects of musical performance but left aside by the literature.

This work will present how we can leverage deep neural networks for drum synthesis and how we can add timbral and dynamics control to the neural synthesizer.

Résumé

Les synthétiseurs audio sont des systèmes électroniques capable de générer des sons artificiels sous un ensemble de paramètres dépendants de leur architecture. Quand bien même de multiples évolutions ont transformé les synthétiseurs de simples curiosités sonores dans les années 60 et précédentes à des instruments maîtres dans les productions musicales modernes, deux grands défis restent à relever: le développement d'un système de synthèse répondant à des paramètres cohérent avec leur perception par un humain et la conception d'une méthode de synthèse universelle, capable de modéliser n'importe quelle source et de la dépasser.

Cette thèse étudie l'utilisation et la valorisation des réseaux antagonistes génératifs (Generative Adversarial Networks, abrégé en GAN) pour construire un système répondant aux deux problèmes exposés précédemment. L'objectif principal est ainsi de proposer un synthétiseur neuronal capable de générer des sons de batteries réalistes et contrôlable par un ensemble de paramètres de timbres prédéfinis, ainsi que de proposer un contrôle de la vélocité de la synthèse.

La première étape dans le projet a été de proposer une approche basée sur les dernières avancées techniques au moment de sa conception pour générer des sons de batteries réalistes. A cette méthode de synthèse neuronale, nous avons aussi ajouter des capacités de contrôle du timbre en explorant une voie différente des solutions existantes: l'utilisation de déscripteurs différentiables. Pour donner des garanties expérimentales à notre travail, nous avons réalisé des expériences d'évaluation à la fois via des métriques objectives basées sur les statistiques mais aussi des évaluations subjectives et psychoĥysiques sur la qualité perçue et la perception des erreurs de contrôle. Pour proposer un synthétiseur utilisable pour des performances musicales, nous avons aussi ajouter un contrôle de la vélocité.

Toujours dans l'idée de poursuivre la réalisation d'un synthétiseur universel et à contrôle universel, nous avons créer ex-nihilo un jeu de données composé de sons de batteries dans le but avoué de créer une base exhaustive des sons accessibles dans l'immense majorité des conditions rencontrées dans le contexte de la production musicale. De ce jeu de données, nous présentons des résultats expérimentaux liés au contrôle de la dynamique, un des aspects phares de la performance musicale mais laissé de côté par la littérature.

Chapter 1

Introduction

Synthesizers in the musical domain are instruments capable of generating artificial sounds under the constraint of a set of control parameters. Growing in popularity since the 1970s [Chowning, 1977; Chamberlin, 1985] and now part of most modern music productions, the audio synthesizer redefined how music is composed and performed, especially with the rise of Digital Audio Workstations (DAW) and the real-time capabilities of modern computers.

While the progress made by synthesizers is more than notable, there is still a fundamental obstacle: the development of user-driven interfaces, allowing for a genuine expression while not constraining the musician to a limited set of sounds. Terms like waveforms, filters, ADSR[1] envelopes, frequency modulation (FM), LFO[2], or VCO[3] are common knowledge only for a few. Therefore, modern synthesizers' vast and complex parameter space is an unquestioned source of hurdles that slows down the creative process or limits the usage of the synthesis possibilities to a few presets. In addition, the learning process behind mastering the capabilities of a synthesizer may seem similar to any other musical instrument, where one has to train hard to master the instrument's sonic capabilities fully. For instance, a drummer will practice the 40 rudiments, for both hand and feet, before being able to master the whole range of sonic possibilities offered by a drum kit. Similarly, a pianist will perform drills to increase his virtuosity and control over his dynamics. However, the process to attain this is quite accessible: the direct (and mechanical) interaction with the instrument produces the sound set available. On the contrary, audio synthesizers require prior signal processing knowledge to let the user control the system purposefully toward a specific sound. The main barrier to entry resides in the large gap between signal processing notions and the musician's creative process. In a way, the synthesizer and its user do not speak the same language; one is implanted in a long scientific history of hard truths and abstract concepts, and the other is in another abstract space but made of emotions, perceptions, aesthetics, and overall experience. Using a synthesizer is equivalent to mapping an abstract and unstructured creative process onto a rigid, synthesizer-specific and fixed parameter set: the user bends itself metaphorically to the synthesizer.

Moreover, most synthesizers focus on harmonic sounds rather than percussion. While it

[1] Attack-Decay-Sustain-Release
[2] Low-Frequency Oscillator
[3] Voltage-Controlled Oscillator

is possible to synthesize "good" drum sounds using traditional additive, subtractive, or FM synthesis at the cost of spending a long time with non-descriptive controls or creating far-from-realistic sounds, most of the renowned and commonly used drum synthesizers are sample-based or use physical models. This creates a trade-off between sonic capabilities and ease of use. To that end, drum synthesizers like the Roland TR-808 or TR-909 are limited in sonic variety. Still, they are easily programmed when a Korg Wavedrum is unlimited in sound and performance but virtually unusable by a musician.

From the context described above, we can identify a few challenges and limitations of modern synthesizers:

1. the need for dedicated and highly specific software or hardware for musical purposes;
2. the necessary mastery of different synthesis techniques and workflows;
3. the need for sample libraries due to the unavailability of technology for modeling specific sound sources or to accelerate the creative process;
4. the creative barrier modern synthesizers impose through their obscure terminology and workflows for the average musician.

The question arises about designing a synthesis technique exposing intuitive and semantically relevant parameters. Such a technique would make terms like ambiance, space, pitch, and timbre the principal properties of the sound synthesis instead of reverb time, diffusion, frequency, filter cutoff, and resonance.

1.1 Objectives

The objectives of this thesis are plural. First and foremost, we aim to engineer a drum synthesizer capable of perceptively good quality while providing real-time capabilities. While low-fidelity is a genre in and of itself, it is most common to expect good sound quality from a modern synthesizer. Second, we expect basic control over the generation by choosing the type of drum or cymbal to synthesize, while allowing interpolation possibilities for advanced users. Finally, we expect this drum synthesizer to provide high-level and semantically relevant controls over the synthesis, first, with timbral properties that would be relevant for the user, e.g., brightness, warmth, or depth. And then offer performance-bound parameters, with the most important for a percussionist: dynamics.

In summary, we expect the following:

- Good synthesis quality (in terms of perceived quality)
- Sufficient computational performance to allow for real-time use.
- Basic control over the type of drum
- Advanced control over timbral properties
- Advanced control over performance dynamics

This study will be limited to real drum and cymbal sounds, as synthetic drum sounds can already be synthesized with state-of-the-art methods, hence their name.

1.2 Background

While the main focus of this work is deep-learning-based generative models for drum synthesis, several associated topics have to be covered to understand the project's context better. These topics are discussed in the following sections of the present chapter.

First, we will briefly describe the physics behind drum and cymbal sounds to understand standard modeling methods and some characteristics of these percussion instruments. By understanding the phenomena at work here, we can gain insights into constraints or prior knowledge to add to the deep generative model. We will then consider traditional methods to synthesize drum sounds and, more generally, how to use drum sounds in a musical production context. We will then review the traditional methods to synthesize drum sounds in a music production context, such as additive synthesis, FM synthesis, or sample-based synthesis. We will also review their limitations in the context of music production. By restraining ourselves to this thesis's target application, we can learn the limits of what is offered to music producers. From this base, we will dive into a succinct description of deep learning, its properties, and its most common architectures. Finally, we will describe how humans perceive sound and how we can use human listeners' perceptions in the context of subjective quality evaluation and psychophysics.

Audio examples for this chapter can be accessed at https://alavault.github.io/stylewavegan_phd/#introduction.

1.3 Physics of drum sounds

Under the umbrella of *drum sounds*, we limit ourselves to studying the components of a modern drum kit. We must distinguish between the actual "drums," shell drums, and the cymbals. This study will not deal with ethnic percussive instruments like the taiko drum or classical instruments like the timpani or the marimba.

1.3.1 Basic Physics of Drum and Cymbals

The physics of drums and cymbals are quite complex but can be summarized into a few main concepts.

For the two-headed drum, i.e., the most common configuration with modern shells, we can model the interaction between the two heads with a two-mass vibrator. This simple model shows the presence of two inharmonic modes, called membrane modes, which in turn shows the coupling between the heads of the drum [Rossing et al., 1992; Rossing, 2000]. A pitch glide can be observed on real drums due to the increased tension locally during the hit. This increases the mode frequency, returning to its expected value after the strike [Fletcher and Bassett, 1978; Rossing, 2000].

Figure 1.1: Modern Drum Kit - Mapex Black Panther Velvetone

For cymbals, physics is mainly described by the study of plates. Similarly to drum shells, the cymbals show vibrational modes with inharmonic relationships. What is more notable with cymbals is their non-linear behavior at high excitation amplitude [Fletcher and Rossing, 1998].

The following sections will focus on the different specifications from audio recordings of these instruments while considering the underlying physical phenomena.

1.3.2 Shells

What we call "shells" can be more formally described as drums that do not convey a definite pitch or, at the very least, a much weaker sense of pitch than classical percussion instruments like the timpani or ethnic percussion instruments like the Indian tabla or mridanga [Rossing, 2000].

As mentioned before, we will focus on the modern drum kit used in popular music production, i.e., a set composed of one or several kick drums (also called bass drums), a snare drum, and a set of toms (also known as tom-toms).

The drums described in the sections below have (in general) two heads, one called the batter head, which the percussionist strikes, and the resonant head, used to tailor the characteristics of the overall sound.

Toms

What was originally an African drum, the tom-tom, is now part of a modern drum kit, primarily found in a set of side drums of different sizes and shapes with two heads in general[4]. In the context of modern music and especially rock, it is more colloquially called a *tom*. [Rossing, 2000]

These drums have varying sizes from 8 to 18 inches. In the modern drum set shown in Fig. 1.1, the number of toms is 4, divided into two rack toms (10 and 12 inches wide) and two floor toms (14 and 16 inches wide). Of course, smaller and bigger tom sets exist and will be chosen by the percussionist depending on the musical context.

The toms are generally the most musical drums in a drum kit, as they can have a semblance of the pitch even if the sinusoids (resonances) constituting the sound exhibit an inharmonic relationship between themselves, as shown in Fig. 1.2a and 1.2b.

What is shown in these figures is the strong amplitude of the sinusoid bound to the fundamental frequency, which gives the tom a semblance of an instrument with pitch. Please note the extended decay time of the sinusoid associated with the fundamental frequency.

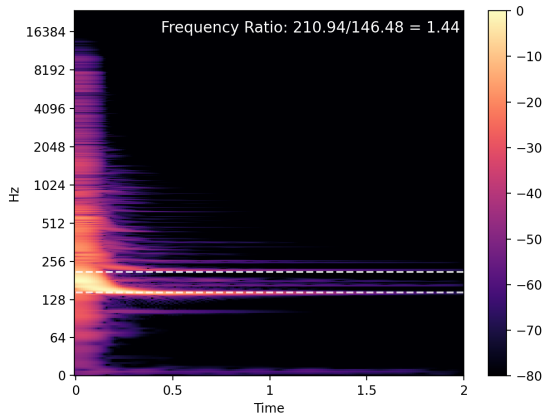
Figure 1.2 clearly shows the frequency decreasing during a short period after the initial hit. Indeed, when a drum head is hit, its tension increases, which causes the pitch of the sound to change. This happens because, like a string, the frequency of vibration of the drum head increases as the tension increases due to the initial hit. As the amplitude decays, the frequency drops slightly non-linearly as a function of time, as shown in Fig. 1.2c [Rossing, 2000]. In the case of Fig. 1.2c, the fundamental frequency is almost a D3 in pitch, with a pitch glide of a minor third (F3 to D3).

Snare Drum

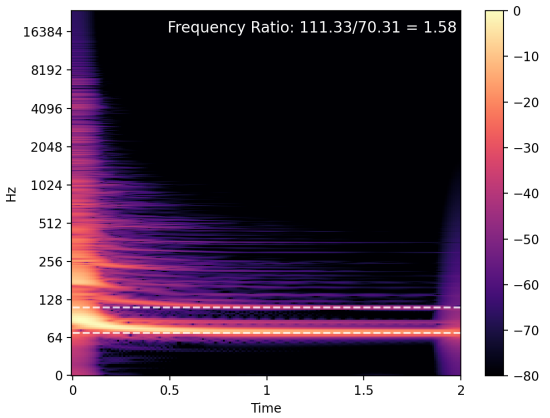
While the snare drum has evolved over several centuries, its modern evolution is a two-headed drum with a width of 14 inches and a depth between 5 and 8 inches [Rossing, 2000; Toulson, 2021]. Still, sizes may vary above and below these values. For instance, the piccolo snare drum, with a width of 10 inches and a depth between 3 and 4.5 inches.

The snare drum has got its name due to the snare wires. Snare wires are an essential component of a snare drum. They are thin, tightly wound metal wires stretched across the snare drum's bottom head, on the outside of the drum as shown in Fig. 1.3a. Generally, the snare wires are attached to a snare bed, a small indentation or groove in the drum's bearing edge. The wires are held at each end of the drum shell with snare straps, which are thin pieces of fabric, string, or plastic attached to the snare wires and the snare throw-off mechanism, as seen in Fig. 1.3b. This mechanism allows the snare wires to be engaged or disengaged from the drum's bottom head, which changes the sound of the drum and can also adjust the tension of the wires against the bottom head of the snare drum. When the snare wires are engaged, they vibrate against the bottom head of the snare drum, creating a characteristic snare sound that is a staple of many genres of music, particularly

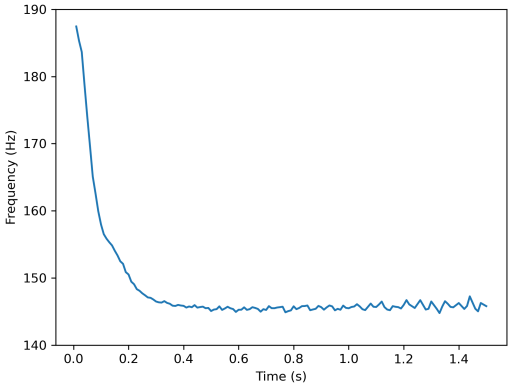
[4]Tom-toms with one head exist and are nowadays used in orchestral percussion ensembles. These one-headed toms could also be heard on rock records of the 1970s



(a) Spectrogram of a recorded sound of a tom (Yamaha Stage Custom 10", rack)



(b) Spectrogram of a recorded sound of a tom (Yamaha Stage Custom 16", floor)



(c) Pitch Glide of a 10-inch tom obtained with CREPE [Kim et al., 2018] (Yamaha Stage Custom)

Figure 1.2: Spectrograms and pitch estimation of sound recordings of toms. The horizontal white dashed lines in the figure represent the first two partials utilized in calculating the frequency ratio.

in rock, jazz, and marching band music.

(a) Bottom head of a snare drum and snare wires

(b) Snare drum throw-off in its engaged position with snare straps visible

Figure 1.3: Details of the bottom of a snare drum and snare throw-off

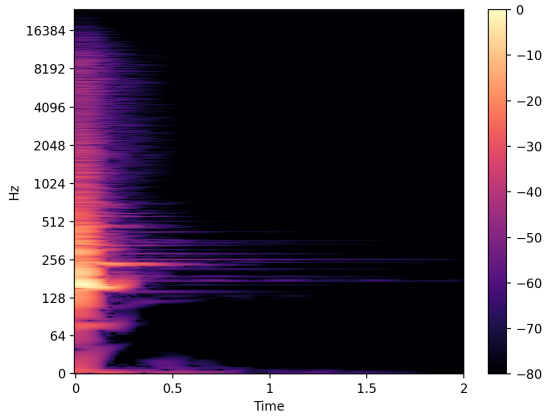
The snare wires' vibration is due to the coupling between the two heads through the enclosed air and the drum shell. Figures 1.4a and 1.4b show spectrogram of a snare drum recorded by microphones on top and bottom of the drum, i.e., one on the batter side and the other on the snare side. Regarding frequencies, the top side recording shown in Fig. 1.4a presents a high harmonic content with a few resonances. On the other side, shown in Fig. 1.4b, the resonances are more numerous, but more importantly, the beginning of the sound is akin to noise. There are no long decaying fundamental and resonances compared to a tom, like in Fig. 1.2b.

In the time domain, the activity of the snare wires against the snare head is shown in Fig. 1.4c. This figure shows the recording of two microphones on top and at the bottom of the snare drum. The top figure shows the snare drum's top side (or batter side), while the bottom figure shows the snare side. We can see that the action of the wires against the resonant head (i.e., snare side head) creates a noisy signal. This is the signature of a snare drum sound.

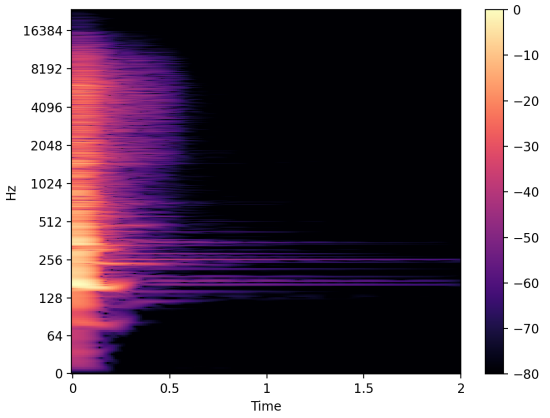
The snare drum behaves more like a tom when the snare wires are off the resonant head, i.e., physically decoupled from the snare side head. This is shown in Fig. 1.4d, where the snare side recording shows no high-frequency noise. Moreover, note the polarity inversion between the two signals due to the microphones being opposite of each other. The coupling between the two heads can explain this: when the batter side is hit, the batter head is pushed down, and by coupling, so is the snare side head. But one will push inwards (the batter side) and the other outwards (the snare side), hence differences in terms of the sound pressure gradient (one positive, the other negative) and finally recorded by the microphones as such.

Kick Drum

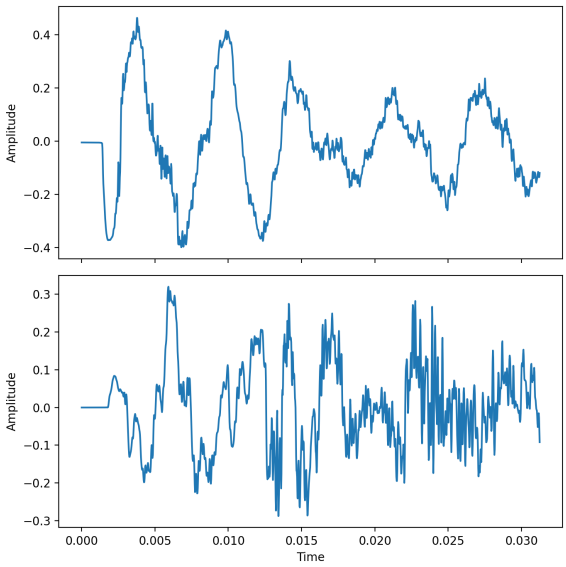
The kick drum (or bass drum) is generally the widest and deepest drum in the drum kit. With a diameter ranging from 18 inches to 26 inches, the kick drum is generally excited



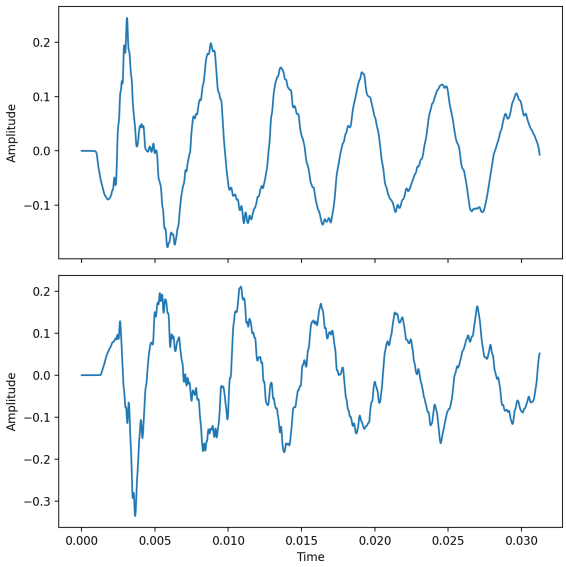
(a) Spectrogram of a recorded sound of a snare drum with snare wires on, recorded from the top side (Yamaha Stage Custom, 14"x5.5")



(b) Spectrogram of a recorded sound of a snare drum with snare wires on, recorded from the snare side (Yamaha Stage Custom, 14"x5.5")



(c) Sound recording of top and bottom snare microphones showing the snare action (Yamaha Stage Custom, 14"x5.5")



(d) Sound recording of top and bottom snare microphones showing the snare action with snares off (Yamaha Stage Custom, 14"x5.5")

Figure 1.4: Spectrogram and Waveforms for snare drum

through a beater linked to a pedal. In modern music production, most kick drums are *ported*, i.e., have a hole cut into the resonant head to reduce the sustain and introduce a microphone inside.

Due to the bigger size, it would be expected to have more sustain out of the kick drum. However, this kind of sound is generally disregarded in modern production. Kick drums are generally dampened using pillows or blankets to give a faster decay; hence a "punchier" sound [Toulson, 2021]. This additional dampening helps make the kick drum sound different from a tom. The effect of dampening on the sounds can be seen by comparing either figure Fig. 1.2a and 1.2b to Fig. 1.5. It's worth noting that the fundamental pitch is significantly lower than the other drums, as expected due to its size.

In Fig. 1.5, we also observe two low-intensity excitations corresponding to small rebounds of the beater on the drum pedal.

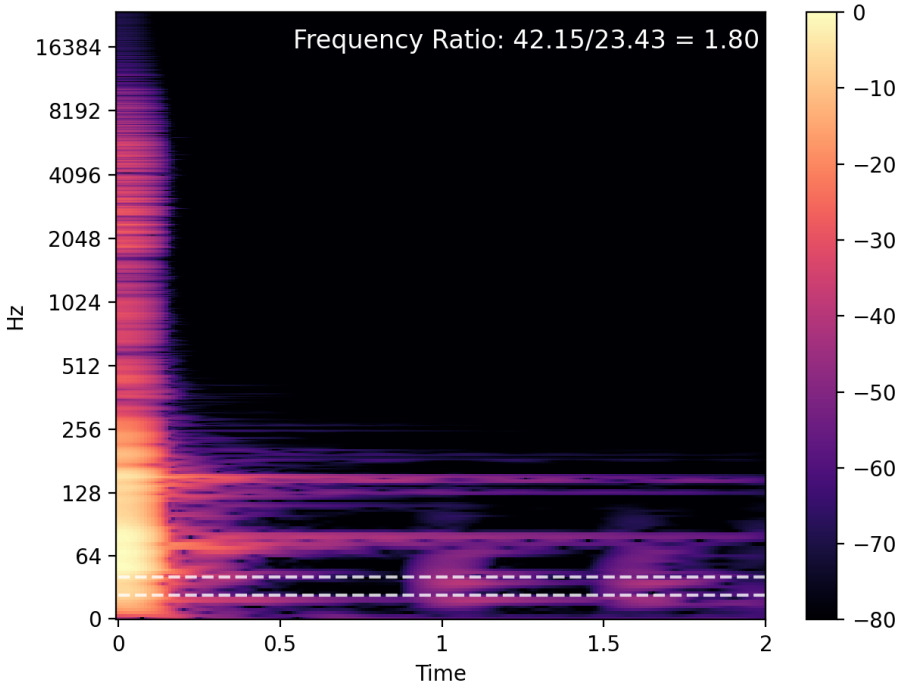


Figure 1.5: Spectrogram of a recorded sound of a kick drum (Mapex Velvetone, 22"x18")

1.3.3 Cymbals

In the context of the modern drum kit, what we can consider cymbals are, in fact, descendants of the Turkish tradition of cymbal-making. These Turkish cymbals are saucer-shaped with a small dome in the center, called colloquially bell. Another kind of cymbal found in modern drum kits comes from the Chinese tradition, where the cymbals have a turned-up edge. These are colloquially known as "china" cymbals in the context of the modern drum kit. Still, its usage is generally limited to effects or accentuation. With different constructions come different sounds. The terms used to describe the cymbal's sounds have been set up by jazz musicians with onomatopoeia, such as "Crash," "Ride," or "Splash." [Rossing, 2000]

Description of the most common types of cymbals

In a modern drum kit, like the one shown in Fig. 1.6, the cymbals are presented on stands, which is quite different from the handheld cymbals found in classical percussion.

As stated before, the name of the cymbals in the modern drum kit descends from jazz drummers' descriptions using onomatopoeia. First, the crash cymbal whose behavior we will study in Sect. 1.3.3 is a cymbal of size ranging from 13 to 21 inches (typically between 14 and 18 inches) and is mostly used for accentuation, both in classical and modern percussion works. In more extreme music genres, the crash cymbal can be used as a timekeeping element to compete with the rest of the ensemble [Pieslak, 2007]. The crash cymbal is described by a fast attack and a quick to moderate decay. The crash cymbal is generally played on its edge by whipping it with the drumstick shaft to obtain the characteristic "crash" sound. To illustrate this point, we can use Fig. 1.7a. Here we

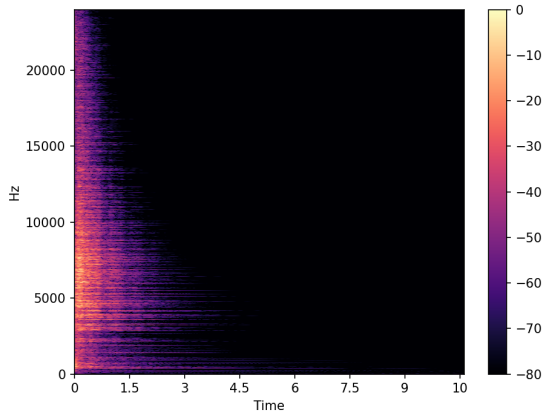
Figure 1.6: Modern drum kit with cymbals (Yamaha Stage Custom). From left to right, the cymbals' types are: hi-hat, crash, ride, crash, and ozone-like (similar in sound to a china cymbal)

can see the cymbal does not sustain for too long (around 4 seconds), and its attack phase can be considered a filtered noise burst. This corresponds to the fast attack and quick decay described above. Small crashes (ranging from 6 to 10 inches) are called "splash" and produce a very short sound, which is reasonably usable only for accentuation.

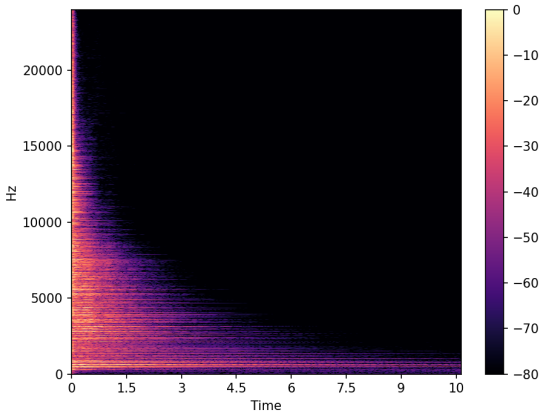
Another part of the modern drum kit cymbal kit is the ride cymbal. Originating from the jazz drum kit, the ride cymbal was its main timekeeping element [Brown, 1990], when a crash would be relegated to accentuation. Generally, a ride cymbal is a cymbal of diameter between 18 and 26 inches. However, the most commonly found models are in the 20 to 22-inch range. Ride cymbals are generally larger and heavier than crash cymbals. Being heavier makes its attack slower and not as suitable for crash technique. Because of this, the ride cymbal is generally played with the tip of the drumstick to obtain a sharp "ping" followed by sustained overtones. This can be seen in Fig. 1.7b where the attack phase is short, but the cymbal sustains for quite some time (around 10 seconds when, in comparison, the crash cymbal in Fig. 1.7a sustains for about 4 seconds). When the ride is played like a crash cymbal, as shown in Fig. 1.7c, the frequency content of the attack changes slightly, but the sustained part remains almost the same.

The hi-hat is the other timekeeping element in the modern drum kit, with the ride cymbal. The hi-hat will be extensively described in Sect. 1.3.3, but can be shortly described as two cymbals on a rod, one fixed and the other moving through a clutch mechanism.

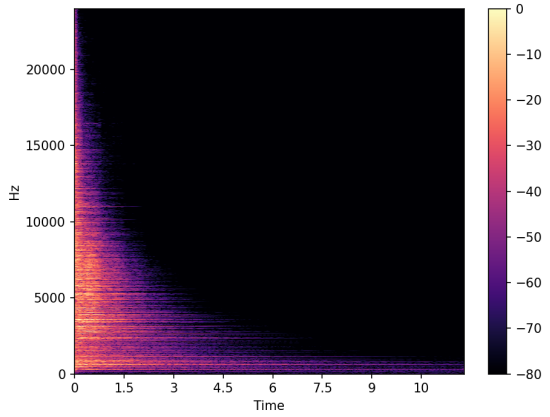
Cymbals of Chinese descent are kept for accentuation purposes or special effects. As shown in Fig. 1.7d, the China cymbal can be considered a very sustained crash. Being a cymbal with a turned-up edge, its sound will differ from a ride or a crash cymbal and will be more akin to a gong.



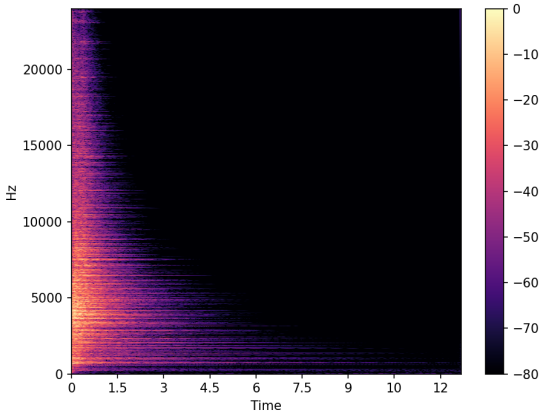
(a) Spectrogram of a recorded sound of a crash cymbal, played like a crash (Sabian 14" AAX Studio Crash)



(b) Spectrogram of a recorded sound of a ride cymbal, played like a ride (Zildjian K Jazz Ride 20")



(c) Spectrogram of a recorded sound of a ride cymbal, played like a crash (Zildjian K Jazz Ride 20")



(d) Spectrogram of a recorded sound of a china cymbal (Sabian Pro Series China 20")

Figure 1.7: Spectrogram of a recorded sound of different cymbal types

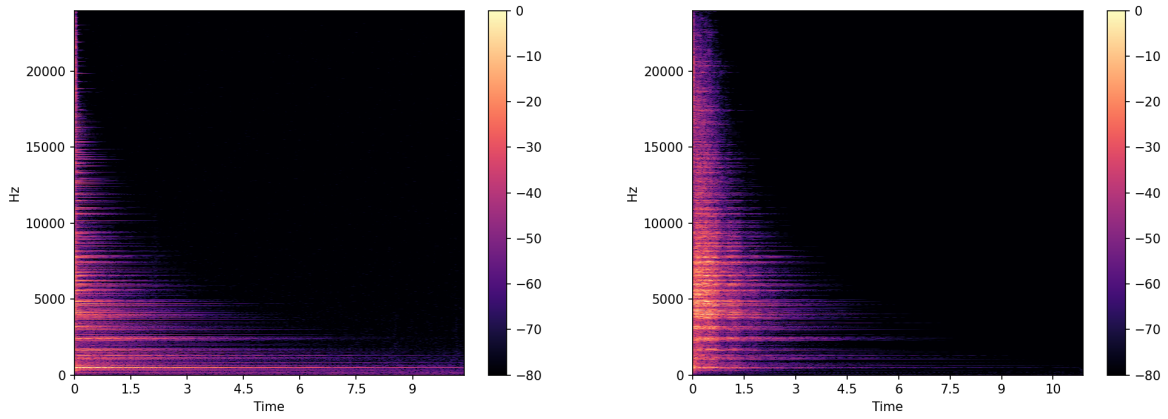
Non-linear behavior of cymbals

Cymbals exhibit a severe non-linear behavior when struck with enough force, as mentioned in [Rossing, 2000]. For this section, we will focus on the behavior of a crash cymbal, which can be extended to other Turkish cymbals.

The non-linear behavior can be described as pseudo-chaotic when subjected to a fixed-frequency continuous non-contact excitation using magnetic drivers [Legge and Fletcher, 1989; Wilbur and Rossing, 1997; Chaigne et al., 2005]. These studies on cymbals and gongs using this excitation method have shown that the progression towards highly non-linear behavior occurs in a few steps, with nonlinearity effects that resemble chaotic behavior observed as the excitation amplitude increases.

However, these studies use an excitation method that differs greatly from a drumstick. The excitation of a cymbal with a drumstick can be seen as an excitation with an impulse, meaning that all frequencies are excited from the beginning. Since the project aims to work on drums played with classical tools, including drumsticks, we shouldn't expect to

observe the results from [Legge and Fletcher, 1989; Wilbur and Rossing, 1997; Chaigne et al., 2005] directly on any of the target sounds. As a result, attempting to show the behavior described in [Legge and Fletcher, 1989; Wilbur and Rossing, 1997; Chaigne et al., 2005] in the spectra of recorded cymbals may not be appropriate, as they always contain a high amount of noise regardless of the hit strength, as shown in Fig. 1.8.



(a) Spectrogram of a recorded sound of a Crash cymbal (Sabian 16" AAX AAXplosion Crash, Low Velocity))

(b) Spectrogram of a recorded sound of a Crash cymbal (Sabian 16" AAX AAXplosion Crash, High velocity)

Figure 1.8: Spectrograms of a recorded crash cymbal sound, played with a drumstick

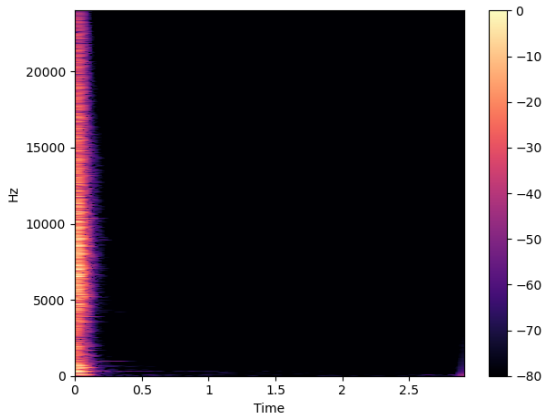
We can make several observations after examining the spectrogram of the crash cymbal excited by a drumstick, as shown in Fig. 1.8. The frequency content of the low-velocity hit in Fig. 1.8a is broadband but resolved, indicating that the drumstick hit is indeed close to an impulse, hence excites all frequencies, but non-linear behaviors are not heavily pronounced as the individual resonances are still quite clear. Conversely, the high-velocity hit in Fig. 1.8b appears to have lost its resonances and become more similar to filtered noise. It is also worth noting that the two sounds are perceived quite differently: the low-velocity sample lacks the noisy and explosive qualities of the high-velocity hit. All in all, when excited with a drum stick, cymbals show many resonances, and the non-linear behavior can be interpreted as filtered noise for higher velocities.

Hi-hat

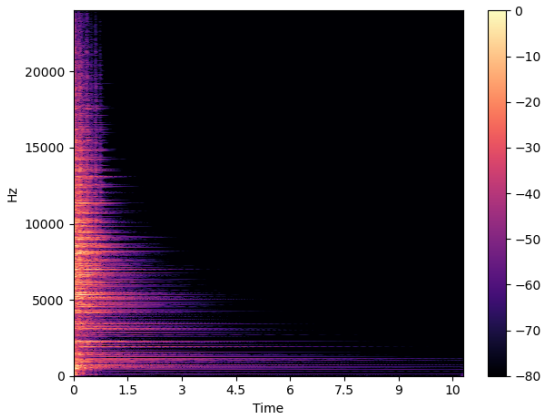
Hi-hats consist of a pair of cymbals, whose diameter is generally between 13 and 16 inches, mounted on a stand with the two cymbals facing each other. The bottom cymbal is fixed, and the top is mounted with a clutch piece on a rod that moves the top cymbal toward the bottom one when an action pedal is pressed. When the pedal is fully pressed, the hi-hat is said to be *closed*. When the pedal is not pressed, the hi-hats are qualified as *opened*, i.e., the top and bottom cymbals are not physically in contact. Please note that intermediary positions are possible through careful use of the action pedal.

Because of the proximity of the cymbals, they can interact by touching each other, creating a more aggressive sound, especially in a half-opened position. In terms of musical application, the hi-hat offers excellent articulation and dynamics, enabling it to produce rhythmic pulses or accents by merely using the pedal. This versatility allows drummers to

create a wide range of rhythmic patterns and textures, making it an essential component in shaping a musical piece's overall sound and feel.



(a) Spectrogram of a recorded sound of a closed Hi-hat (Zildjian K-Custom Dark Hat 13")



(b) Spectrogram of a recorded sound of an open Hi-hat (Zildjian K-Custom Dark Hat 13")

Figure 1.9: Spectrogram of a recorded sound of Hi-hat with different opening levels

To showcase the distinctions between the levels of openness in the hi-hat, we have included Fig. 1.9a and 1.9b. When the hi-hat is in a closed position (as shown in Fig. 1.9a), it produces a brief sound that decays quickly but with a lot of harmonic content similar to noise. This modeling of the hi-hat as a short noise burst can be found usage in the litterature for sensitory experiments [Liski et al., 2021]. On the other hand, when the hi-hat is open (as shown in Fig. 1.9b), it generates a longer sound consisting of many partials, similar to what is observed with the crash cymbal. This is because when the hi-hat is open, the top and bottom cymbals collide with each other several times after the initial strike, which results in a unique and sustained sound (0.2s for the closed hi-hat, 3s for the open hi-hat) because the cymbals can move more freely. As the cymbals gradually lose energy after the initial hit, the frequency content of the sound decreases, and direct contact between the cymbals ceases. However, they continue to resonate, unlike in the closed position, where the resonance is abruptly stopped, as the cymbals cannot resonate freely in this position. Please note that strikes of lower intensity in the open position might not be strong enough to excite the lower hi-hat. In this case, we obtain a sound similar to a low-intensity stricken crash.

1.4 Drum Synthesis

1.4.1 Signal-based drum synthesis method

Additive Synthesis

Additive synthesis is a sound synthesis technique that creates timbre by superposing sine waves and noise [Serra and Smith, 1990; Bonada et al., 2011; Rodet, 1997]. In this standard model, a signal S can be decomposed into a sinusoidal part s and a stochastic part r, which can be expressed as follows:

$$S(t) = s(t) + r(t) = \underbrace{\sum_{i=1}^{N} c_i(t)}_{\text{Sinusoidal Part}} + \underbrace{r(t)}_{\text{Noise/Stochastic Part}} \tag{1.1}$$

The sinusoidal part $s(t)$ is the sum of N sine waves $c_i(t)$. In this context, the c_i are called *partials* and have time-varying parameters (amplitude a and phase ϕ). Under the assumption that $a_i(t)$ varies slowly, a partial can be defined as below, with $a_i(t) \geq 0$ the time-varying amplitude and $\phi_i(t)$ the phase of the sinusoidal partial, which has an instantaneous frequency $f_i(t)$:

$$\begin{cases} c_i(t) = a_i(t)cos(\phi_i(t)) \\ \dfrac{d\phi_i}{dt} = f_i(t) \end{cases} \tag{1.2}$$

In other words, each partial c_i is a sine wave with frequency and amplitude that modulates according to an envelope or a low-frequency oscillator and additional filters [Smith and Serra, 1987]. As shown in eq. (1.2), the sine wave instantaneous frequency may also be modulated so that the frequency varies over time. To ensure that the sinusoidal part s will not start to represent noise, additional constraints on c_i must be added, notably the constraints that the amplitude and the instantaneous frequency, $a(t)$ and $f(t)$, may only vary slowly over time.

A noise signal r is added to simulate the random component the pure sinusoidal model can't describe easily due to the abovementioned constraints on the oscillators. This additional noise is the residual between the pure sinusoidal model and the target sound. It is considered a random signal [Rodet, 1997; Bonada et al., 2011], which is generally modeled using a filtered white noise.

Additive synthesis is a powerful technique for generating various sounds, from musical instruments to human voices. However, for high-quality resynthesis of natural sounds, the number of oscillators can be quite high, posing the problem that meaningful control of the parameters is difficult.

FM synthesis

FM synthesis, as initially proposed by Chowning [Chowning, 1977], has been used to generate a wide range of sounds, including bell-like sounds. While it is possible to create drum sounds using FM synthesis, this approach is not typically considered a standard method for synthesizing drum sounds. Clarke [Clark, 2003] has explored this approach in his work, but it should be considered more of a curiosity than a reliable method for producing realistic drum sounds. While additive synthesis can generate inharmonic sounds, FM synthesis stands out because it can produce complex inharmonic partials through a more efficient and simpler algorithm regarding the number of parameters and oscillators required.

The simplest expression of an FM-modulated oscillator is:

$$e(t) = A\sin(2\pi\alpha t + I\sin(2\pi\beta t)) \tag{1.3}$$

In this case, FM synthesis involves a carrier with a fixed frequency α and a modulator with a fixed frequency β. When the amplitude of the modulator changes, it modulates the carrier's phase, leading to the generation of sidebands at different frequencies around α. The modulator wave's amplitude and frequency characteristics determine the sidebands' specific pattern and distribution. The amount of frequency modulation determines the extent of the spectral changes and the complexity of the resulting sound. Higher modulation amounts lead to a richer and more complex timbre with more pronounced sidebands in the modulated carrier.

In case α and β are not in a rational ratio, it is possible to generate inharmonic signals, which might sound like an interesting property in the case of drums and cymbals synthesis. However, the inharmonicity follows a pattern that does not match the drums or cymbals' frequency structure, or it would require several FM blocks in parallel.

Sample-based synthesis

Sampling in audio synthesis refers to capturing a small portion of an audio recording and reusing it in a new recording or inside a wavetable. The most common case where we find sampling is in electronic music genres, where a producer takes a snippet of an existing recording and uses it as the basis for a new track. Sampling can be used to create new melodies, rhythms, and sound effects and is a critical element of many modern music genres. In music production, the "Amen Break" has become one of the clichés of sampling [Collins, 2007].

In the context of synthesis, sampling could refer to wavetable synthesis (e.g., Fairchild CMI or XFer Serum) or a straight-up sound sampler (e.g., the AKAI S900 or Native Instruments Kontakt). Wavetable synthesis is a type of audio synthesis that uses preexisting audio samples, known as wavetables, to create sound [Andresen, 1979]. To create different pitches, the wavetable is played back with different step sizes in such a synthesizer [Roads and Strawn, 1996; Stilson and Smith, 1996]. It should be noted that the step size might not be an integer and would require interpolation between samples.

The synthesizer can modify the sound beyond pitches by adding audio effects, such as filters and envelopes. Wavetable synthesis is often used in electronic music, as it allows for a wide range of sounds to be created quickly and easily, especially with synthesizers cited before.

For the case of sample databases, also known as ROMpler (ROM Player), the sampler uses a database of actual recordings like [Hawthorne et al., 2019], which are then played back depending on several performance parameters such as pitch, velocity, or modulation. For example, string ensemble libraries use the modulation parameter to perform a *crescendo*. The idea of a ROMPler predates the era of digital samples with units like the Mellotron [Holmes, 2020]. A Mellotron is a keyboard instrument containing a set of tape recordings of real instruments, each covering a range of notes on the keyboard. A tape containing the recording of the corresponding note is played back whenever a key is pressed, selecting the appropriate tape at the appropriate speed to match the pitch.

Sample-based synthesis can generate highly realistic sounds (e.g., for drums, Toontrack Superior Drummer, or GetGoodDrums sample libraries). Still, it is limited by the inherent constraints of the recordings used as sources. Also, the libraries must be massive to

represent an exhaustive list of all the different ways a drummer can play the drum kit. For example, hitting a hi-hat stand can be found in some recordings (e.g., the introduction of The Offsprings - Come Out and Play) but is never considered in the recording of a drum library. Similarly, wavetable synthesis may employ a single wave, restricting the synthesis controls to pitch, filters, and envelopes.

1.4.2 Physics-based models for drum synthesis

Physical modeling is another approach to drum synthesis. Instead of letting an end-user play with different oscillator mixtures, a preexisting physical model is used to describe the acoustical behavior of the instruments. By solving their associated differential equations, it is capable of producing realistic-sounding examples. Beyond realism, such physics-based methods can provide some intuitive synthesis' controls, such as parameters for drum heads like tension or stiffness and excitation signal parameters such as strength or friction (e.g., brushes) [Roads et al., 2013].

However, the computational cost of these methods and their inherent limitations due to the modeling of real instruments [Smith, 2010; Bilbao, 2012; Bilbao and Webb, 2013] pose significant challenges. First, their focus solely on real instruments restricts their applicability and scope. Even though it would be possible to create sounds resembling unconventional instruments like a 10-inch wide by 16-inch deep snare drum, these models are still bound by the constraints of their underlying equations and resolution algorithms. In addition to their computational heftiness, physical models also encounter difficulties with their intricate control methods. While parameters directly related to physical properties, such as membrane stiffness, prove valuable to expert users familiar with the nuances of drum heads and tuning, an average end-user can easily become overwhelmed by the sheer number of parameters in complex but exhaustive physical models.

To make physical models more accessible, a more inclusive approach could incorporate higher-level controls, particularly perception-related ones. In theory, manipulating all physical parameters of a model to achieve consistent timbre alterations and control through perceptual features is feasible. However, there is a lack of practical exploration in this area, and physical models still impose significant demands on end-users and computers alike. To address these limitations, there is a need for further research and development to refine physical models and make them more user-friendly and adaptable to a broader range of musical goals. This is where physically informed deep learning could be used, but it is beyond this document's scope.

1.5 Deep Learning

Deep learning is a subfield of machine learning inspired by the brain's structure and function, specifically the *neural networks* that make up the brain. It involves training artificial neural networks on large datasets, allowing the neural network to *learn*. In this context, *learning* means using optimization methods such as stochastic gradient descent to adapt the weights to minimize an objective function.

Deep learning has achieved state-of-the-art results in many areas, including image and

speech recognition [LeCun et al., 1998; Krizhevsky et al., 2012; He et al., 2016], image and speech synthesis [van den Oord et al., 2018; Karras et al., 2020; Xiao et al., 2022], natural language processing [Stiennon et al., 2020], and even playing games such as chess and go [Silver et al., 2016, 2018]. It has also been applied to many real-world problems, including healthcare, finance, and transportation.

Required components of a neural network There are three essential components that are required to construct a neural network, all of which significantly influence its performance. The first is the network architecture, which determines the number of layers and the types of neurons used. These neurons and layers use weights as parameters to determine the contribution of their respective inputs in producing the output. During the training process, these weights are adjusted to influence the behavior of neurons and ultimately improve the network's performance according to the training criterion. The second component is the optimization algorithm, which adjusts the weights and biases of the network to minimize the loss function. Let us consider a classification task as an example. In this case, the optimizer would adjust the weights so that the predicted output aligns with the expected output according to the loss function. The third and arguably most critical component is the training dataset, which is utilized for training the network through the selected architecture and optimization algorithm.

Classic Architectures Several popular neural network architectures are commonly used in deep learning, including multi-layer perceptrons (MLP), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). The MLP uses a stack of fully connected layers, i.e., matrix multiplications plus bias term, with non-linearities, also known as activation functions, on the output of each layer. The universal approximation theorem for MLP [Cybenko, 1989] states that a neural network with a single hidden layer can approximate any continuous function on a compact input domain to arbitrary accuracy as long as a sufficiently large number of neurons is used. While MLPs were popular in the early days of neural networks, their popularity has waned recently as newer and more powerful architectures have emerged. However, this theorem demonstrating the immense representational power of MLPs provided a theoretical foundation for their early practical applications in machine learning and artificial intelligence.

A CNN is an evolution of the MLP, where convolution layers replace the fully connected layers. The convolution layer applies a convolution operation to the input data using a set of learned kernels, which are then transformed element-wise by a (non-linear) activation function to obtain a feature map. During backpropagation training [Kelley, 1960; LeCun, 1985], the convolutional layers learn these kernels, which represent filters, adjusting the weights of each to improve the overall network performance gradually. Convolutional neural layers achieve shift-invariance by restricting the receptive field of the hidden units to a local neighborhood, which enables them to extract local features efficiently. This organization allows reusing the same feature at different places. This was crucial in CNN's success in image and signal processing tasks. This contrasts with fully connected layers in multi-layer perceptrons, which do not have such a property.

Finally, there are RNNs. While MLPs and CNNs are feedforward architectures, RNNs have connections that allow a recurrence within a hidden layer. In other words, the output of a hidden layer is fed back into its input. This recurrence can be represented as a

graph with cycles, while a feedforward network is represented as acyclic graphs. It's worth noting that any RNN can be unrolled into a feedforward network, as shown in [Rumelhart and McClelland, 1987], and this property allows backpropagation training of RNNs [Goodfellow et al., 2016]. The recurrent architecture is better suited than the feedforward one in the case of sequence learning with long-term dependencies. Its evolutions like the Long Short-Term Memory [Hochreiter and Schmidhuber, 1997] show that the existence of an internal state is extremely interesting regarding sequence learning such as speech recognition [Sak et al., 2014]. The other interesting point for sequence learning is that RNNs can work with varying length inputs. However, it should be noted that RNNs are harder to train than MLP or CNN, mostly due to vanishing/exploding gradient and more complex architecture to take care of long-term dependencies compared to MLP or CNN.

A Data-Hungry Method One of the critical challenges in deep learning is the need for a large amount of labeled data to train the network effectively [Russakovsky et al., 2015]. However, it is possible to use transfer learning to circumvent this issue when the application allows it [Pan and Yang, 2010; Torrey and Shavlik, 2010]. Transfer learning entails retraining a pre-trained network on a generally smaller but application-specific dataset. It can also be used when a massive model has been trained already, and a new dataset is available. Transfer learning is useful in this case to avoid retraining the model from scratch, as transfer learning can yield better performances than random initialization [Yosinski et al., 2014].

Overall, deep learning has dramatically advanced the field of machine learning and has the potential to solve a wide range of problems in various contexts. One significant advancement in deep learning is the emergence of a new generation of generative models that can be applied to various tasks, like image, text, or audio generation. Among these models, some can be used for drum synthesis, and particularly for this thesis, the Generative Adversarial Networks [Goodfellow et al., 2014]

1.6 Sound Perception and subjective evaluation

This section will briefly review how humans perceive sound and how to get perceived quality assessments and psychophysical measurements when working with audio signals.

1.6.1 Sound and Hearing

First, we will describe how a sound can be described as a physical phenomenon. A sound is a vibration that propagates as an acoustic wave through a material medium (gas, liquid, or solid) [ANSI/ASA S1.1-2013, 2020]. This acoustic wave propagates as pressure variations, and in the case of air, through air pressure variations. The human ear can respond to a pressure variation stimulus one billion times less than atmospheric pressure. On the other end of the spectrum, the pain threshold amounts to 1 millionth of this quantity but is still a thousandth of atmospheric pressure [Rossing, 2000]. The sound pressure level (SPL) is generally given in decibels (dB-SPL). In this case, 0dB corresponds to $2 * 10^{-5} Pa$ at $1000 Hz$ and is the threshold of human hearing.

While SPL can be accurately measured, as it is a pressure measurement, loudness is a subjective quality. In this context, loudness can be defined as the auditory perception of a sound pressure level. Since it is related to auditory perception, i.e., hearing, two individuals might perceive loudness differently at a given SPL level. Loudness also depends on the frequency of the sounds: for the same SPL level, 50Hz and 1000Hz sine waves will give different perceived loudness. This phenomenon is described in the ISO standard 226 [ISO 226, 2003]. The loudness of a sound is not described as a physical occurrence, but as the result of the physical stimulation of the hearing mechanism that leads to the perception of sound. All in all, a sound can be described considering two aspects:

- A physical phenomenon bound to acoustic waves
- The perception of the aforementioned acoustic waves

1.6.2 Subjective evaluation of quality

In speech and audio processing applications where humans are the end-users, these end-users are also the ultimate evaluation of performance and quality. In this context, subjective evaluation refers to experimental setups where human subjects evaluate or quantify performance and quality using the subject's senses. From a high-level perspective, the task is simple; subjects are asked to answer questions such as:

- Does X sound good?
- How good does X sound?
- Does X or Y sound better?
- How intelligible is X?

In general, subjective evaluation in speech and audio refers to the perceptual assessment of sound quality [Giordano et al., 2012]. However, it is important to note that there are situations, such as in telecommunications, where the evaluation focus may be on the system's intelligibility, such as during participation in a dialogue over a telecommunication network.

Aspects of quality

To evaluate subjective quality aspects, the appropriate evaluation questions are tightly bound to the target application and context. For example, the quality requirements for a teleconference system differ from those in a music production environment or even medical applications like hearing aids. As most subjective evaluation methods of perceived quality are targeted toward speech synthesis, we will describe the different techniques and requirements in that context.

The perceived quality of an audio signal can be described, for example, through characteristics such as:

- Noisiness, which describes the amount of noise the speech signal is perceived to have (synthesis artifact or perceptually uncorrelated noise, unlike sibilant, for instance)
- Distortion, which describes how much speech signal parts are destroyed (perceptually correlated noise). However, it is relatively common to consider uncorrelated noises as distortions.
- Intelligibility, which describes the level to which the meaning of the speech signal can be understood.
- Listening effort, which refers to the amount of work a listener has to use in listening to the signal and how much listening fatigue and annoyance a user experiences,
- Resemblance, which describes how close the speech signal is to the original signal
- Naturalness, which describes how natural an artificial speech source sounds, often used as a last resort when no other adjective feels suitable.

Note that we can also think of intelligibility and listening effort as different aspects of sound quality. Indeed, we can perceive noisiness and distortions but still not be annoyed by the quality and can listen to the sound effortlessly. The listening effort is usually related to the loss of intelligibility. If the end-user needs to listen carefully, it becomes exhausting in the long term. However, context plays a crucial role in determining these levels and aspects of quality. Intelligibility is a fundamental quality aspect; communication cannot happen without it. Once achieved, individuals may seek high-quality communication for better comprehension, engagement, and reduced listening effort.

To evaluate the performance of an audio synthesis model, it is essential to compare the generated samples with known recorded samples of the target instrument. In this case, the evaluation aims to measure how accurately the model can reproduce the instrument's sounds. For instance, in the context of neural instrument synthesis, comparing synthesized sounds to either training data or data obtained through already available methods [Renault et al., 2022; Lavault et al., 2022a] is an answer to "Which method sounds better?". However, in the context of music production, generating samples that are not necessarily realistic may be interesting for exploring sound spaces. In this case, quality might not be as much of a concern compared to the case of speech or musical instrument synthesis. For instance, the ease of control with a neural synthesizer can also be measured, regardless of the output quality [Ramires et al., 2022].

Choice of Test Participants

When designing a subjective listening experiment, one of the first and most important questions is the choice of participants, mainly whether the listeners are either naive or expert. By naive listeners, we refer to listeners who do not have prior experience in analytic listening or expert knowledge of the subject tested. On the other hand, expert listeners are subjects trained in critical listening and can evaluate minor differences in sound samples through ear training or domain knowledge.

Expert listeners are, of course, the most sought-after listeners because they can give accurate and repeatable results and might also provide feedback on the artifacts and distortions found during the listening tests.

However, the problem is that such valuable listeners are rare and do not generally represent the end-user capabilities well. It is uncertain whether the preferences of expert listeners align with average users. For example, expert listeners might notice a distortion, like an exaggerated sibilance or resonance that an average user would not perceive due to lack of ear training.The same can also be said about music mastering where expert engineers can perceive distortion effects of dynamics compressor when the average listener cannot [Hjortkjær and Walther-Hansen, 2014; Katz, 2014] The expert would therefore be unable to enjoy the system under test, while it would be deemed excellent for the average user.

In music production, the target audience is typically end-listeners who may not have a technical background in music production. Therefore, the terms 'naive' and 'expert' may not be the most relevant for subjective evaluation in this context. However, if the clients or target audience are music producers, it may be appropriate to consider experts as the relevant group of testers. These specialized groups of testers would be extremely beneficial to experiments aiming to evaluate perceptual control mechanisms [Ramires et al., 2022]. If the clients or target audience are musicians, knowing the intricacies and expected behavior of the tested instrument is crucial for evaluating the sound quality of synthesis models. Such understanding can provide valuable insights into the shortcomings of certain methods compared to others.

1.6.3 Evaluation of Stimuli Perception

In terms of subjective evaluation, we can go beyond subjective quality measurement and enter into the field of psychophysics. Psychophysics is a branch of psychology that studies the relationship between physical stimuli and the sensations and perceptions they produce. It is concerned with how the physical characteristics of stimuli, such as their intensity, frequency, and duration, are related to the psychological experiences they produce, such as the sensations of loudness, brightness, and pain. In the context of this thesis, the main research question related to the field of psychophysics is whether individuals can distinguish between different levels of stimuli.

As an example, we will focus in this section on a simple attribute of sonic perception: the loudness of sound, since we can easily translate this example to various sonic interactions.

Perception Thresholds

As described in [Giordano et al., 2012], the perception of a sound involves two thresholds: the absolute threshold, i.e., the smallest or highest detectable value of a stimulus attribute (e.g., the lowest detectable sound level or the highest level that causes pain or damage), and the differential threshold, i.e., the smallest discriminable difference in a stimulus attribute (e.g., the smallest discriminable difference in level). The absolute threshold can be considered a particular case of the differential threshold.

When measuring an absolute threshold (in our example, the absolute threshold for sound level), test participants are repeatedly presented with a small set of stimuli ranging from near silence to clearly perceivable. In this case, participants are asked if they detect a stimulus. In the case of the differential threshold, the overall method changes a bit. In an experiment aimed at measuring the differential threshold, on each trial, the participants

are presented with two stimuli: a standard stimulus whose properties remain constant across all trials (e.g., a 60dB SPL sound) and a comparison stimulus that varies from trial to trial (in our example, a sound from a set ranging from barely quieter to barely louder than the standard stimulus). Participants indicate in which of the paired stimuli the target attribute has the largest or the lowest value. For our loudness perception example, the question can be which of the two stimuli is louder.

Experimental Setup

The simplest of all psychometric experiments is the constant stimuli method [Giordano et al., 2012], which follows the experience described above. This method is called constant stimuli because it refers all measurements to a reference stimulus. Once the experiment has been performed, the data gathered can be analyzed using a fitted psychometric function. A psychometric function is a mathematical function that relates an individual's performance on a psychological or psychophysical test to the level of some psychological construct, such as loudness. These functions are typically used to evaluate the validity and reliability of psychological tests and to understand how an individual's performance on a test relates to their underlying psychological abilities. The absolute threshold can thus be defined as the stimulus intensity, which is perceived 50% of the time (in other words, the stimulus intensity where the decision is purely random). The differential threshold is defined as the average of the stimulus values judged louder than the standard 25% and 75% of the time.

In most cases, none of the results are directly linked to the precise response probabilities for calculating the thresholds. In part for this reason, and partly for the need to integrate experimental data across all the investigated stimuli, a psychometric function is usually fitted to the observed response probabilities for all stimuli with a cumulative normal distribution, for instance, among others [Wichmann and Hill, 2001a,b]). In this case, the threshold measures are derived from the parameters of the fitted function.

Of course, the method of constant stimuli has many shortcomings. The first of the list is its susceptibility to biases. For this, more advanced techniques exist, but go beyond the scope of this chapter.

1.7 Contributions

The contributions of this thesis can be listed as follows:

- New drum sound data set with objective and subjective measurement of hit velocity and extended tuning annotations described in Section 3.4
- An adaptation of a state-of-the-art deep image generator for direct waveform generation [Lavault et al., 2022b], found in Section 4.2
- Objective and subjective evaluation of the sound quality of the proposed method demonstrating significant improvement compared to the state of the art [Lavault et al., 2022a,c] shown in Section 4.2

- A deep-learning method for high-level control with timbral features of neural drum synthesis, improving on state-of-the-art neural drum synthesizers in terms of objective measurements of controllability [Lavault et al., 2022b] in Section 4.3
- Subjective evaluation of the perception of the error margin on high-level control parameters and comparison to objective measurements concluding on the imperceptibility of the average error in most cases [Lavault et al., 2022c], found in details in Section 4.3
- An extension of the aforementioned high-level control to incorporate dynamics control in a neural drum synthesizer in Section 4.5.

Chapter 2

State of the Art

2.1 Introduction

This chapter describes general neural network audio synthesis methods and their application to drum synthesis. In addition to studying the existing contributions for synthesis tasks, we will briefly review the control methods of generative neural networks. Finally, we will describe subjective and deep-learning-based objective evaluation metrics of generation quality.

2.2 Synthesis with deep neural networks

This section will describe different synthesis methods involving neural networks and how to control these neural synthesizers.

2.2.1 WaveNet

The WaveNet model, initially presented in [van den Oord et al., 2016], was primarily developed for synthesizing speech in text-to-speech systems. The general idea of WaveNet is to condition the generation of a given sample by all the previous samples. This can be mathematically represented as the joint probability of a waveform $x = \{x_1, \cdots, x_T\}$ factorized as a product of conditional probabilities as follows:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}) \tag{2.1}$$

Therefore, each audio sample x_t is conditioned on the samples at all previous time steps.

To that end, the authors introduced causal convolutions, ensuring the model cannot violate the ordering in which the data is modeled. This can be done by zero-padding the signal before the convolution operation appropriately.

In addition to causal convolutions, the authors also used dilated convolutions. A dilated convolution is a convolution operation applying a filter that covers a larger area than

its actual length, achieved by skipping input values at specified intervals, also known as dilation rate. This convolution operation can be considered equivalent to using a larger filter derived from the original filter but expanded with zero values. This offers the advantage of increasing the receptive field of the convolutional layers, while keeping the actual filter size and number of necessary layers small.

These two special convolutions combined are illustrated in Fig. 2.1. The causal part means the element at a time step t is only dependent on the past events and not the future, which in the figure can be visualized as a clear left-to-right structure (i.e., past to present). The dilated convolution can be considered skipping part of the connections between the elements while still keeping the causal relationship, which here means an element can only be connected to an element that is either at time t or before. All of this makes WaveNet an autoregressive model. It is worth noting that contrary to an RNN, WaveNet has no recurrent path, which helps keep the training process simpler.

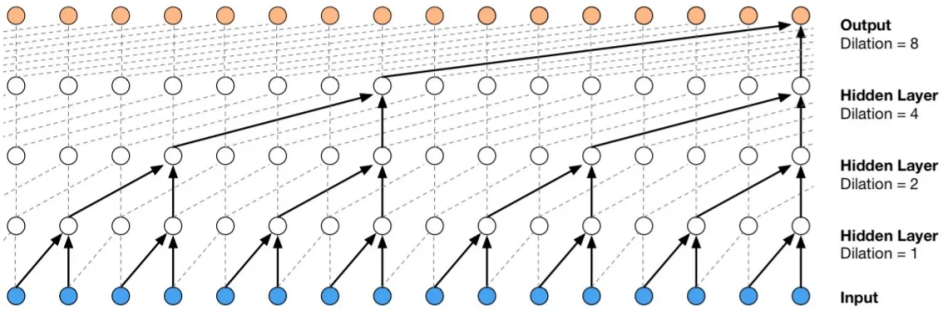


Figure 2.1: Visualization of a stack of dilated causal convolutional layers, reproduced from [van den Oord et al., 2016], figure 3

By means of conditioning the network, the authors could use it for text-to-speech synthesis but also showed preliminary results on music synthesis without any formal evaluation apart from the demonstrations in their supplementary material.

The first formal use of WaveNet for musical instrument synthesis was in [Engel et al., 2017]. It is also this article that introduced the NSynth dataset. NSynth is a massive data set (300k samples) of musical notes over 1000 unique instruments. The samples are four seconds-long monophonic 16kHz files. Still, none of the instruments in the datasets are among the instruments of interest for this project, making this dataset less useful.

Using the NSynth dataset, [Engel et al., 2017] compared a spectral-based autoencoder to a modified WaveNet autoencoder. In addition to the WaveNet architecture described in [van den Oord et al., 2016], [Engel et al., 2017] used an encoder that takes raw audio waveform as input, from which this encoder produces an embedding. This embedding controls then the WaveNet decoder, which is simply a modified version of WaveNet described above, which then becomes:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}, f(x)) \tag{2.2}$$

where $f(x)$ is the embedding from the encoder. The embedding $f(x)$ is then used to bias every layer in the WaveNet Decoder with a different linear projection of said embeddings. The overall network architecture is shown in Fig. 2.2.

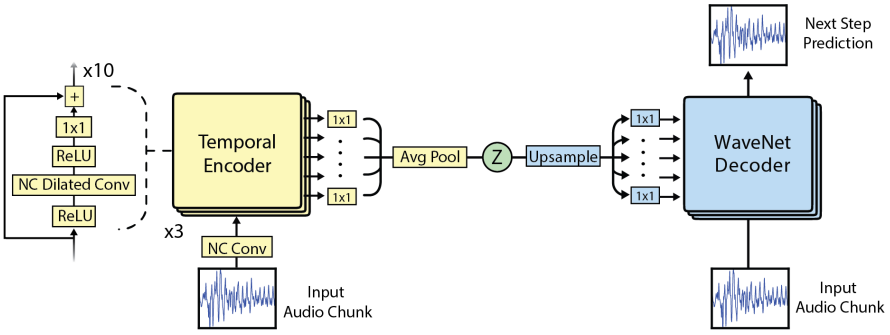


Figure 2.2: WaveNet autoencoder. Note that the embeddings are distributed in time and upsampled to the original resolution before biasing each decoder layer. Reproduced from [Engel et al., 2017], figure 1b.

To measure the reconstruction quality, the authors used an external classifier on the pitch values and a setup similar to the Inception Score to estimate quality. It should be noted that the authors did not provide any subjective evaluation of quality.

Back to drum synthesis, there are no instances of WaveNet specifically used for drum synthesis.

2.2.2 Auto-Encoders

Deterministic Auto-Encoder

An autoencoder (AE) is a neural network designed for generative and dimensionality reduction tasks [Bourlard and Kamp, 1988; Vincent et al., 2010]. It is composed of an encoder and a decoder. The encoder maps the input data to a latent space, representing the data's essential features. The decoder then maps this latent representation back to the original data space, reconstructing the input data. If the autoencoder sets its output equal to the input for all cases, it doesn't add much value. Usually, they are restricted to approximate copying and only copy the inputs similar to the training data.

When the latent space is of a smaller dimension than the data, the autoencoder is called under-complete. The low-dimensional latent space forces the autoencoder to capture the most salient features of the training data [Goodfellow et al., 2016].

The learning process is described simply as minimizing the following loss function:

$$L(x, Decoder(Encoder(x))) \tag{2.3}$$

L represents a loss function penalizing the dissimilarity between the output $Decoder(Encoder(x))$ and the target input x. A commonly used loss function for this purpose is the mean squared error.

Interestingly, when the decoder is linear, and the mean squared error is employed as the loss function, an autoencoder can capture the same subspace as Principal Component Analysis (PCA). In this scenario, an autoencoder trained to perform the copying task

inadvertently learns the principal subspace of the training data as an additional outcome. Using non-linear encoders and decoders can yield non-linear and more general versions of PCA [Goodfellow et al., 2016]. In other words, auto-encoders described above are deterministic and mostly useful for dimensionality reduction. It should be noted that they cannot be used as generative models per se as they do not define a distribution.

Variational Auto-Encoder

Variational Autoencoders (VAEs) [Kingma and Welling, 2014] are generative models that learn to generate new data samples by leveraging their probabilistic nature, distinguishing them from autoencoders. Let x represent the training data, z denote the latent variables, and $p(x, z)$ be the parametric model governing their joint distribution, referred to as the generative model. This model is defined over these variables. We aim to perform maximum likelihood learning to determine the generative model parameters. Specifically, we aim to maximize the marginal likelihood of the dataset X, denoted as $\log p(X)$, which is computed by summing the logarithm of the generative model probability for each $x \in X$, i.e., maximizing the following equation:

$$\log p(X) = \sum_{x \in X} \log(p(x)) \tag{2.4}$$

However, computing this likelihood directly is usually impractical due to its intractability. To address this, we introduce an inference model, denoted as $q(z|x)$, a parametric model defined over the latent variables. We optimize the variational lower bound on the marginal log-likelihood for each observation x, denoted as $\log p(x)$. The lower bound, denoted as $L(x, \theta)$, is given as follows:

$$\log p(x) \geq E_{q(z|x)}[\log p(x, z) - \log q(z|x)] = L(x, \theta) \tag{2.5}$$

Here, θ represents the parameters of the generative model p and the inference model q. When the latent variables z are continuous, we can optimize the lower bound using a re-parameterization of $q(z|x)$, as introduced in [Kingma and Welling, 2014].

The re-parameterization, known as the re-parameterization trick, enables backpropagation with neural networks. The re-parameterization trick involves expressing the distribution $q(z|x)$ as a deterministic mapping $g_\varphi(\varepsilon, x)$ of a continuous random variable ε. For example, in the case of a univariate Gaussian distribution, instead of directly sampling z from $q_{\mu,\sigma}(z) = N(\mu, \sigma^2)$, we can rewrite it as $z = g_{\mu,\sigma}(\varepsilon) = \mu + \varepsilon \cdot \sigma$, where $\varepsilon \sim N(0, 1)$.

This approach, which optimizes the variational lower bound using a parametric inference network and re-parameterization of continuous latent variables, is commonly referred to as VAE. When rearranging the lower bound $L(x, \theta)$, we end up with:

$$\begin{aligned} L(x, \theta) &= E_{q(z|x)}[\log p(x, z) - \log q(z|x)] \\ &= E_{q(z|x)}[\log p(x|z) + \log p(z) - \log q(z|x)] \\ &= E_{q(z|x)}[\log p(x|z)] - E_{q(z|x)}[\log q(z|x) - \log p(z)] \\ &= E_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) \end{aligned} \tag{2.6}$$

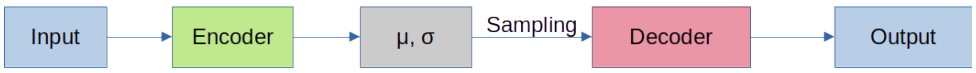


Figure 2.3: Visualization of the architecture and sampling from a VAE. The (μ, σ) block represents the output of the encoder, that are the parameters of a multidimensional Gaussian distribution.

The first term in eq. (2.6) can be interpreted as the expectation of the negative reconstruction error, while the Kullback-Leibler divergence term $D_{KL}(q(z|x)||p(z))$ acts as a regularizer. To align with the terminology of autoencoders, this generative model can be viewed as a regularized autoencoder, with $q(z|x)$ serving as the encoder and $p(x|z)$ as the decoder. To sample from a VAE, randomly generate a latent variable using the mean and standard deviation predicted by the inference model q and pass it through the decoder p to generate a new data sample. Regarding control, the latent representation inside the VAE is interesting because it gives external control without complex methods, the latent space being smooth by design. Figure 2.3 presents a VAE architecture and sampling method schematic.

Wasserstein Auto-Encoder

The Wasserstein Auto-encoder (WAE) is a recent evolution of auto-encoders that extends the idea of regularized auto-encoders. It was introduced in [Tolstikhin et al., 2018], who proposed two ways to regularize the network: one using a critic network, inspired by the Generative Adversarial Network (GAN) [Goodfellow et al., 2014; Gulrajani et al., 2017], and another using the Maximum Mean Discrepancy (MMD). The WAE architecture is designed to learn a continuous and smooth latent space distribution, which can be used to generate high-quality data. Regularizing the latent space distribution with the Wasserstein distance allows the WAE to generate less blurry images than traditional Variational Auto-Encoders (VAEs). The critic network approach involves training a separate discriminator network to distinguish between samples from the prior distribution and the learned latent space distribution, hence penalizing generated samples far from the target distribution. Alternatively, the MMD approach measures the difference between the empirical distribution of the latent space representations and the prior distribution. The MMD penalty encourages the learned latent space to be close to the prior distribution by penalizing large differences between the two distributions.

Auto-Encoders for Drum Synthesis

The first strong proposition for drum synthesis was [Aouameur et al., 2019]. Here, the generative model consists of two networks: a Conditional Wasserstein autoencoder (CWAE), which learns to generate Mel-scaled magnitude spectrograms, and a Multi-Head Convolutional Neural Network (MCNN), which estimates the audio from the computed magnitude spectrogram. They also provided a creative interface using the most significant axes obtained with a PCA on the latent variable. However, the authors of [Aouameur et al., 2019] didn't provide objective or subjective quality measurements apart from demo sounds.

Currently, no known examples of members of the auto-encoder family being used to gen-

erate audio with external timbral features. Some studies, such as [Aouameur et al., 2019; Caillon and Esling, 2021], have developed interfaces to easily control audio generation by exploring the latent space and the release of Max4live plugins to the general public. However, these interfaces do not provide the capability for explicit control over timbral properties. While some dimensions of the latent space may correspond to a perceptible variation in timbre, it is not explicitly controllable. The same can be said about dynamics control, through MIDI velocity or not.

2.2.3 Adversarial models

Generative Adversial Networks

Initially described in [Goodfellow et al., 2014], Generative Adversarial Networks (GAN) are an approach for obtaining generative models via the adversarial training of two models. One of these two is an actual generative model G, called the generator, capturing the target data distribution, and a discriminative model D called the discriminator that estimates the probability that a sample came from the training data rather than the generative model. This framework is a minimax two-player game with the value function $V(G, D)$. A possible formulation for this can be found in [Goodfellow et al., 2014]:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z}(z)[\log(1 - D(G(z)))] \tag{2.7}$$

Where:

- p_{data} represents the distribution over the training data
- p_z represents the distribution over the input variables, often containing random noise.

Moreover, $D(x)$ represents the probability that x came from the data p_{data} rather than p_g, where p_g is the distribution induced by the generator. This can be rewritten as a two-step optimization process:

$$\begin{aligned} \min_D V_{GAN}(D) &= -\mathbb{E}_{x \sim p_{data}(x)}[\log(D(x))] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ \min_G V_{GAN}(G) &= \frac{1}{2}\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \end{aligned} \tag{2.8}$$

which is the training algorithm used in practice instead of the direct minimax game. A visualization of the GAN framework can be found in Fig. 2.4.

Evolutions of GANs

When GANs were first presented, they were a groundbreaking contribution, but soon after, rapid developments were made to improve the results, particularly in image synthesis. The

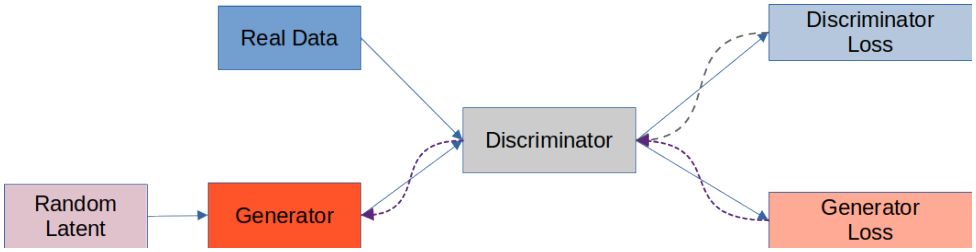


Figure 2.4: Visualization of the GAN framework. The grey dashed line represents the backpropagation for the discriminator training, and the purple dashed lines represent the backpropagation for the generator training.

first thing that saw improvements was the adversarial loss. It was found that the original loss function using the sigmoid cross-entropy may lead to vanishing gradients problems during the learning process.

Improvement on the adversarial loss One of the first propositions was the Least-Square GAN (LSGAN) [Mao et al., 2017] to deal with the vanishing gradient problem found with the original formulation of the GAN. By replacing the cross-entropy from eq. (2.8) with a square function, the vanishing gradient problem is mitigated. In this case, the overall objective becomes:

$$\begin{aligned} \min_D V_{LSGAN}(D) &= \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[D(G(z))^2] \\ \min_G V_{LSGAN}(G) &= \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-1)^2] \end{aligned} \tag{2.9}$$

The authors achieved better image quality and more stable training using the least-square loss (eq. (2.9)) when compared to the original GAN formulation.

Another variation on the GAN loss function was the Wasserstein GAN (WGAN) [Arjovsky and Bottou, 2017]. With the same goal as LSGAN, WGAN is an approach to stabilize GAN training and improve the overall quality of the generative model. The key idea behind WGAN is using the Wasserstein distance to measure the distance between the true and generated probability distributions. Unlike traditional GANs that use the Jensen-Shannon divergence, Wasserstein GANs use a more reliable distance metric to train the generator and discriminator networks. The generator learns to generate samples closer to the true distribution by minimizing the Wasserstein distance between the two distributions.

Compared to the original formulation of GANs, using the Wasserstein distance brings several advantages. Firstly, it helps avoid the vanishing gradients caused by the logarithm function in the original GAN loss. Vanishing gradients can be a significant issue in GANs as they can slow down the learning of the generator and may cause it to converge to suboptimal solutions. By contrast, the Wasserstein distance smoothes the objective function (it becomes linear when LSGAN is quadratic while the original GAN is logarithmic), easing the training of both the generator and the discriminator. Moreover, using the Wasserstein distance produces more robust and less overpowered discriminators. In vanilla GANs, the training process encourages the discriminator to become too efficient,

often leading it to achieve near-perfect accuracy. As a result, it can become challenging for the generator to produce samples that will fool the discriminator. Thus, mode collapse and a lack of diversity in generated samples can occur. Since Wasserstein GANs use a discriminator whose aim is not to classify real and fake data but to estimate how far or close the real and fake distributions are, allowing the generator to produce varied outputs that still fall within the probability distribution of the real data.

Mathematically, the Wasserstein GAN [Arjovsky and Bottou, 2017] can be described as:

$$\min_G \max_D \left[\mathbb{E}_{x \sim P_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z \sim P_z(z)}[D(G(z))] \right] \tag{2.10}$$

subject to:

$$\forall x, y \in \mathcal{X} : |D(x) - D(y)| \leq |x - y| \tag{2.11}$$

where G is the generator network, D is the discriminator network, $P_{\text{data}}(x)$ is the distribution of real data, $P_z(z)$ is the prior distribution of noise input, $\mathcal{X}$ is the space of input data. However, the 1-Lipshitz constraint on D described in eq. (2.11) is unusable in terms of computation. To this end, WGAN-GP (for Wasserstein GAN with Gradient Penalty) was introduced in [Arjovsky and Bottou, 2017].

The gradient penalty can be described as follows:

$$\mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}} \left[(||\nabla_{\tilde{x}} D(\tilde{x})||_2 - 1)^2 \right] \tag{2.12}$$

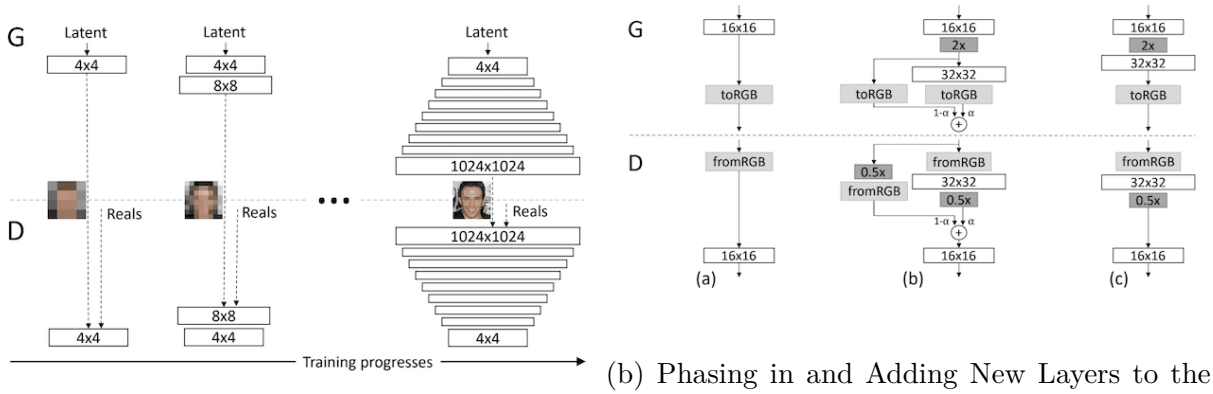
where $\tilde{x}$ is a random interpolation between a real and a generated sample, $P_{\tilde{x}}$ is the distribution of $\tilde{x}$, and $\nabla_{\tilde{x}} D(\tilde{x})$ is the gradient of the discriminator's output with respect to $\tilde{x}$. In other words, WGAN-GP penalizes the discriminator if its gradient is not 1 between the real data space and the generated data space. This is not a Lipshitz constraint, which in turn was described in [Petzka et al., 2018] with WGAN-LP, where the penalty becomes:

$$\mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}} \left[\max(0, ||\nabla_{\tilde{x}} D(\tilde{x})||_2 - 1) \right] \tag{2.13}$$

Network architecture and training Beyond the new losses, the subject of GANs saw two major evolutions of importance for the present thesis and the overall generative model ecosystem: the progressive growth of GANs (PGAN) [Karras et al., 2018] and StyleGAN (SGAN)[Karras et al., 2019, 2020].

The first evolution of interest to us was the progressive growth of GAN. In this framework, the adversarial process is not altered, but there is a twist during training. As the name suggests, the critical notion is progressively growing the generator and discriminator. By starting from a low resolution (in the paper, 4×4 images) and adding new layers of higher resolution after a fixed number of training steps, this model was able to learn increasingly fine details. The overall process is shown in Fig. 2.5a while the addition of new layers is detailed in Sect. 2.2.3. This method reportedly sped up the training process as well as stabilized it. This sped-up and stabilized network allowed the authors to generate high-resolution images up to 1024×1024.

However, the success was short-lived, as the next evolution arrived with StyleGAN [Karras et al., 2019]. While conserving the progressive-growing methodology, the authors divided their generators into mapping and synthesis networks. In StyleGAN, the input noise vector is not directly transformed into an image but instead into an intermediate latent



(a) Effect of Progressively Adding Layers to the Generator and Discriminator

(b) Phasing in and Adding New Layers to the Generator and Discriminator Models in a PGAN network.

Figure 2.5: Visualization of the "Progressive Growing of GANs," reproduced from [Karras et al., 2018]

space $\mathcal{W}$ by the mapping network. By manipulating the values of these components in the intermediary latent space, the generated images can be modified to exhibit specific desired attributes without explicit conditioning. For example, increasing or decreasing a particular component can change the level of detail in the generated image or modify its color palette. This level of control enables users to guide the generation process and influence specific characteristics of the generated images.

The intermediary latent is used inside the synthesis generator through Adaptative Instance Normalization (AdaIN) to achieve such result. The key idea is to transform an intermediary latent vector $w \in \mathcal{W}$ into a *style* through a learned affine transform, which gives us the style vector $y = (y_s, y_b)$. This style vector controls the AdaIN operation after each convolution operation. We then have:

$$AdaIN(x_i, \underbrace{y}_{=(y_s,y_b)}) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \tag{2.14}$$

Each feature map x_i is normalized separately and then scaled and biased using the corresponding scalar components from style y, separated into y_s and y_b as scale and bias style terms.

StyleGAN was the first to enable intuitive and scale-specific control of the synthesis. Quickly afterward, StyleGAN2 superseded StyleGAN [Karras et al., 2020], where several improvements were brought up and produced high-resolution photo-realistic face images. In addition to these improvements, the authors also showed the limits of Progressive Growing and PGANs and reverted to a more straightforward training of the full network instead of the per-layer progressive training.

Finally, it is worth noting the original article [Goodfellow et al., 2014] did not introduce any means to control the content to be generated. Later proposals introduced the possibility of controlling the generation. This will be detailed in Sect. 2.2.6.

GANs for general audio synthesis

Although GANs have been predominantly used in tasks related to image generation, their application in audio synthesis has also been significant. Regarding instrument synthesis, the first notable effort was GANSynth [Engel et al., 2019]. GANSynth uses a Progressive GAN base to generate real-time audio signals, allowing users to control different aspects of the generated sound. It was trained on NSynth [Engel et al., 2017] and can generate a wide range of sounds, from individual notes to complex melodies and chords. It is an example of how GANs can be used for audio generation and manipulation, and represents an important step towards making machine learning more accessible for musical applications.

Drum Synthesis with GANs

Of course, GANs are also of use for drum synthesis. In this context, the first example of GAN was [Donahue et al., 2019], where a GAN was used for several audio generation tasks, including drum synthesis, in the temporal domain.

Following the advancement allowed by PGANs, Nistal et al. [Nistal et al., 2020] proposed a spectral-domain generator for drum sounds incorporating high-level control features from the AudioCommons feature set [Pearce et al., 2016]. In parallel, Drysdale et al. [Drysdale et al., 2020] also used a PGAN for drum synthesis in the temporal domain for synthetic sounds only. A newer version backed with a VST plugin was released in 2022 [Nistal et al., 2022].

Since StyleGAN, the successor of PGAN, showed significant improvements in image quality, the results on drum synthesis also followed suit [Drysdale et al., 2021]. The goal here was to explore the use of style inversion for creative synthesis.

2.2.4 Differentiable Digital Signal Processing

Google's Differentiable Digital Signal Processing (DDSP) [Engel et al., 2020] is another approach to enhance machine learning models' audio signal processing capabilities by making these audio processing blocks differentiable and hence compatible with deep learning frameworks. DDSP uses DSP blocks to synthesize sounds and a deep neural network to control the synthesis models, allowing for a clear division of tasks. Unlike additive synthesis, where adjusting hundreds of parameters was difficult given a set of controls like instrument type, pitch, and velocity, DDSP models with learned controls facilitate music synthesis while reusing well-known additive synthesis techniques and making them easier to use.

Compared to autoregressive, variational, or adversarial models, the DDSP approach utilizes naturally strong domain knowledge and integrates it with deep learning rather than using the deep neural network capabilities to learn this knowledge from the training data.

DDSP allows new ways to integrate standard audio processing blocks inside a deep-learning framework out-of-the-box, mixing the strong inductive biases of the signal-based approach and the expressivity of neural networks.

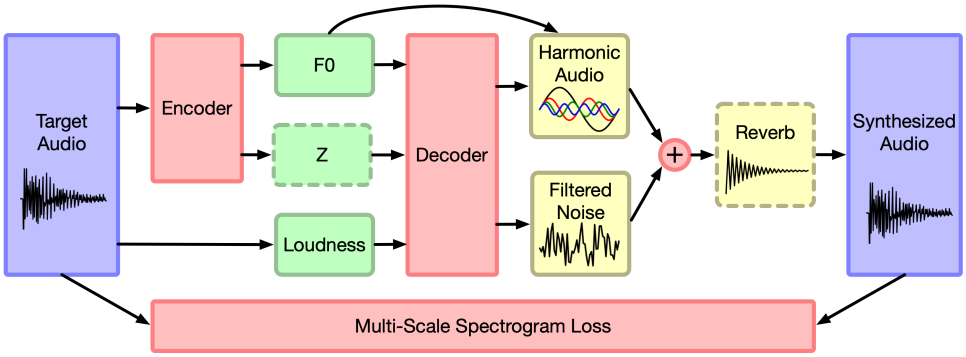


Figure 2.6: DDSP autoencoder architecture. Red blocks are part of the neural network architecture, green blocks are the latent representation, and yellow blocks are deterministically controlled synthesizers and effects provided by DDSP. Reproduced from [Engel et al., 2020].

The authors show that combining interpretable modules allows the manipulation of each separate model component. The applications include the separate manipulation of pitch and volume, the ability to realistically extend to pitches that were not seen during training, the capability to remove the effects of room acoustics without prior knowledge of the recorded room, the potential to apply the extracted room acoustics to different settings, and the timbre transfer.

To showcase the capabilities of this DDSP module, the authors provided a modified autoencoder and a multi-resolution spectral loss as shown in Fig. 2.6. The encoder is a mix of several specialized encoders that aim at estimating target parameters such as the F0. The decoder estimates the control parameters for the differentiable synthesizer and noise generator from this parameter set. Finally, the multi-resolution spectral loss function enables the network to learn more fine-grained spectral features of the audio, which can be essential for instrument modeling and timbre transfer tasks.

To summarize, DDSP provides a set of differentiable audio processing blocks, making them compatible with modern deep-learning frameworks. It allows for a clear separation between synthesis (with DSP blocks) and control tasks (with DNNs), making manipulating and modifying sound characteristics easier. Finally, an audio plugin for Digital Audio Workstations using a trained (and trainable) DDSP model is available.

For the moment, the main applications of DDSP apart from the ones described in the original paper are keyboard synthesis, either piano [Renault et al., 2022], or recreation of a classic FM synthesizer [Caspe et al., 2022]. As for drums, no example can be found in the literature yet.

2.2.5 Diffusion models

The first introduction of the diffusion models was in [Sohl-Dickstein et al., 2015], where the authors aimed to propose a novel approach that simultaneously achieves both flexibility and tractability. Diffusion models are, at their core, a generative Markov chain. This Markov chain converts a real-world distribution, implied by the training data, into a target data distribution, generally a Gaussian, through diffusion. The reverse process,

learned by a network, is called denoising and is the useful generation process here and is illustrated in Fig. 2.7.

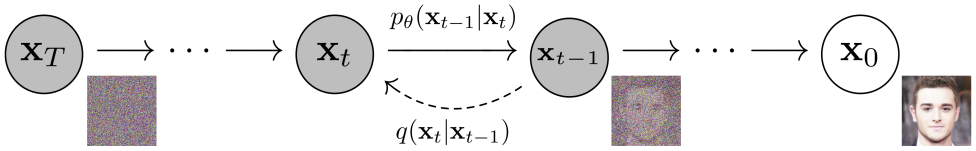


Figure 2.7: Diffusion probabilistic models are parameterized Markov chains trained to denoise data gradually. A neural network parametrizes the denoising model p_θ. Reproduced from the companion website of [Ho et al., 2020]

To align with the content depicted in Fig. 2.7 (reproduced from [Ho et al., 2020], we will now introduce the equations within a diffusion model. In this model, the chain denoted as q introduces minimal Gaussian noise during each step t. The parameters of the Gaussian distribution are adjusted based on a noise schedule controlled by $beta_t$, i.e., when and how much noise to add during the generative process. This schedule governs the variation in the noise characteristics. This iterative process is repeated for a total of T steps. As the number of time steps increases, more original input features are gradually eroded.

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) := \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2.15}$$

The goal of the denoising process p_θ is to learn the denoising steps to undo the forward process iteratively. In this way, the reverse process appears as if it is generating new data from random noise.

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) := (x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.16}$$

Given the similar form to a VAE, since an objective for p would be to maximize the likelihood, the ELBO loss seen with the VAE can be reused in this context [Ho et al., 2020].

While diffusion models offer a tractable method by defining the end point of the chain as the target distribution and making intermediate steps in the diffusion chain tractable, a drawback of this method is that it can be computationally expensive at inference time due to the number of steps, which may limit its practicality in certain contexts, mostly embedded or real-time applications.

Recent improvements in diffusion models like [Dhariwal and Nichol, 2021] have shown more than excellent results in image generation, significantly improving upon the state-of-the-art generative models like StyleGAN 2. In [Xiao et al., 2022], the authors' used a GAN to take care of the denoising, making the small steps in the denoising process larger. Most notably, this system could provide a mix of GAN and diffusion model qualities, i.e., inference speed and overall quality for GANs and mode coverage for diffusion, almost solving the classic trilemma of generative models: quality, mode coverage, and inference speed.

Following the successes for image generation, it was not long before diffusion processes were used for sound processing [Kong et al., 2021; Juanpere and Välimäki, 2022]. Drum

sound generation also benefited from diffusion models with [Rouard and Hadjeres, 2021]. In this article, the authors provided a diffusion model engineered for drum generation with supplementary controllable sampling schemes. In particular, what was proposed in this article was a class-mixing strategy, allowing the controllable creation of clear chimeric sounds and an inpainting method to generate different tails given a fixed attack portion. In this context, inpainting involves filling in missing or corrupted parts of the sound using the diffusion process. This inpainting capacity is especially interesting for sound design work.

2.2.6 Control strategies

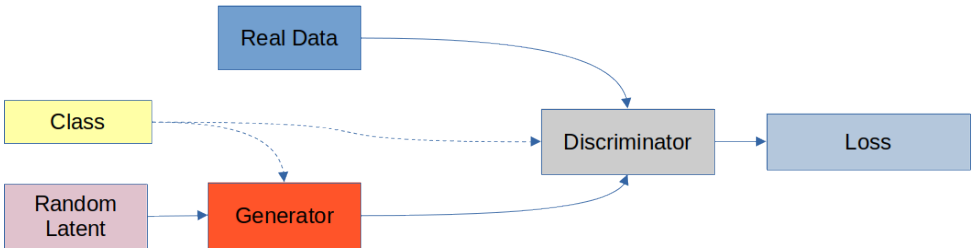
Controlling the output of a generative model can be difficult without resorting to supervised learning. Supervised approaches train the network to generate specific outputs given certain input conditions, making control over the generator more explicit. While VAEs are often praised for their explicit latent vector, allowing greater control over the output [Aouameur et al., 2019], it can be difficult to know which latent space dimensions correspond to specific controls.

Controlling GANs

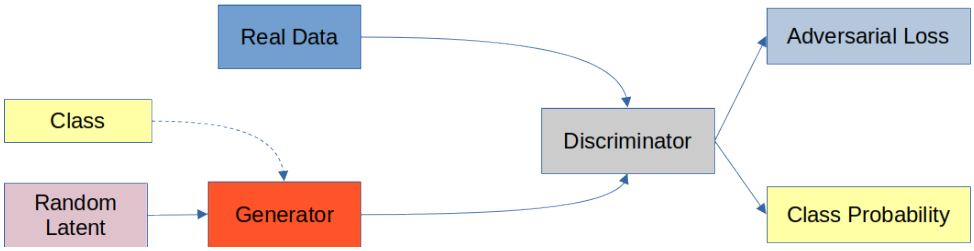
While vanilla GANs are a popular generative model in their own right, they generally don't have a latent space as well-behaved as the VAEs. The first example of a conditional GAN was in [Mirza and Osindero, 2014]. The idea here is to provide both the generator and discriminator with the labels to enforce a class-dependent generation and discrimination, as shown in Fig. 2.8a. However, the network has no incentive to follow this information through its loss.

To that end, the literature proposes a solution in the form of Auxiliary Classifier Generative Adversarial Network (ACGAN) [Odena et al., 2016]. ACGAN incorporates supplementary data such as class labels and utilizes an auxiliary classifier network in combination with the discriminator in addition to the class label concatenated to the latent in the generator, as shown in Fig. 2.8b. The auxiliary classifier network is responsible for classifying the generated images based on their respective class labels. This allows for control over the class label explicitly in the generator. While this method allows for supervised generation on discrete labels, it is incompatible with continuous variables due to its construction.

In [Mescheder et al., 2018], the authors used an even more straightforward method: concatenating an embedding of the labels to the latent and utilizing the label index to take the correct output from the discriminator different heads (one output head per class). This method was then reused with StyleGAN [Karras et al., 2019]. This is the closest method to providing a direct and intuitive way to control the generator output.



(a) Visualization of the Conditional GAN [Mirza and Osindero, 2014]



(b) Visualization of the Auxiliary Classifier GAN [Odena et al., 2016]

Figure 2.8: Visualization of GAN control-methods.

Controlling Diffusion Models

Auxiliary embeddings provide external control in diffusion models. This is illustrated in [Dhariwal and Nichol, 2021], where the authors integrated class information into normalization layers through an embedding. Additionally, the same paper introduced another technique called classifier guidance, which suggests utilizing gradients derived from a pre-trained classifier to motivate the network to acquire class-specific knowledge. It is important to emphasize that the classifier guidance technique can be used for unconditional and conditional synthesis. However, it is not a control method per se. Notably, this technique demonstrated a behavior of balancing diversity and quality, i.e., increasing one often comes at the cost of reducing the other. As another example, [Kong et al., 2021] used class conditioning similarly. The shared embedding is used in two ways: through 1×1 convolutions, mapping it to a different space, and as a bias term. Overall, the idea is to use embeddings of the control vectors to modify the activations to match.

Control over timbral features

When it comes to drum synthesis control, beyond the usual labels that are the type of instrument to generate like a snare drum, kick drum, hi-hat, or crash cymbal, the Audio-Commons feature set [Pearce et al., 2016] is a common choice for high-level control over the timbre. Prior research has explored timbral features for drum synthesis in [Ramires et al., 2020] and [Nistal et al., 2020]. These two studies employ neural drum synthesizers that can generate drum and cymbal sounds based on continuous feature control inputs. The methods use the feature values directly and leverage the expressive power of neural networks (a U-Net in [Ramires et al., 2020] and a GAN in [Nistal et al., 2020]) to utilize the control.

Ramires et al. used the features as input for their U-Net, allowing the network to learn the different associations without additional constraints. In contrast, Nistal et al. [Nistal et al., 2020] used the discriminator in their GAN for discrimination and feature esti-

mation. They also constrained the generator to follow the input to encourage using the features at the generator's input, similar to the approach in [Odena et al., 2016].

2.3 Evaluation Strategies for Generative Models

The evaluation of generative models proves to be a non-trivial undertaking. In the context of GANs, this is particularly valid, as they employ implicit density estimation, which consequently lacks a direct approach for evaluating the probability density of individual elements in the training set.

Although subjective evaluation is the most human-like way to measure sound quality, it can be influenced by individual preferences, biases, and other factors. This makes it difficult to control and obtain consistent results, especially when evaluating complex or subtle aspects of sound quality. Consequently, many tests may be required, which can be costly. On the other hand, objective evaluation has the advantage of being quantified and standardized, allowing for more reliable and reproducible results. Second, objective evaluation is often more efficient and cost-effective than subjective evaluation because it can be automated and performed quickly using specialized software. As a result, it makes evaluating large data sets more manageable. It also facilitates comparisons between configurations or models on the same data set. We will describe here a few evaluation strategies for both the synthesis quality and timbral control.

2.3.1 Quality Evaluation

Mean Opinion Score

The Mean Opinion Score (MOS) is a widely used method for evaluating the subjective quality of audio signals. It is commonly used to evaluate the perceptual quality of various audio processing systems, such as speech coding, music compression, and sound synthesis.

To obtain the MOS value, a group of human listeners is asked to evaluate the quality of the audio samples on a numerical scale, typically ranging from 1 to 5 [ITU, 1996], where 1 represents the lowest grade, and 5 represents the highest. The scores obtained from each listener are then averaged to obtain the final MOS value for the audio samples.

However, a sufficient number of listeners should be included in the listening test to obtain a reliable MOS value, typically ranging from 20 to 50 or more, depending on the study's requirements. Moreover, setting up a MOS experiment is costly in time and money. Therefore, careful consideration is required when designing and conducting MOS experiments to ensure they are well-controlled and provide meaningful results. It should also be noted that the listening conditions should be controlled to provide the least bias due to the reproduction system.

Overall, the MOS is the judge of peace regarding quality evaluation. Still, its cost will hinder testing multiple models or comparing several solutions efficiently, which saw the rise of neural network-based evaluation metrics.

Inception Score for Objective Generative Model Evaluation

The Inception Score (IS) [Salimans et al., 2016] measures the quality of generated images from a generative model, such as a generative adversarial network (GAN). It is calculated by measuring the marginal likelihood of the generated images and the conditional likelihood of the generated pictures given the classes predicted by the Inception network.

The Inception network is a deep convolutional neural network trained on ImageNet [Deng et al., 2009], a large dataset of real-world images, and can extract a rich set of features from images. We can also use the Inception Network to predict the classes of images, such as objects and scenes.

We can calculate the IS as follows:

$$IS = \exp(\mathbb{E}_x[D_{KL}(p(y|x) \parallel p(y))]) \tag{2.17}$$

where x is a generated image, y is the predicted class of the image, and $p(y|x)$ and $p(y)$ are the conditional and marginal likelihoods of the image and predicted class, respectively.

The Inception Score is a metric that can also be applied to audio data, but it requires the Inception model to be retrained specifically for this purpose. This was demonstrated in [Nistal et al., 2021], where they used Mel-Spectrograms instead of images to distinguish between different types of drums and cymbals and then reused the classifier features to calculate the IS. Another example of the Inception Score used for audio is in [Engel et al., 2019]. However, the authors did not use the Inception model directly. Instead, they applied a similar calculation on the features of a pitch classifier trained on the NSynth dataset.

However, Barratt et al. showed the limits of the IS in [Barratt and Sharma, 2018]. As stated in that paper, the Inception Score shows excellent sensitivity to small changes in weight, even if these changes do not make the final classifier accuracy change. Furthermore, the Inception network used for the score calculation was trained on one data set (ImageNet). Being trained on one massive but specific dataset makes its extension to other datasets nontrivial. The main problem is the possible misalignment of classes between the dataset to test and ImageNet. In [Barratt and Sharma, 2018], the authors showed that the classification of examples from CIFAR-10 does not yield coherent results, invalidating the hypothesis of good classification of the examples. Another point is that the Inception Score is not robust against overfitting, which makes it even more necessary not to use it as a holistic metric. Although these problems were only demonstrated on images, it is reasonable to assume that similar outcomes could be observed on spectrograms and audio data.

Fréchet Distance for Objective Generative Model Evaluation

The objective of this section is to provide an explanation of the Fréchet distance and its application in the evaluation of generative models.

Fréchet Distance between probability distributions The Fréchet Distance [Fréchet, 1957] is a distance between probability distributions. The intuition behind the works is the concept of global distance between two distributions, especially the distance defined as the minimum among all global distances. The squared deviation defines the global distance in the Fréchet distance. For two distributions X and Y, the Fréchet distance is defined formally as:

$$d_F(X,Y) = \min_{\alpha,\beta} \mathbb{E}[\|\alpha - \beta\|^2] \tag{2.18}$$

where α and β have distribution X and Y.

We can also define this over the set of couplings H of X and Y. In that case, $H(X,Y)$ is a joint probability measure whose marginals are X and Y, respectively.

$$d_F(X,Y) = \min_{H} \mathbb{E}_{(\alpha,\beta)\sim H}\|\alpha - \beta\|^2 \tag{2.19}$$

which can finally be written in integral form as :

$$d_F(X,Y) = \min_{H} \int \|\alpha - \beta\|^2 dH(\alpha,\beta) \tag{2.20}$$

In other words, it is the 2-Wasserstein distance on $\mathbb{R}^n$ [Vaserstein, 1969].

A special case for normal distributions has been derived in [Dowson and Landau, 1982]. For two multivariate Gaussian distributions with means μ_X and μ_Y and covariance matrices Σ_X and Σ_Y, the Fréchet distance between these distributions is:

$$d^2 = |\mu_X - \mu_Y|^2 + \mathrm{tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2}). \tag{2.21}$$

Application to the evaluation of Generative Models Fréchet Inception Distance (FID) [Heusel et al., 2017] measures the distance between the distributions of two sets of images. It is commonly used to evaluate the performance of generative models, such as generative adversarial networks (GANs).

Mathematically, FID is defined as the Fréchet distance between the feature distributions of two sets of images, as measured by the Inception network. The feature distributions are the distributions of the activations of the Inception network when applied to the two sets of images. The Fréchet Inception Distance assumes the distributions being measured are Gaussian, since the Gaussian distribution is the maximum entropy distribution for a given mean and covariance.

Under these assumptions and taking into account eq. (2.21), the FID is calculated as follows:

$$FID = ||\mu_r - \mu_g||_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{2.22}$$

Where μ_r and μ_g are the means of the feature distributions of the authentic and generated

images, respectively, and Σ_r and Σ_g are the covariances of the feature distributions of the authentic and generated images, respectively.

The audio equivalent of the FID is the Fréchet Audio Distance [Kilgour et al., 2019] (FAD), which measures the distance between the distributions of two sets of audio samples. Its goals are similar to the FID, but it is used as a metric of synthetic audio against real-world audio.

The FAD calculation uses the embedding from a VGGish model [Hershey et al., 2017] instead of the Inception network. VGGish is a CNN-based classifier designed for audio, based on the VGG architecture originally developed for image recognition [Simonyan and Zisserman, 2015]. It has been trained on a large dataset of YouTube videos, similar to the YouTube-8M dataset [Abu-El-Haija et al., 2016], and can recognize over 3000 audio classes. No information on the different classes was given in [Kilgour et al., 2019], but it is reasonable to assume it is a subset of [Abu-El-Haija et al., 2016]. The activations obtained from the 128-dimensional layer directly preceding the final classification layer serve as the embeddings. The input to the VGGish model consists of 96 consecutive frames of 64-dimensional log-mel features extracted from the magnitude spectrogram of 1 second of audio. The only difference from the FID is the network used to get the embeddings. The underlying statistical process remains the same: the assumption of Gaussian distribution of the embeddings and the same computation of the distance.

It also should be noted that the FAD was developed and described as a metric to measure distortions to evaluate music enhancement models. While its use as an objective empirical metric for evaluating generative models has already been seen in the literature [Nistal et al., 2020; Rouard and Hadjeres, 2021], it should be noted that there is no experimental proof of the correlation between the FAD and qualitative metrics such as Mean Opinion Score for generation evaluation since the FAD was built as a distortion measurement metric. We can, however, expect interpretable results on how well two different models perform on the same dataset. A further subjective evaluation must be performed to obtain a quality evaluation measurement.

Handling of Mode Collapse

Mode collapse in a GAN occurs when the generator produces a limited variety of outputs that do not fully represent the diversity of the training data. This happens when the generator learns to map multiple input samples to the same output, resulting in the loss of information about the true distribution of the data. The generator gets stuck in a local minimum and fails to explore the full possibilities the training data offers. This can happen when the discriminator becomes too powerful and quickly identifies the generator's outputs as fake, leading the generator to converge to a limited set of samples that can fool the discriminator.

Subjective evaluation can only provide a limited view of the audio generated and may not fully capture the extent of the mode collapse. This is where objective metrics can help to quantify the diversity and quality of the generated audio, in addition to subjective evaluation.

The Inception Score was designed with mode collapse in mind, as the aim in [Salimans et al., 2016] was to create a metric that would detect the quality and the diversity.

However, as stated in [Barratt and Sharma, 2018], the Inception Score can be fooled by networks capable of outputting neither.

Regarding the FID, it measures the distance between the distribution of real images and the distribution of generated images in a feature space, calculated using a pre-trained Inception network. The FID is more robust to mode collapse than the Inception Score while still considering the quality and diversity of generated images. Indeed, in the case of mode collapse, where the generator produces a limited variety of images, the FID score would be high because the generated distribution is far from the real distribution, indicating the poor performance of the generator. Therefore, the FID can be a useful metric for detecting mode collapse in GAN-generated images.

2.3.2 Timbre Control Evaluation

Objective evaluation can be utilized to assess the coherence of the timbral features used as input to the model. However, ensuring that any modification in these features correspondingly impacts the model's output is equally important.

To address this, [Ramires et al., 2020] introduced an ordering criterion, emphasizing the need to objectively evaluate the coherence of the timbral features used as input to the model. This criterion was then used by [Nistal et al., 2020] in their effort of timbral control for neural drum synthesizer. The main idea behind this criterion is to ensure the model behaves correctly by confirming that any change in the timbral features used as input leads to a corresponding change in the model's output. Ramires et al. aimed to ensure the coherence of timbral features used as input to the model by varying individual features while holding the others constant. The accuracy of the generated output was assessed via the same features used for training. It should be noted that Ramires et al. used a min/max scaling to force the features to be between 0 and 1 on their dataset. For each individual feature, three values were set: low (0.2 on the normalized scale), mid (0.5), and high (0.8). The resulting outputs were $\hat{x}^i_{low}$, $\hat{x}^i_{mid}$, and $\hat{x}^i_{high}$, and their corresponding features were fs^i_{low}, fs^i_{mid}, and fs^i_{high} for the i-th feature. The hypothesis was that for coherent modeling, the models must follow the order $fs^i_{low} < fs^i_{mid} < fs^i_{high}$. The ordering accuracy was tested three times, with E1 checking the condition $fs^i_{low} < fs^i_{high}$, E2 checking $fs^i_{mid} < fs^i_{high}$, and E3 checking $fs^i_{low} < fs^i_{mid}$ over all values of i. Although this metric gives a general idea of how well the system functions, it does not provide any information about how closely the generated samples match the intended input.

There is currently no state-of-the-art method for subjectively evaluating timbre control in neural synthesis. To evaluate timbre control subjectively, we can consider two perspectives. The first is whether the synthesis quality deteriorates with more control over the process. This raises the question whether increasing the number of control parameters in a synthesizer can lead to an undesirable output, an essential consideration when designing a synthesizer that allows for control of individual timbral features. The second perspective is determining how noticeable control errors are to human listeners. In other words, if the control parameters used to synthesize a sound are inaccurate, to what extent can a listener identify these inaccuracies in the final output?

These perspectives are critical in understanding the limitations and opportunities of neural synthesis, and they can guide the development of more effective models that meet the

needs of both creators and listeners.

Although there are no examples of subjective evaluation for timbre controls for the two main perspectives described above, it is possible to reuse the methods discussed in Sect. 1.6. More precisely, psychophysical experiments can be performed to determine the extent of control errors that human listeners can detect and conclude on the capabilities of the tested control method.

2.4 Conclusion

This short state-of-the-art review shows the lack of high-quality control for neural drum synthesizers with a common compromise between high-quality and precise timbral control before 2019, as shown in Tabl. 2.1. However, new methods were found quickly after the beginning of the thesis, in early 2020 and beyond, moving from evolved GAN models [Nistal et al., 2020; Drysdale et al., 2020] to diffusion models [Rouard and Hadjeres, 2021] as shown in Tabl. 2.2. It also reviews the most common objective metrics for measuring audio generation quality. It allows comparing networks trained on the same dataset while considering the limitations compared to subjective evaluation.

Reference	Sample Rate	Duration	Timbral Control
WaveGAN [Donahue et al., 2019]	16kHz	1.1s	✗
NDM [Aouameur et al., 2019]	16kHz	1.3s	✗
NeuroDrum [Ramires et al., 2020]	16kHz	1s	✓

Table 2.1: Comparison of state-of-the-art neural drum synthesizers before the start of the thesis (before the end of 2019)

Reference	Sample Rate	Duration	Timbral Controls
DrumGAN [Nistal et al., 2020]	16kHz	1.1s	✓
Adversarial Audio[Drysdale et al., 2020]	44.1kHz	0.4s	✗
CRASH [Rouard and Hadjeres, 2021]	44.1kHz	0.4s	✗
DrumGAN VST [Nistal et al., 2022]	44.1kHz	0.5s	✓

Table 2.2: Comparison of state-of-the-art neural drum synthesizers published during the thesis (from 2020 onwards)

At the start of the project (i.e., end of 2019), the only available solutions were [Donahue et al., 2019] [Aouameur et al., 2019] and [Ramires et al., 2020], as shown in Tabl. 2.1. In this context, we investigated GANs, more precisely StyleGAN, to generate drum sounds. The novel (at that time) style-based approach appeared very convincing for image generation and basic control over the generation. We can also leverage the fact GANs use discriminators instead of direct reconstruction losses. Combining a discriminator with "handicapped" generators should allow us to obtain new interesting sounds.

More recent research encourages the usage of diffusion-based methods for high-quality images and even drum sounds with interesting inpainting capabilities. This was enough

of a breakthrough to question the validity of a GAN-based approach, especially in the middle of a thesis.

The first argument against a change in direction is the overall inference time. Diffusion models have a high computational cost that makes them unusable on consumer-grade computers letting GAN-based approaches a head start upon diffusion models. As this project aims at creating a synthesizer with acceptable inference performances, using GANs appears to be a great compromise between quality and inference speed as VAEs do not provide the highest quality available in the state-of-the-art.

Another issue with considering the new diffusion methods is that they would require a complete redesign of our synthesizer and, as such, more time spent rebuilding a system. Given the timeline of the project and the publication of [Rouard and Hadjeres, 2021] at the end of 2021, it would have been too short to consider, as it would have taken at least a year to change direction completely.

For network evaluation, we have two main evaluation points and two methods: sound quality and control and objective and subjective methods. In terms of sound quality and even if the Fréchet Audio Distance (FAD) has already been used in state-of-the-art literature for objective quality evaluation, one should remember that it was not engineered with GAN evaluation in mind, which in turn implies we have to use subjective evaluation as a final metric.

Regarding control evaluation, the state-of-the-art provides an ordering criterion with clear limitations. It should then be noted that no subjective evaluation has been conducted on this subject. We can also add the evaluation metric used for the control evaluation case and introduced in [Ramires et al., 2020] is weak as it is only an ordering criterion. In addition, the results with this ordering metric are far from perfect, as we will see later in this document in Table 4.8. This objective evaluation of timbral control shows clear limitations. Instead of using the raw values of the descriptors, using the descriptors as differentiable functions should yield better results, especially in light of the strong results obtained with DDSP.

From this, we can say that generative adversarial networks are a great compromise to mix high-quality generation, precise and coherent high-level controls, and good inference time performance, even if newer propositions can supersede them on certain points. Given the objectives listed in Sect. 1.1, they appear to be the best compromise for the present study.

Chapter 3

Databases

3.1 Introduction

As our drum synthesis method is data-driven, exploring the data used for the training and inference is necessary. We will especially describe the two datasets we used in our experiments and how we built them.

3.2 ENST Drums

The ENST-Drums database [Gillet and Richard, 2006] is a large and somewhat varied database aimed at research for automatic drum transcription and general drum audio processing.

This dataset was built by recording three professional drummers specializing in different music genres. Each drummer provided about 75 minutes of audio recordings. As these drummers come from different musical backgrounds, they all played drumkits with the toolset of their respective genres: sticks, rods, brushes, or mallets. The drum kits recorded range from small jazz-oriented drum kits to larger rock drum sets.

In addition to 8 individual audio channels, the performances were filmed from two angles, providing further data to process for tasks such as performance analysis.

3.2.1 Subset of Interest

Some parts of ENST-Drums are not helpful for our application to drum synthesis. First and foremost, the videos of the performances and the recordings of these performances are not interesting for our target application. Since we aim to generate one-shot samples, we will select samples without cross-talk, i.e., no full performances and only single hits using close mics only.

Regarding cymbals, the data set provides no close miking except for the hi-hat. Moreover, the cymbals are recorded using a stereo pair of overhead microphones. The overhead

microphones are positioned at a considerable distance from the components of the drum kit (around 1m), which means that they do not satisfy the close-miking condition we established for constructing our dataset.

Regarding drums, close mics are available for all instruments: kick drum, snare drum, and toms. These monophonic recordings represent a drum kit's expected sound when recorded. However, not all microphones are helpful at all times. For instance, the kick drum microphone can pick up the snare when the snare drum is played, and inversely. To avoid this problem, we only keep recording the related close mic for each instrument. In other words, the kick drum samples come from the kick drum microphone only, as do the other drums and their specific, close microphones. The subset we will use will comprise close-miking of kick, snare, tom, and hi-hat and the relative population of the dataset is given in Tabl. 3.1.

Element	Proportion
Kick	4%
Snare	18%
Toms	45%
Closed hi-hat	11%
Open hi-hat	22%

Table 3.1: Dataset population

3.2.2 Shortcomings

The first and obvious shortcoming with our subset is the small size of the overall data set, and even more so for the subset. While the problem might not be a problem for classification or transcription using non-machine-learning-based approaches, our generation task with neural networks will need more data to avoid overfitting on a subset lacking variations. Given [Nistal et al., 2020] successfully trained a GAN for drum synthesis with a data set made of 200k samples, we aim for a data set of approximately the same size.

3.3 Data augmentation

As the selected subset of ENST-Drums comprises 350 samples, we must augment the subset to avoid overfitting and encourage better generation variability. For this, we reused the proposed method of [Jacques and Roebel, 2018] initially aimed at drum transcription.

We used `SuperVP` [1] to process the ENST-Drums subset. The modifications to the sounds consist of a gain applied to transient/attack components [Röbel, 2003], noise components, and independent transposition of the signal source and the spectral envelope. Table 3.2 shows the set of parameters. The limits have been obtained using informal subjective evaluation of the modified sounds to avoid transformations that can be perceived as unnatural

[1] SuperVP is available free of charge in the form of a Max/MSP object at https://forum.ircam.fr/projects/detail/supervp-for-max/

by a human listener. A more thorough evaluation of the extreme values was conducted as part of our neural synthesizer's subjective sound quality evaluation in Sect. 4.2.5. Counting the different combinations of parameters amounts to 360 possible combinations, which increases the number of samples available for training from 350 to approximately 120k.

Process	Parameters
Remix attack	0.1, 0.3, 0.6, 1.5, 2, 3
Remix noise	0.6, 1.5, 2, 3
Transposition	0, ±100, ±200
Spectral envelope transposition	0, ±200

Table 3.2: Augmentation operations and parameters

3.4 Apeira Drums

In the following section, we will introduce the content of our new dataset of drum sounds called Apeira-Drums. This dataset contains high-quality (48kHz-24bits) recordings of drum sounds using several microphones and drums. We also provide information on drum tuning and cymbal choices. The recording protocol was documented in case further recordings need to be added with other drums, cymbals, or drummers. Finally, we also describe the hit velocity measurements done during recording sessions. These measurements aim to give a ground truth for MIDI velocity estimation using either state-of-the-art piezoelectric sensors or special accelerometer units.

3.4.1 Structure of the dataset

The dataset consists of drum sounds and measurements sorted into different categories:

- Close miking
- Overheads
- Ambiance microphones
- Microphone crosstalk
- Accelerometer data
- Trigger info

The "close miking" samples are the raw drum sounds recorded by the microphones of interest over the drum. The "Accelerometer data" and "Trigger info" are either processed data from a Redison Senstroke smart drum stick sensor (containing an accelerometer, full sensor shown in Fig. 3.1) or the raw signal recorded through piezoelectric triggers, which are in contact with the drum head at the moment of impact. "Overheads" samples are comprised of a pair of stereo overheads. These are not deemed as critical as the

close miking samples but can be added to taste by the dataset end-user. "Ambiance microphones" are for room microphone recordings, using premium recording gear as part of the signal path for added color and creative processing. "Microphone crosstalk" refers to all the audio picked up by other microphones than the one intended to capture the sound source of interest.

Figure 3.1: Redison Senstroke sensor mounted on a Vater 5A drumstick.

3.4.2 Recording Setup

The dataset is composed of drum sounds recorded at 48kHz-24bit, a step-up when compared to ENST-Drums [Gillet and Richard, 2006], IDMT-Drums [Dittmar and Gärtner, 2014], and FreeSound [Font et al., 2013]. The raw recordings were cut using the transient detection algorithm from [Röbel, 2003]. All the running microphones went through Yamaha SB16-ES preamplifiers, except when stated otherwise. We used premium preamplifiers on ambiance recordings to color the recordings with the euphonic characteristics of these premium preamplifiers. This was chosen during the recording process during informal A/B testing.

All preamplifiers are bound to a digital Dante network through an Ethersound-Dante-AES/EBU interface. The recording was made using Cubase 10.5 at a native sample rate of 48kHz and a resolution of 24 bits. All recordings were made at L'Ampli. Studio 3 in Le Creusot, France. Fig. 3.2 shows the recording setup and layout.

3.4.3 Dataset Contents

We selected two Yamaha Stage Custom drum kits and three snare drums to ensure maximum variety. There are five crash cymbals, one ride, one pair of hi-hats, and two effect

Figure 3.2: Photography of the recording setup (Yamaha Stage Custom)

cymbals. All the drums and cymbals are recorded using a pair of Vater 5A drumsticks except for the kick drum, where a felt beater and a wooden beater were used. The supplementary material accompanying the dataset shows the complete list of drums and cymbals. The dataset amounts to 1492 drum and cymbal close mic sounds. We also add the stereo overheads and the variety of room recordings.

Figures Fig. 3.3a and 3.3b show the mean envelopes computed on the dataset samples. These envelopes were generated using the close mics sounds from the dataset, with one envelope per type of drum or cymbal. For each sample of one given type, the final envelope is the filtered mean of the analytical part of the Hilbert transform of these peak-normalized samples. We defined the peak normalization for a signal x as :

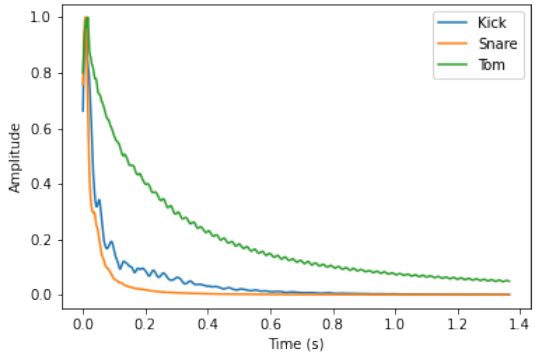
$$\hat{x} = \frac{x}{\max(|x|)} \tag{3.1}$$

The Hilbert transform is calculated using the Discrete Fourier Transform on the dataset's first 65636 samples of each normalized sound. This length is the same length used in [Lavault et al., 2022b]. However, all the sounds in the dataset are longer. We can see that 1.4s is too short for the cymbals to decay fully.

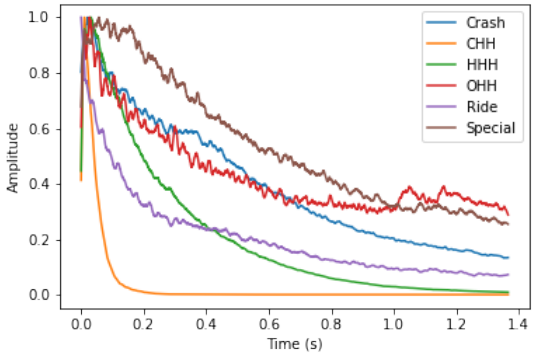
3.4.4 Trigger and accelerometer data

The accelerometer data from the Senstroke sensor needs some explanation. Note that the output of the accelerometer is included in the dataset.

The raw output of the sensor looks something similar to:



(a) Mean normalized envelopes for shell drums



(b) Mean normalized envelopes for cymbals

Figure 3.3: Mean normalized envelopes on the dataset, shortened to 1.4s

```
21580; 0.969...; 0.02...; 0.24...; 0.02...; 0.0; 0.0; 0.0; 30.89; 255
```

Our most interesting data lies in the second to last column, which gives an estimated MIDI velocity whose estimation will be described in the next paragraph. The rest represent a timestamp, an estimate of the position, and the magnetometer output. Please note that the position algorithm was not initialized in our experiments. This does not influence the MIDI velocity measurement, as these two measurements are separated.

Apeira Technologies, which funded this thesis, developed the Senstroke signal processing program. As a result, we have detailed knowledge about the sensor and its processing. The MIDI velocity estimation in the Senstroke relies on the angular speed combined with an impact detection algorithm that we cannot disclose in this context. If we denote ω the angular speed at the moment of impact, the estimated velocity V is given by :

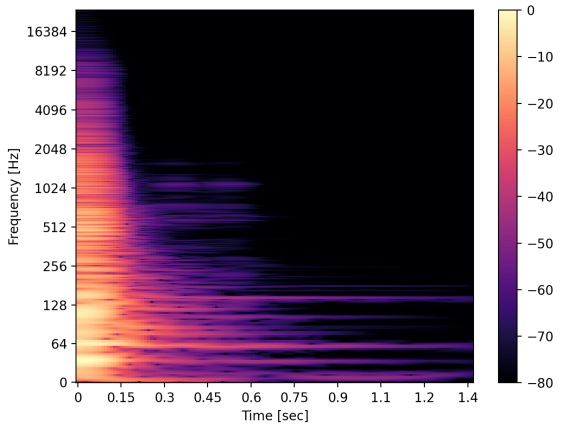
$$V = 5.3 \times \omega \tag{3.2}$$

It represents an estimation of the MIDI velocity on a commercial electronic drum kit. The scaling factor was defined to match the MIDI velocity output of a state-of-the-art electronic drum kit. Contrary to the MIDI standard, this value is neither an integer nor bounded by 127.

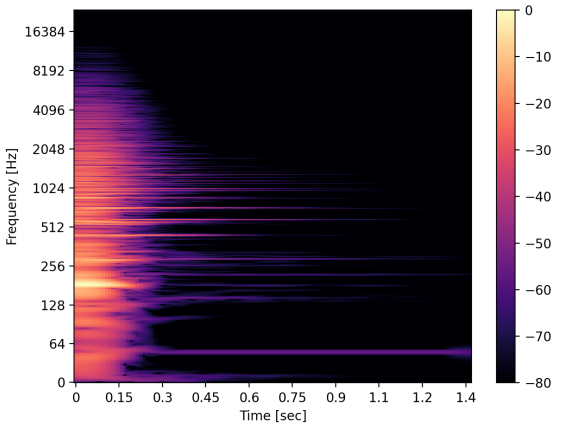
The *triggers* are standard in live audio and electronic drum kits. They are generally used to trigger sounds using specialized sound modules, hence the name. To work effectively, these sensors are placed in contact with the drum head, in general, by holding on to the drum's rim. The idea of recording trigger signals in Apeira Drum is to have a reference signal independent of the microphone. While mostly useful for automatically segmenting raw audio files, they are also expected to provide a velocity estimation since they are used in state-of-the-art MIDI velocity-compatible drum modules.

Spectrograms of such signals are shown in Fig. 3.4. The main difference with microphone signals is the weaker decay at higher frequencies (500 to 5000 Hz) and the lack of prominence of lower frequencies ($\leq$ 100 Hz). The first point can be explained by the contact between the head and the sensor making higher order modes prominent on it but not necessarily in the air: it can be further illustrated by comparing Fig. 1.2a and 3.4c. The diminution of lower frequencies is due to the limited physical capabilities of the sensor to handle them at a mechanical level.

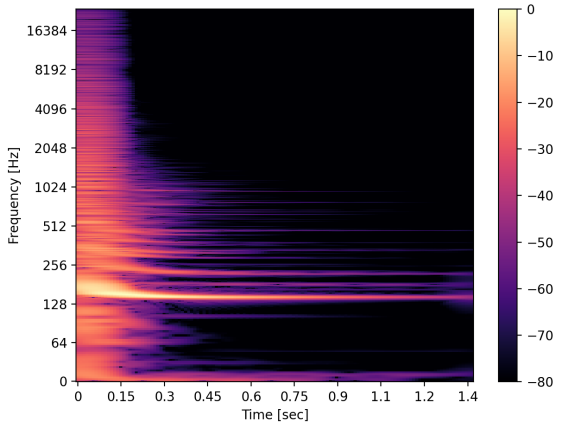
We also noticed that at very high velocities (*f* and above), the trigger signals present traces of distortion. As these do not come from the recording setup, we assume the sensor can saturate when excited with enough force. This can be illustrated in Fig. 3.4d, where the plot above shows the signal from a trigger at *pp* velocity, i.e., the lowest available, compared to a trigger signal at velocity *fff*, i.e., the highest available. The trigger distortion at high velocities can be seen in Fig. 3.4c and 3.4d by the presence of frequencies above 8000Hz for a short time near the transient. For clipped edges on the waveform, as shown in Fig. 3.4d: the hard clipping induced by the sensor saturating creates higher frequencies that are not representative of the movement of the drum head.



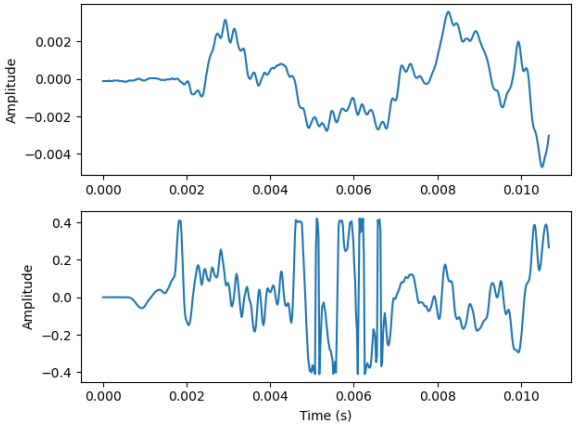
(a) Trigger signal for the kick drum (Yamaha 22")



(b) Trigger signal for the snare drum (Yamaha Snare)



(c) Trigger signal for the toms (Yamaha 10" tom)



(d) Example of clipping in trigger signals. The plot above shows a trigger signal at a lower excitation level (*pp*), and the plot below shows a trigger signal at a high excitation level (*fff*) on the same snare drum.

Figure 3.4: Examples of trigger signal spectrograms on real drums (Apeira Drums)

3.4.5 Recording Protocol

The recorded drummer was instructed to play eight hits for the drums and 6 hits for the cymbals at 8 different dynamics levels, going from *pianississimo (ppp)* to *fortississimo (fff)*. The dataset we describe here contains four dynamic levels out of the eight. These

4 levels are *pianissimo (pp)*, *mezzo-piano (mp)*, *forte (f)* and *fortississimo (fff)*. This is done for convenience, to help with smaller proof-of-concept experiments while remaining larger than IDMT-Drums or ENST-Drums. Each set of hits in our recordings consists of drum or cymbal strikes with the same dynamics level without any variation in technique or crescendo effect.

Different techniques were recorded whenever it made sense, like hitting the hi-hat with different opening levels or parts of the stick.

3.4.6 Repeatability

To ensure the recording session is as repeatable as possible, we compiled all the drum tunings used during the recording of this dataset. We used a calibrated DrumDial for the measurements. Even for expert musicians and engineers, drum tuning is challenging [Toulson and Hardin, 2020]. Using a calibrated tensiometer allows for a repeatable recording setup, e.g., using the dataset as a ground truth against other drummers in different locations. This allows us to be able to repeat the recordings of the drums used in our session, with another drummer or in another studio to do one of the following:

- Recording of the same drums with different drummers,
- Effect of tuning on a given drum and head combination,
- Effect of microphone placement for a given tuning combination.

3.4.7 Discussion

We have to concede that Apeira Drums is not as massive in terms of the number of samples as the one used by Sony in [Nistal et al., 2020; Rouard and Hadjeres, 2021]. We considered applying the same augmentation technique we used for our augmented ENST-Drums dataset but with reduced limits to minimize artifacts in the Apeira Drums dataset. However, due to time constraints, we could not do so as the development of Apeira Drums was completed towards the end of the project. This will limit the use of the dataset to velocity study in Sect. 4.5. In addition, we plan to release the subset described here to the community with all the sounds and sensor data and a commercial version based on the whole dataset aimed at music producers.

3.5 Dataset used in the document

In this section, we describe the different contents and abbreviations we will use in the following chapters :

ENST-OG: Subset of ENST Drums containing all closed-miked drums and hi-hat. This data set represents real drum sounds. It is used in our perceptual evaluation of

sound quality. This data set has all its elements at a 44.1kHz sample rate. The samples have varying lengths, but all have a duration above 1 second.

ENST-AUG: Augmented version of ENST-OG, as described in [Lavault et al., 2022b]. Note that the augmentation process preserves the labels. It is used as a training data set for our neural drum synthesizer. This data set has the same sample rate as ENST-OG, and the length of the sounds is unchanged.

ENST-AUG-EX: Subset of ENST-AUG containing only extreme examples of augmentation. It is used to evaluate sound quality to evaluate the perceptual coherence of the augmented training data (ENST-AUG) compared to the original drum samples in (ENST-OG)—same sample rate and duration since it is a subset of ENST-AUG.

SWG-SQ (StyleWaveGAN Sound Quality): Samples generated with StyleWaveGAN trained on ENST-AUG without descriptors and with random latent for all labels in ENST-AUG. This set is only used in the sound quality evaluation. These sounds have a duration of 1.5s at a 44.1kHz sample rate.

DG-SQ: Samples for kick, snare, and cymbals labels provided by Javier Nistal generated with a version of DrumGAN providing drum type conditioning according to [Nistal, 2022]. These were trained on the private data set used in [Nistal et al., 2020]. This set is used in the sound quality evaluation as the state-of-the-art reference. The elements from this data set have a duration of 1.1s and a sample rate of 16kHz.

SWG-CQ (StyleWaveGAN Control Quality): Snare samples generated with StyleWaveGAN trained on ENST-AUG using descriptor controls. This data set is used for the control quality evaluation. Only snare samples are used to remain consistent with the objective evaluation performed in [Lavault et al., 2022b]. The sounds from this data set have the same sample rate and duration as SWG-SQ.

AD: Version of Apeira Drums described previously, with all drums and cymbals.

AD-SStroke: Subset of Apeira Drums aligning cymbals close mic recordings and Senstroke velocity estimation

Chapter 4

Drum Synthesis with Adversarial networks

4.1 Introduction

After an overview of the state of the research field, this chapter will present the main contribution of this thesis: StyleWaveGAN. StyleWaveGAN is a neural drum synthesizer with drum instrument conditioning and additional timbral and velocity controls.

We will separate this chapter into four main sections. First, Section 4.2 will introduce StyleWaveGAN and compare results obtained with StyleWaveGAN against results obtained with other neural synthesizers from the state-of-the-art. This first comparison will notably evaluate the audio quality of the generated sound. In Section 4.3, we will present the result of the proposed method for timbral control using differentiable timbral descriptors. In the same section, we will also describe the result of a psychophysical experiment to obtain information on the perception of control error, i.e., answering the question, "Is the control error perceivable?". In Section 4.4, we will discuss a hybrid synthesis method that combines an oscillator bank with StyleWaveGAN to mitigate the shortcomings of the original version. Finally, we will show how we built a velocity descriptor based on the signal energy and how we can use this descriptor as velocity control in Sect. 4.5.

Note that the content presented in Sect. 4.2 and 4.3 of this chapter are extensions of the work previously published [Lavault et al., 2022b,a,c], such that there is significant overlap between them. The other sections presented here are unpublished works.

4.2 StyleWaveGAN (Basic structure)

4.2.1 Introduction

Our first contribution to the neural drum synthesis field was StyleWaveGAN [Lavault et al., 2022b]. Based on a StyleGAN-like architecture [Karras et al., 2019][Karras et al., 2020], StyleWaveGAN is a modification made to output waveforms instead of images.

Trained on the augmented subset of ENST-Drums described previously (ENST-AUG), StyleWaveGAN can generate drum sounds of kick, snare, toms, and hi-hat (closed or open) while offering better performance than the state-of-the-art for objective and subjective evaluation metrics.

This section will first detail the structure of StyleWaveGAN and how it differs from StyleGAN [Karras et al., 2019, 2020]. We will then present results using two evaluation methods, one involving an objective measurement with the Fréchet Audio Distance [Kilgour et al., 2019](FAD), already described in Sect. 2.3.1, and the other involving human listeners for subjective quality assessment.

4.2.2 Structure

This section will describe how StyleWaveGAN descends from StyleGAN and how it is adapted to work for direct waveform generation.

Generative Adversarial Networks and StyleGAN

We have seen in Chapter 2 that Generative Adversarial Networks (GAN) describe a family of training procedures in which a generative model (the generator) competes against a discriminative adversary (the discriminator) that learns to distinguish whether a sample is real or fake [Goodfellow et al., 2014].

Instead of a vanilla GAN, we use an evolution called StyleGAN. [Karras et al., 2019, 2020]. StyleGAN attempts to mitigate the entangled representation when using noise as the generator's latent input. The key idea here is to use a *style encoding*, a vector obtained through a mapping network and then used to control (through an affine transform) every layer of a synthesis network. The affine transform from the mapping network output to a style vector can be expressed as:

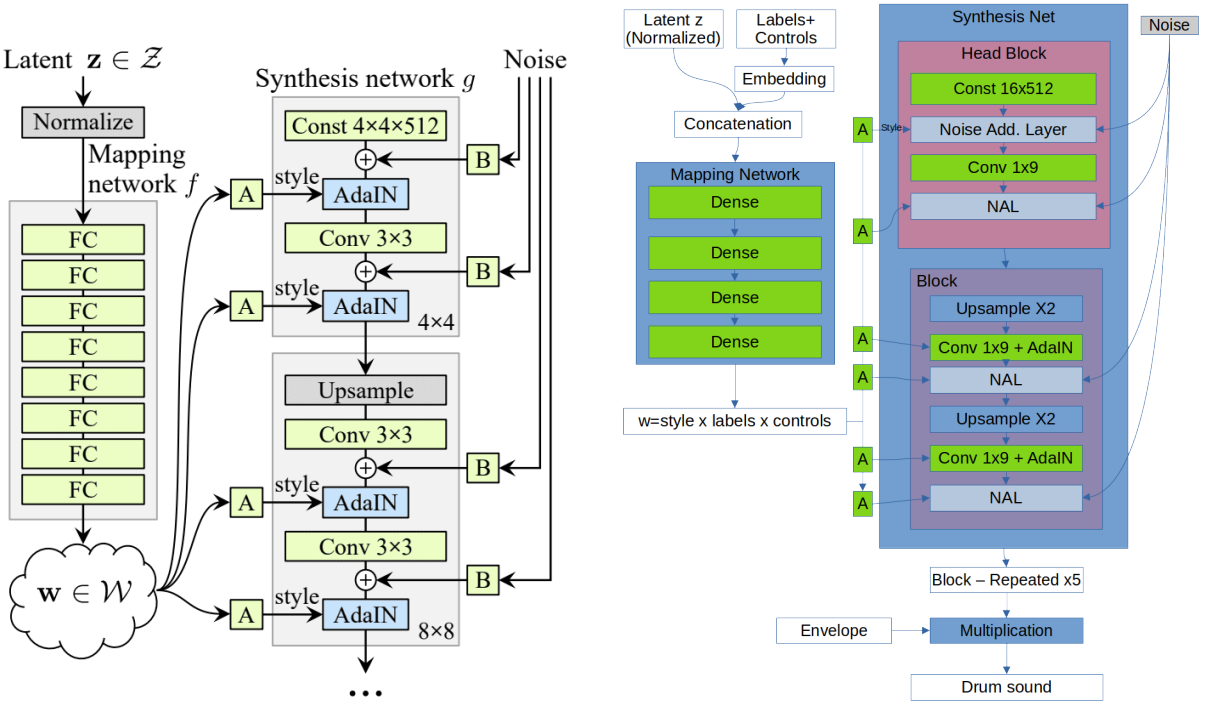
$$y = A \times w + b \tag{4.1}$$

where y is the style vector, A is the learned matrix corresponding to the linear part of the transform, w is the intermediary representation from the mapping network, and b is a bias term to make this transform affine. The "A" blocks in Fig. 4.1a and Fig. 4.1b correspond to eq. (4.1), where each block has a different set of weights.

Proposed Architecture

Since StyleGAN was initially used for high-quality image generation, we must modify it for direct waveform generation. In particular, we transform 2D convolutions (3×3) into 1D causal convolutions (which means we ended up with filters of length 9) [van den Oord et al., 2016], and the upsampling is done with an averaging filter before each convolution block in the synthesis network. Similarly, the learned $4 \times 4 \times 512$ starting tensor in StyleGAN (c.f. Fig. 4.1a) is replaced by an equivalent learned layer of format 16×512. The mapping network has 4 layers instead of 8. We use the same number of filters per layer as StyleGAN2 [Karras et al., 2020]. Like StyleGAN2, the synthesis network uses

input/output skips, and the discriminator is a residual network. The loss function we use is WGAN-LP [Petzka et al., 2018] instead of WGAN-GP [Arjovsky and Bottou, 2017] used in StyleGAN. This loss function follows the principle of the Wasserstein GAN [Gulrajani et al., 2017]. The differences between these losses have been described previously in Sect. 2.2.3. Figure 4.1 shows a side-by-side architecture comparison between StyleGAN and StyleWaveGAN.



(a) StyleGAN, reproduced from [Karras et al., 2019]

(b) StyleWaveGAN (skip connections are not shown for clarity). NAL stands for Noise Addition Layer.

Figure 4.1: Comparison between StyleGAN and StyleWaveGAN. Here, "A" stands for a learned affine transform, and "B" applies learned per-channel scaling factors to the noise input. Since StyleWaveGAN uses style-controlled noise layers, it has no per-channel global scaling like StyleGAN.

In this work, we follow [Ramires et al., 2020; Drysdale et al., 2020] using a temporal signal representation. We know that in [Nistal et al., 2021], the authors found the magnitude+instant frequency and complex STFT (i.e., spectral representations) are the best representations on NSynth with the same network architecture. However, the sounds in NSynth are mostly non-percussive, which differs greatly from drum sounds.

In our case, we conducted informal perceptual evaluations performed in the initial phase of this study to support our idea that the temporal representation produces better audio quality than spectral representation for the same network size: we suppose the preference towards temporal is because of the high amount of noise and the importance of the transient in the drum sounds, which were not as precise with the spectral version of our network.

Noise Addition Layers

StyleGAN introduced explicit noise inputs in the generator network to generate stochastic detail. More precisely, these noise inputs are single-channel images of uncorrelated Gaussian noise. Such noise image is broadcasted to all feature maps using learned per-feature scaling factors (B blocks in Fig. 4.1a) and then added to the output of the corresponding convolution, as shown in Fig. 4.1a. The noise layer in StyleGAN can be described as:

$$B(n) = \mathcal{C}_{i=1}^{C} \lambda_i n \tag{4.2}$$

where B is the layer's output, λ is a learned per-channel scaling factor, n is the noise image, and $\mathcal{C}$ is a concatenation operator on the channels going from 1 to C, with C being the number of channels of the previous convolutional layer's activations.

As stated above, this means the same noise is replicated on every channel but with a different scaling λ_i. In that case, n has format $M \times T \times 1$ where M is the batch size, T is the time duration at the layer, and the layer's output $B(n)$ has format $M \times T \times C$. The layer's output is then summed with the activations of the previous convolution layer. It should be noted that the noise addition in StyleGAN is independent of the style mechanism (A blocks in Fig. 4.1a).

For StyleWaveGAN, however, we modified the noise addition layers B of StyleGAN to make them style-dependent, as described in eq. (4.3). We also added an envelope (a fixed linear fade out) on the noise directly to avoid noisy tails. Controlled noise addition is helpful in our drum synthesis application since some classes need more noise than others to obtain a good-quality synthesis. We can summarize StyleWaveGAN's noise-addition layer with the following equation:

$$NoiseLayer(w, n) = AdaIN(\mathcal{C}_{i=1}^{C} n, y) + b_{NAL} = AdaIN(\mathcal{C}_{i=1}^{C} n, Aw + b) + b_{NAL} \tag{4.3}$$

where $NoiseLayer(w, n)$ is the layer's output, w is the mapped latent, $y = Aw + B$ is the affine transformation from mapped latent w to style y with learned parameters A and b (as described in eq. (4.1)), n is the single-channel white noise which is then replicated on every channel with the concatenation operation $\mathcal{C}$ described above, AdaIN the operation described in eq. (2.14), and finally, b_{NAL} is a bias term.

The fade-out is omitted in the equation for the sake of clarity. The layer's output is then summed to the activations of the previous convolution layer.

Pre-computed Output Envelopes

One of the drawbacks of having noise addition layers is the lack of control of the decay of said noise. Even with a parametric fade-out of the noise learned by the layer, which gave better results in informal testings, the generated sounds have an audible noisy tail, making them easily identifiable by a human listener. We added pre-computed envelopes after the network's output to mitigate this problem.

We obtained these envelopes from the training data, assigning one envelope to each drum or cymbal type. For each sample of one given type, the final envelope is the filtered mean of the analytical part of the Hilbert transform of these normalized samples. A small fade-out is applied to avoid audible clicks at the end of the generated sounds.

Let us consider $\mathcal{H}$ the Hilbert transform, $\mathcal{F}$ a filter (in our case, a Savitzky-Golay filter [Savitzky and Golay, 1964]), and n_c the number of samples in the class c. We obtain the envelope for the peak-normalized sounds of class c, $(s_{i,c})_{1 \leq i \leq n_c}$, by computing :

$$e_c = \mathcal{F}\left(\frac{1}{n_c}\sum_i |\mathcal{H}(s_{i,c})|\right) \tag{4.4}$$

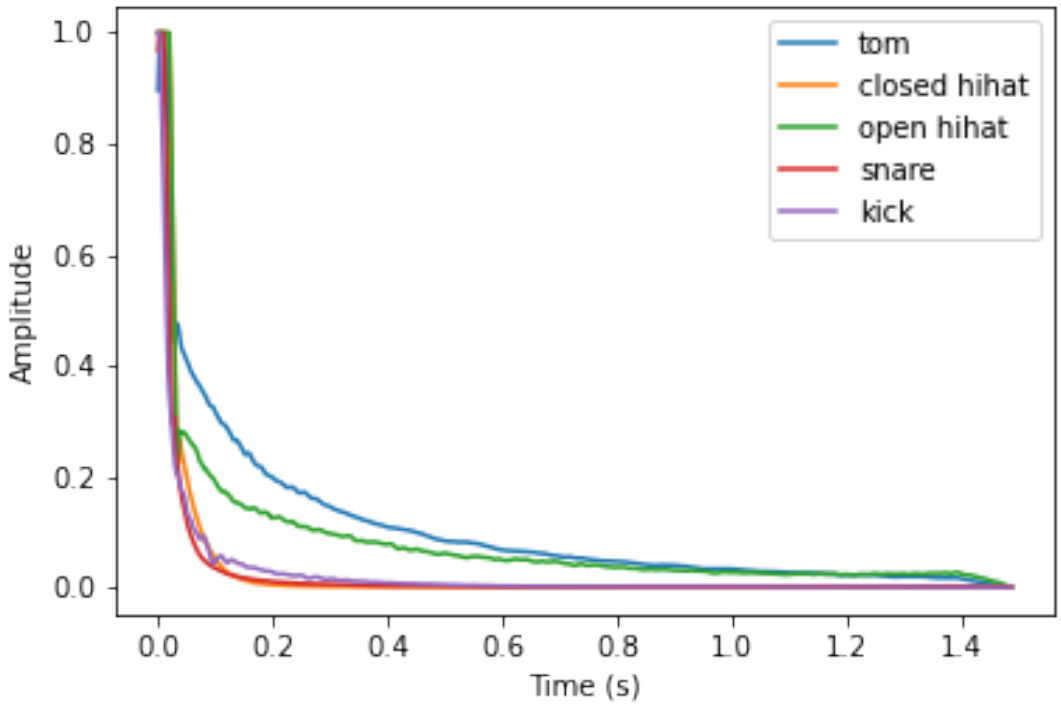


Figure 4.2: Generated envelopes from the training dataset (ENST-Drums)

The final audio is obtained by multiplying the output of the synthesis network and the matching envelope element-wise. If the output is correct, the internal energy needs to be quasi-stationary because the envelope would represent the energy in this case. We also hypothesize that it will help the generator by reducing the dynamic range generated by the non-linearities inside the network.

The output time signal of the synthesis y is obtained from the output x of the last convolutional layer of the said network by multiplying the envelope signal e_c for drum class c defined above as in the following equation:

$$y = x \odot e_c \tag{4.5}$$

where $\odot$ is the Hadamard (i.e., element-wise) product.

Controlling the Network

To match the project prerequisites, our synthesizer should be able to generate specific types of drums according to a class, e.g., generating a snare drum, a kick drum, or a

closed hi-hat. But we also want to introduce a notion of intuitive control of the timbral properties. The timbral control will be detailed in Sect. 4.3. However, the control method is the same for the labels and the timbral features.

In our experiments, we use 5 labels, which are the drum types in ENST-AUG, i.e., kick, snare, tom, closed hi-hat, and open hi-hat. Regarding the implementation, the one-hot encoded labels and audio descriptors (*labels x controls* in Fig. 4.1b) are fed into an embedding layer whose output is concatenated to the latent z and fed to the mapping network. These same labels and descriptors are also concatenated after the mapping network, which can be seen by the *style* $\times$ *labels* $\times$ *controls* block in Fig. 4.1b. Using this method, we expect a better disentanglement between the class label and the descriptors during the style encoding.

AutoFade

Progressive Growing of GANs has been proposed in [Karras et al., 2018] and used in [Drysdale et al., 2020; Nistal et al., 2020]. However, it's worth noting that Progressive Growing was used in [Karras et al., 2018, 2019] but not in [Karras et al., 2020]. Since StyleWaveGAN is based on StyleGAN 2 [Karras et al., 2020], we did not use Progressive Growing. However, in our experiments, we developed and evaluated a variant of Progressive Growing called AutoFade.

It is a ResNet architecture with a convolution path and a bypass, where a learned parameter is used to fade more or less of one path. Rather than fixing a value like ResNet [He et al., 2016], we let the network choose the best value as part of the training process without the need to train it similarly to Progressive GAN. AutoFade especially avoids the checkpoint management and supervised phasing of the layers of Progressive GAN.

Before we proceed with the mathematical description of AutoFade, let us review the fundamental components of a ResNet block [He et al., 2016]:

$$y = \mathcal{F}(x, W) + x \tag{4.6}$$

where x is the input to the block, $\mathcal{F}$ is a nonlinear function that represents the residual mapping to be learned, W represents the weights of the residual mapping, and y is the output of the block. This means $\mathcal{F}$ does not need to encode information already in x, and the gradient flows easier to previous layers, mitigating vanishing gradient problems.

A similar version called the Highway Network [Srivastava et al., 2015] proposed the following variation :

$$y = \mathcal{F}(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T)), \tag{4.7}$$

where x is the input, W_H, and W_T are the learnable parameters for the transformation and gating functions, $\mathcal{F}(x, W_H)$ is the nonlinear transformation function, $T(x, W_T)$ is the gating function that controls the flow of information, and y is the output. The gating function can take various forms, such as a sigmoid function or a rectified linear unit (ReLU) function. The idea behind the highway network is to allow the information to

bypass the transformation function if it is deemed unnecessary or redundant, making it a superset of ResNet.

In the case of AutoFade, this becomes:

$$y = \sin(\alpha)\mathcal{F}(x, W) + \cos(\alpha)x \tag{4.8}$$

It's worth noting that α is independent of x or $\mathcal{F}(x, W)$ in eq. (4.8).

It makes this structure an intermediate between ResNet and Highway Networks. For the latter, it would mean that α would depend on the input x. AutoFade is similar to Progressive Growing in that the α parameter changes over time. However, contrary to Progressive Growing, the parameter is not forced to increase by hyperparameters but changes with the gradient information. The "growing" depends on the data and the training iteration. Since we use trigonometric functions in eq. (4.8), we guarantee the conservation of the standard deviation if both inputs have equal variance.

Regarding our preliminary results with AutoFade, we did not observe any advantages in utilizing progressive growing or AutoFade within the generator, consistent with the findings reported in [Karras et al., 2020]. However, we did notice an improvement when we employed the AutoFade technique for the discriminator. As a result, in the subsequent sections, we will focus solely on examining the effectiveness of the AutoFade feature as part of the discriminator.

4.2.3 Training Setup

The training procedure is the same as StyleGAN 2 [Karras et al., 2020], except that we trained the network on 2M samples instead of 25M samples. With a batch size of 10, it adds up to 200k iterations. This is the same number of iterations as one of the 6 progressive growing steps of DrumGAN [Nistal et al., 2020]. It should be noted that the batch size in [Nistal et al., 2020] is at least 12. Full network training on a single Nvidia GeForce 1080GTX takes 7 days without descriptors and 10 days with descriptors. Please note that adding labels does not change the training time.

Dataset

We use the ENST-AUG dataset described in Sect. 3.5. As shown in Table 3.1, our augmented dataset is quite unbalanced, so to obtain a balanced dataset, we use a sampler that randomly takes elements from sub-datasets (one per label) such that training data becomes uniformly distributed over drum classes.

Note that dataset balancing is standard in classification tasks but uncommon for generation tasks. One such example is [Su et al., 2020], but it is the only one we could find in a generation context.

Baseline

The main objective of StyleWaveGAN is to provide better quality than other GAN-based neural synthesizers. By comparing the sound quality of these references on the same training dataset, we can determine whether our model improves on these references or not. The most appropriate candidates we could use for our baseline are DrumGAN [Nistal et al., 2020] and [Drysdale et al., 2020] since they are GAN-based models and made up the state-of-the-art in terms of GAN drum synthesizers at the beginning of the project. Unfortunately, these are not reproducible because of missing source code or/and missing or unknown meta parameters. Therefore, we will compare to [Ramires et al., 2020] using the distributed code and a re-implementation of [Donahue et al., 2019], both trained on our augmented dataset.

Because NeuroDrum [Ramires et al., 2020] works at a 16kHz sample rate, we adapted our model to use this sample rate for this comparison. We also compared with WaveGAN [Donahue et al., 2019] using our dataset with 44.1kHz. Here, we configured both networks to generate 0.3s (@44.1kHz).

4.2.4 Objective evaluation of sound quality

We used the Fréchet Audio Distance (FAD) [Kilgour et al., 2019], described in Sect. 2.3.1, as a reference-free evaluation metric already described in a previous section. We compared the embedding of the augmented database to the embedding obtained from 64k samples generated by the evaluated network. We aim to measure the sound quality, as estimated by the FAD, between StyleWaveGAN and baseline networks and different configurations of StyleWaveGAN. Regarding computational inference cost, we achieve a generation rate of 52 drum sounds/s on one 1080GTX with the network in full resolution (1.5s at 44.1kHz sample rate).

The results with the FAD are presented in tables Tab. 4.1 to 4.4.

Network	FAD
Baseline [Ramires et al., 2020]	25.35
StyleWaveGAN@16kHz	**11.48**

Table 4.1: FAD comparison to NeuroDrum [Ramires et al., 2020] (lower is better)

Network	FAD
Baseline@44.1kHz [Donahue et al., 2019]	13.08
StyleWaveGAN@44.1kHz (SWG)	**7.75**
SWG +Discriminator with AutoFade (AF)	**6.84**
SWG + Balanced dataset (B)	7.89
SWG + AF + B	7.92

Table 4.2: FAD on networks without any conditioning (lower is better)

The first result for the unconditioned synthesis is that, compared to the baseline, we improved our result in terms of FAD, as shown in Tab. 4.1 and 4.2.

Network	FAD
SWG + labels	6.85
SWG + labels + AF	6.72
SWG + labels + AF + Balanced data (B)	6.65
SWG + labels + AF + B + Envelope	**3.62**

Table 4.3: FAD on label-conditioned networks (lower is better)

Class	SWG	SWG + AF + B	SWG + AF + B + Env
Kick	8.79	11.71	**3.58**
Snare	7.87	7.53	**4.29**
Tom	8.17	8.09	**6.27**
Closed HH	10.12	6.97	**4.23**
Open HH	8.26	8.91	**4.12**

Table 4.4: Intra-class FAD for label-conditioned StyleWaveGAN

We can also see from Tabl. 4.2 that using AutoFade in the discriminator helped to get a better generation in this context. These results confirm the benefit of AutoFade when added to the discriminator, which leads us to keep it in the discriminator for all following experiments.

The results with dataset balancing are mitigated. Without the label conditioning, using it did not decrease the FAD: since it makes the training and evaluation dataset different (in proportions), the learned distribution differs, negatively impacting the FAD. This result in the unsupervised context can be seen in Tabl. 4.2. However, it improved the supervised generation, as seen in Tabl. 4.3. The impact on the intra-class FAD of AutoFade and dataset balancing is shown in Tabl. 4.4. It lowers the FAD generally, except for the kick and open hi-hat. Since we see more improvements than deterioration in FAD, we will use the data balancing approach in our subsequent experiments. Output envelopes have a powerful impact on the FAD for all drum classes. They reduce the FAD by nearly a factor of two for all drum classes, besides for the tom. Since it gave the most significant improvement in terms of FAD, we will use envelopes estimated from the dataset in all of our subsequent experiments.

4.2.5 Subjective evaluation of sound quality

In Sect. 4.2.4, a comparison of StyleWaveGAN, NeuroDrum [Ramires et al., 2020], and WaveGAN [Donahue et al., 2019] was made using the reference-free Fréchet Audio Distance. As FAD measurements alone are insufficient, a subjective evaluation was conducted to assess a few selected methods from the state-of-the-art. We will describe the details of this subjective evaluation in the following section. We will first describe the different datasets used in this context. Then, we will explain the results of this evaluation and the extended feedback from expert listeners. Finally, we will discuss the limitations of this particular evaluation.

Evaluation Methodology

Choosing a baseline The results obtained in Sect. 4.2.4 showed a significant improvement in FAD between StyleWaveGAN and either NeuroDrum or WaveGAN. NeuroDrum and WaveGAN are reproducible but are older baselines, so we wanted to compare StyleWaveGAN to more recent methods.

For the following subjective test, we have considered three other state-of-the-art methods to be the baseline for our subjective evaluation experiment, which are DrumGAN [Nistal et al., 2020], Drysdale et al. [Drysdale et al., 2020], and CRASH [Rouard and Hadjeres, 2021]. [Nistal et al., 2020] and [Drysdale et al., 2020] both use GANs and are much closer in terms of methods to StyleWaveGAN than [Rouard and Hadjeres, 2021], which is based on an entirely different approach and requires significantly longer inference times. Accordingly, we considered the comparison with either [Nistal et al., 2020] and [Drysdale et al., 2020] most interesting since they use the same GAN framework as StyleWaveGAN. However, we couldn't reproduce any of these methods due to the lack of publicly available source code and meta-parameters. Therefore, we had to rely on comparing results produced by the original authors using their respective training sets.

A problem is the varying means of conditioning used in the different methods. While [Drysdale et al., 2020] only provides conditioning with drum type, DrumGAN presented in [Nistal et al., 2020] only has perceptual feature conditioning. A later version [Nistal et al., 2022] uses drum type conditioning and perceptual feature conditioning, but was not available when these experiments took place.

It should be noted that the generation capacity of these models differs considerably. DrumGAN can generate samples of 1.1 s at 16kHZ, while [Drysdale et al., 2020] and [Rouard and Hadjeres, 2021] generate samples of 0.4s and 0.5s at 44.1kHz, respectively. In the following discussion in Sect. 4.2.5, the test participants indicate that the decay time is essential to evaluate the realness of drum sounds. This gives an advantage to DrumGAN with its longer samples. However, the sample rate of DrumGAN is lower than StyleWaveGAN, which constitutes a major disadvantage in subjective listening tests.

It is worth noting that Drysdale et al. focus on sample-based electronic music (EM). Samples used in EM are inherently synthetic and are built to sound different from real drums. Since our subjective testing aims to evaluate how close the synthesized sounds are to a real drum, having synthetic samples in the training set will continually be assessed as worse. Given the limitations discussed before, we selected DrumGAN [Nistal et al., 2020] as our baseline, representing one of the state-of-the-art models for our perceptual test as it produces the most extended samples and uses a similar training method as StyleWaveGAN. And to obtain sounds generated by DrumGAN for our experiment, we asked Javier Nistal, author of DrumGAN, to provide a set of cherry-picked DrumGAN samples labeled Kick, Snare, and Cymbals. Since these experiments took place before the release of DrumGAN VST [Nistal et al., 2022], these provided samples came from the original DrumGAN.

Evaluation Setup This study uses datasets described in Sect. 3.5. Our subjective evaluation aims to assess the generated sound quality between DrumGAN (DG-SQ) and StyleWaveGAN (SWG-SQ) against real data references, unprocessed (ENST-OG) and

processed (ENST-AUG-EX). ENST-OG is a real-world reference, and ENST-AUG-EX contains extreme augmented examples used during the StyleWaveGAN training and can be considered a worst-case scenario regarding training data quality. SWG-SG contains samples generated by the best-performing network configuration regarding FAD, i.e., supervised StyleWaveGAN with AutoFade, dataset balancing, and output envelopes.

DG-SQ (DrumGAN) provides a state-of-the-art baseline that has been successfully compared to other approaches in the literature, as seen in [Nistal et al., 2020]. However, since the samples used are at 16kHz sample-rate (compared to SWG 44.1kHz), the comparison will be unfair to DrumGAN. As such, the comparison between the two methods can only be considered as a sanity check.

Listeners and test conditions

Both listening tests took place remotely. Because of this, the evaluation conditions were not as controlled as usual listening tests. To add as much control to the test as possible, the test participants were asked to take the test in a silent environment or, at least, reduce the background as much as possible. To add a form of loudness reference, a test sound (a snare drum sound at -23LUFS-M) was used for test participants to set the loudness to a "comfortable level". All test stimuli were normalized using the Loudness Unit (and a level of -23 LUFS-M).

For the perceptual quality evaluation, all test participants were presented with 24 samples to rate on a 5-point scale. The five levels of the scale are "Bad," "Poor," "Fair," "Good," and "Excellent" and are represented with values of 1,2,3,4, and 5, respectively. We randomly picked these samples among the ENST-OG, ENST-AUG-EX, SWG-SQ, and DG-SQ datasets. The mean opinion score (MOS) is calculated as the average score given by the test participants. Nine (9) participants took part in this test. Even if the number of participants (9) is low, most (5) are audio professionals and can be qualified as expert listeners. Regarding listening equipment, the participants, either professional or not, used their own listening devices, like studio speakers or headphones.

Listening device	Sound Quality
Studio Speakers	2
Headphones	7
Earbuds	0
Age Group	**Sound Quality**
Age 0-17	0
Age 18-25	1
Age 26-40	4
Age 41-65	4
Age 66+	0

Table 4.5: Listening devices and age groups for the subjective evaluation of sound quality

Results of the subjective evaluation of sound quality

The total Mean Opinion Score (MOS) with their confidence interval at 95% is shown in Tabl. 4.6 and Fig. 4.3.

MOS / Dataset	All Labels	Cymbals	Kick	Snare
ENST-OG	4.2 ± 0.3	4.1 ± 1.1	4.1 ± 0.6	4.4 ± 0.3
ENST-AUG-EX	3.8 ± 0.5	3.3 ± 1.3	4.0 ± 0.5	3.9 ± 0.5
SWG-SQ	3.5 ± 0.4	3.9 ± 0.7	3.0 ± 0.7	3.6 ± 0.8
DG-SQ	2.3 ± 0.5	2.3 ± 1.3	2.8 ± 0.6	1.6 ± 0.8

Table 4.6: MOS on different datasets depending on the instrument label (1 is lowest, 5 is highest)

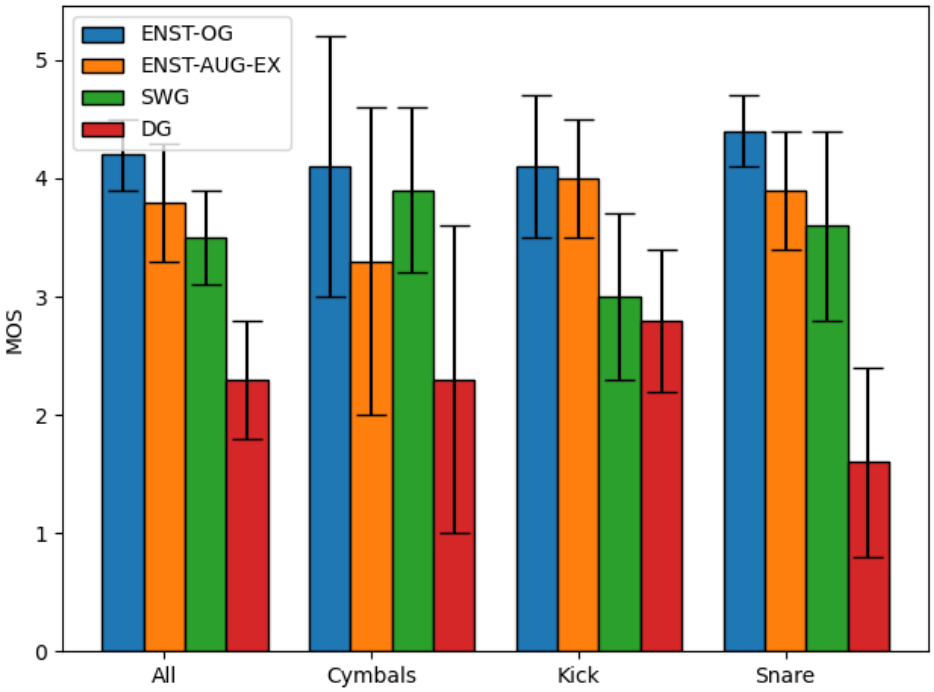


Figure 4.3: MOS on different datasets depending on the instrument label (1 is lowest, 5 is highest)

As a first result, we can see the score of the augmented samples in ENST-AUG-EX is slightly lower than the actual samples from the original dataset ENST-OG. It indicates that the extreme cases of our augmentation strategy are indeed too extreme and less natural than the originals. This is especially true for the snare and the cymbals. If we reuse the same augmentation pipeline in future works, we should select less extreme augmentation parameters. The main problem reported by test participants is that pitch change negatively affects the sound's attack, making them sound less natural than the real data. It is sufficiently present to be perceived and graded worse than an authentic sample. However, these extreme parameter combinations remain rare in the complete set of augmented sounds and act as a worst-case scenario in this evaluation.

Rest to compare StyleWaveGAN samples to DrumGAN samples. However, since DrumGAN samples use a lower sample rate (16kHz) and are not coming from a model trained on

ENST-AUG, we can expect them to have less high-frequency content (8kHz and above). While this is not too much of an issue with the kick drum due to the lack of meaningful high-frequency content, it makes DrumGAN appears worse for drum and cymbal sounds with high amount of noise content. In other words, snare drum and hi-hat. Another problem is the training dataset, as these DrumGAN samples are coming from the dataset described in [Nistal et al., 2020] and not ENST-AUG. Given all these limitations, having such sounds in the experiments can only serve as a sanity check for the other tested sounds, and not so much a regular comparison.

All these limitations are transcribed into the results. For the snare drum and the hi-hat, the lower sample rate of DrumGAN samples makes them take a significant drop in terms of MOS. However, for the kick drum, SWG and DrumGAN perform sensibly the same in terms of perceived quality: the lower sample rate does not impact the perceived quality here, and more details on the reasons for SWG shortcomings are described below. Since all the potential issues listed above happened, we can say that the augmentation process and StyleWaveGAN results should compete fairly against a DrumGAN with a higher sampling rate, like DrumGAN VST [Nistal et al., 2022], when trained on the augmented ENST-Drums dataset.

We now discuss the StyleWaveGAN results in detail. Given that we trained StyleWaveGAN on the entire dataset of augmented samples, a perfect model should produce results between the test results of the actual data and the extreme examples of the augmented data. We note that StyleWaveGAN achieves this performance level only for cymbals. Snare and kick synthesis are evaluated as less natural than the augmented samples. A discussion with the participants of the perceptual tests, who are audio professionals, reveals the following problems. First, StyleWaveGAN fails to produce the long tail of resonances that characterize kick drum sounds. In addition, this discussion brought up that the model lacks energy in the frequency band below 100Hz for the kick drum, making it sound not as good. For snare drum synthesis, the main issue is that SWG creates hybrids of sounds generated with sticks, mallets, and brushes. Concatenating, for example, an attack of a snare sound obtained with a stick and the decay of a snare sound obtained with a brush creates fair-sounding but unrealistic samples.

These difficulties with kick and snare sounds indicate that the current implementation of the discriminator is insufficient. However, increasing the capacity of the discriminator would extend the training duration, which is already quite long (7 days on one 1080 GTX). We showed evidence of the capacity of our network processed samples using state-of-the-art solutions (SuperVP) as well as an indirect comparison to DrumGAN. Given the limited time we had for this project, the subsequent investigations will focus on the question of sound control. We will keep the question of improving the discriminator for future studies.

4.3 Timbral controls

4.3.1 Introduction

Having achieved experimental validation of the enhancements brought by StyleWaveGAN over state-of-the-art neural drum synthesizers, with favorable results across both quanti-

tative and qualitative metrics, we are now ready to focus on timbral controls, reassured in the soundness of our approach. When it comes to timbral features, the AudioCommons feature set [Pearce et al., 2016] is the most commonly found in the state-of-the-art. Regarding the state-of-the-art, there exist two proposals: [Ramires et al., 2020] and [Nistal et al., 2020].

In their study, Ramires et al. employed a U-Net architecture and incorporated timbral features and the energy envelope of the target signal as conditioning inputs. However, the U-Net's objective function centered solely on reconstruction loss without explicitly accounting for control precision.

On the other hand, Nistal et al. applied the ACGAN principle [Odena et al., 2017] to estimate the timbral features. In this setup, the generator uses the timbral features as conditioning, just like Ramires' U-Net, while the discriminator is designed to achieve two tasks: distinguishing real and fake samples, as in a standard GAN discriminator, and predicting the accurate timbral features of both real and fake samples. An additional mean squared error (MSE) loss term is incorporated into the generator's objective function to encourage the generator's effective utilization of the timbral feature conditioning.

However, both these methods use the descriptor *values* only. In Ramires' case, the descriptor values are only found as conditioning input, not as part of the training objective. Nistal et al. address this limitation by utilizing the discriminator to estimate the timbral features through the ACGAN principle explained earlier. However, it remains uncertain how accurately the control values are approximated and to what degree the characteristics contributing to the timbre descriptors are manifested in the generated sounds.

To tackle this issue, we will study how we can incorporate the features' *information* directly into the training loop and measure how accurate the control can be, objectively with ad-hoc metrics and subjectively with psychophysical tests. fully met for either [Ramires et al., 2020] or [Nistal et al., 2020].

4.3.2 AudioCommons descriptors

The Audio Commons project implements a collection of perceptual models that describe high-level timbral characteristics of a sound [Pearce et al., 2016]. These features are specially crafted from the study of popular timbre designations given to a collection of sounds from the Freesound dataset. The perceptual models were built by combining existing low-level features found in the literature [Peeters, 2004], which correlate with the chosen timbral designation. These descriptors are built to match the training dataset used by Pearce et al. such that the descriptor range goes from 0 to 100 on their dataset.

The available timbral features are summarized in Tabl. 4.7.

4.3.3 Differentiable AudioCommons descriptors

Contrary to [Ramires et al., 2020] and [Nistal et al., 2020], we reimplemented the algorithms describing those timbral features to use them during the training process as differentiable functions.

Model	Description
Booming	refers to a sound with deep and loud resonant components. It is computed from the weighted average of loudness estimation on 1/3 octave bands, weighted towards lower frequencies.
Brightness	Refers to the clarity and amount of high-pitched content in the analyzed sound. It is computed from the spectral centroid and the spectral energy ratio.
Depth	refers to the sensation of perceiving a sound coming from an acoustic source beneath the surface. A linear regression model estimates the depth from the spectral centroid of the lower frequencies, the proportion of low-frequency energy, and the low-frequency limit of the audio.
Hardness	refers to the stiffness or solid nature of the acoustic source that could have produced a sound. It is estimated using a linear regression model on spectral and temporal features extracted from the attack segment of a sound event.
Roughness	refers to a sound's irregular and uneven sonic texture. It is estimated from the interaction of peaks and nearby bins within frequency spectral frames. When neighboring frequency components have peaks with similar amplitude, the sound is said to produce a 'rough' sensation.
Sharpness	refers to a sound that might cut if it were to take on physical form. It is computed from the loudness on 1/3rd octave bands, which is then weighted and averaged against the energy.
Warmth	refers to sounds that induce a sensation analogous to that caused by the physical temperature. It is calculated with a linear regression between the fundamental frequency, the high-frequency decay, and the energy in the "warmth" region (below 900Hz)

Table 4.7: Summary of AudioCommons models [Pearce et al., 2016]. The descriptors we chose for controlling the synthesis of StyleWaveGAN are highlighted in bold.

While neural networks should be capable of estimating these timbral features, they can encounter some difficulties. The most comparable method to ours, DrumGAN [Nistal et al., 2020], employing the ACGAN principle, utilizes the GAN's discriminator, trained on the fly, to predict the desired control feature value from the input sounds. However, this method doesn't necessarily guarantee the same level of accuracy as directly implementing the timbral features based on a reference implementation. Notably, if the generator and discriminator fail to converge and replicate the training data distribution, it remains uncertain how accurately the control values are approximated and to what extent the characteristics contributing to the timbre descriptor are faithfully reflected in the generated sounds. Moreover, implementing the features directly allows for a correct evaluation of signals with descriptor values outside the range of values available in the training dataset. For this reason, we chose to implement these descriptors in our model directly as differentiable functions. The code for our implementation of these descriptors can be accessed at https://github.com/ALavault/tf_timbral_models.

Implementation methodology Before considering the actual implementation methodology, it's worth noting that the original implementation of AudioCommons relied on

Numpy [Harris et al., 2020]. While Numpy is an excellent toolbox for signal processing in Python, it does not support automatic differentiation. This means we had to re-implement most signal processing blocks (spectrograms and filters most notably) to fit the differentiable operators available in Tensorflow.

For instance, the spectrograms have been reimplemented using the STFT block from Tensorflow, while the filters have been reimplemented using a spectral conversion. We cannot expect a one-to-one correspondence because these implementations differ slightly from the original AudioCommons descriptors. However, we have achieved a maximum error of less than 1% on the ENST-OG dataset. We assume this error is imperceptible, and the subsequent results from our subjective evaluation support this.

Reducing the number of features StyleWaveGAN uses only 3 out of the 8 descriptors found in bold in Tabl. 4.7. These descriptors were selected following an informal survey among drummers and musicians. The survey consisted of the following question: "Among the following features (cf. Tabl. 4.7), which one would you like to have in a drum synthesizer ?" Brightness and warmth were deemed necessary as they represent opposite ends of the frequency spectrum and are standard terms in the music production jargon. Depth was interesting since it allowed for temporal manipulation of the lower frequencies, especially their decay. While the other AudioCommons features are also of interest, we focused on these three as they represent the preferred choices of potential future users.

Integration in StyleWaveGAN Finally, we integrated these differentiable descriptors into the training loop as a constraint on the generator. By including a penalty term, specifically the $L1$ norm between the features from the current batch of training data x and the features calculated on the generated batch $G(z)$, the generator is encouraged to learn how to follow the descriptor input and produce sounds that match the desired features. In our case, x is a batch of drum sounds drawn from the training database. In other words, the generator is adapted to minimize the following loss:

$$l_{new}G(z,c,x) = l(G(z,c,F_T(x))) + \underbrace{|F_T(x) - F_T(G(z,c,F_T(x)))|}_{\geq 0} \tag{4.9}$$

where l is the original loss of the StyleWaveGAN generator, x is a batch of training data, z is the latent vector, c are the one-hot encoded labels, and $F_T(\dot{})$ are the values of the features given its input. Here, $|F_T(x) - F_T(G(z,c,F_T(x)))|$ serves as a penalty to incite the network to follow the control input. Remark that the labels c are not mandatory for the timbral control to work. Our differentiable descriptors replace the proxy network, meaning that the estimation of the features can be done with better robustness than the one done by the proxy network. This also allows our discriminator to use its total capacity for its discrimination task contrary to [Nistal et al., 2020], which uses the discriminator's capacity to estimate the timbral features.

4.3.4 Objective evaluation

In this section, we discuss the results of our control method compared to [Ramires et al., 2020] and [Nistal et al., 2020] using the metric from [Ramires et al., 2020]. We also

propose a new metric for objective evaluation to address the shortcomings of the aforementioned metric.

Brightness

We only focus on one class (snare) and one descriptor (brightness) for the first presentation of the idea. Figure 4.4 shows the relation between target and synthesized brightness for NeuroDrum and StyleWaveGAN. To create a baseline to compare ourselves to, we trained NeuroDrum on this subset with only the brightness descriptor. Results are shown as mean values and standard deviation in black dots (StyleWaveGAN) and blue crosses (NeuroDrum). The solid red vertical lines show the limiting values in the training dataset. The reference target values used for the ordering comparison according to [Ramires et al., 2020] are marked with dotted blue lines. Finally, a histogram of the brightness values on the target dataset is overlaid in light blue.

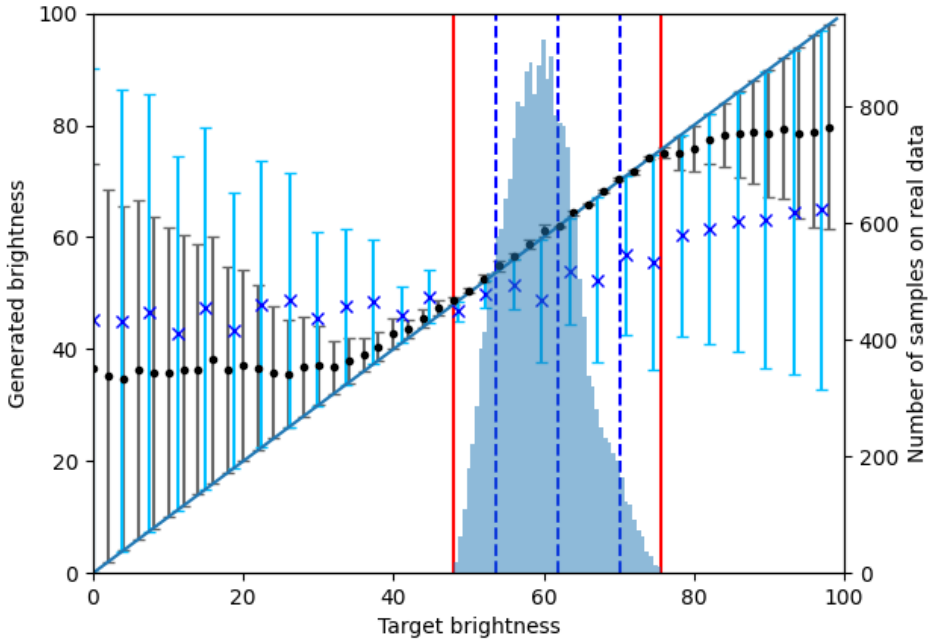


Figure 4.4: Target brightness vs. generated brightness (single descriptor). Black dots are for StyleWaveGAN, and blue crosses are for NeuroDrum

Figure 4.4 demonstrates that while the mean value of the perceptual brightness of a sound produced by NeuroDrum increases with the target brightness, it remains far off the target brightness most of the time. In contrast, the synthesized brightness of StyleWaveGAN is very close to the target value for all values present in the training set. It remains somewhat close to the target outside the brightness limits of the training data.

To compare to [Ramires et al., 2020; Nistal et al., 2020], we are using the ordering criterion first introduced in [Ramires et al., 2020] and later used in [Nistal et al., 2020], and described extensively in Sect. 4.3.4.

The small error in the synthesized feature values generated with StyleWaveGAN results in a consistent ordering for all three criteria. Table 4.8 reproduces the results for brightness

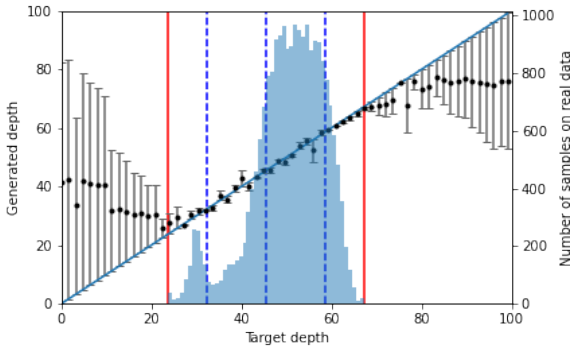
Features	E1		E2		E3	
dataset / Network	D1	D2	D1	D2	D1	D2
DrumGAN	0.74	-	0.71	-	0.7	-
NeuroDrum	**0.99**	0.91	**0.99**	0.80	**0.99**	0.68
SWG	-	**1.00**	-	**0.94**	-	**0.98**

Table 4.8: Ordering accuracy for the feature coherence tests for brightness on samples generated with the baseline NeuroDrum [Ramires et al., 2020] and DrumGAN (from [Nistal et al., 2020]). D1 refers to the results obtained in [Nistal et al., 2020] and reproduced here, and D2 refers to the results obtained on our augmented dataset, ENST-AUG. (Higher scores are better)

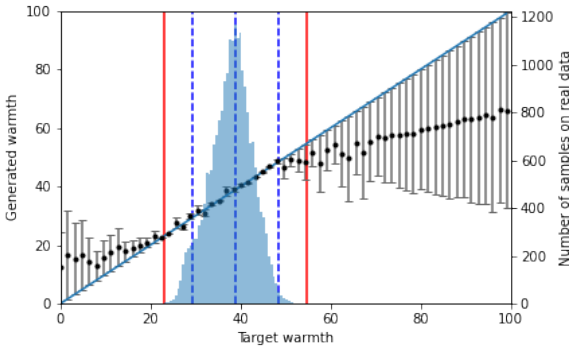
control taken from table 3 in [Nistal et al., 2020] comparing NeuroDrum and DrumGAN, trained on a different dataset under the column "D1". The results under column "D2" are for our network, trained on our augmented dataset. We matched and improved the results from NeuroDrum in this configuration. Since NeuroDrum performed consistently better than DrumGAN on D1 and StyleWaveGAN performed consistently better than NeuroDrum, we can assume that StyleWaveGAN should perform better than DrumGAN if trained on the same dataset. All these results support our hypothesis that replacing a trained feature estimator as in [Nistal et al., 2020; Odena et al., 2017] by directly implementing the feature estimator allows for a significantly improved control consistency of the final network.

Other Descriptors

We will discuss here the results for the depth and warmth descriptors. Results are shown in Tabl. 4.9 and Fig. 4.5a and 4.5b. In Fig. 4.5a and 4.5b, a histogram of the dataset values can be observed overlapped in light blue similarly to Fig. 4.4.



(a) Target depth vs. Generated depth (single descriptor)



(b) Target warmth vs. Generated warmth (single descriptor)

Figure 4.5: Results for depth and warmth descriptors in the single descriptor case.

Figure 4.5a shows the results for the depth descriptor. This figure shows that the depth descriptor behaves similarly to the brightness descriptor, shown in Fig. 4.4. Note, however, that the extrapolation appears slightly less consistent and smooth.

From this result, we can infer that the network has a harder time learning the depth descriptor. The deviation from the reference line for low depth (depth < 30, marked by the first blue dashed line) can be explained by the low number of samples to train the network at this level, i.e., less information to learn from, which may result in overfitting to the limited data available and reduced generalization capability.

Figure 4.5b shows the results for the warmth descriptor. The performance is on par with the brightness descriptor except for the region between the third dashed line and the rightmost red solid line. This can be explained by a lack of training data in this region, as shown on the overlaid histogram. This region represents the values that are greater than 0.8 when using the min/max normalization, i.e., $\{x \in X | \frac{d(x)-\min_X d(x)}{\max_X d(x)-\min_X d(x)} \geq 0.8\}$ where X represents the samples from the dataset and d the descriptor of interest.

These results from Fig. 4.5a and 4.5b are coherent with results from the criterion shown in Tabl. 4.9. The deviation for low depth values is correlated to a lower E3 feature (low vs. mid) and similarly for higher warmth values with a slightly lower E2 feature (mid vs. high). The rest of the evaluation criteria are over 90% of correct ordering, which is coherent with the almost linear but slightly noisy examples shown in Fig. 4.5a and 4.5b but is slightly worse than the brightness descriptor.

Descriptor \ Evaluation Point	E1	E2	E3
Depth	0.99	0.99	0.71
Warmth	1.00	0.86	0.90

Table 4.9: Ordering accuracy for other features of interest using StyleWaveGAN (higher is better)

Results with Multi-dimensional Descriptor Controls

Using three individual networks for controlling the individual descriptor is not that interesting for a real-world application. In this next step, we investigate controlling the network with a 3-dimensional vector of warmth, depth, and brightness descriptors.

Ordering Criterion We use the same label to evaluate the control quality with the ordering accuracy but change the evaluation method slightly. While we use the same ordering criterion, we generate samples in a way that can create sounds outside of the training dataset. More precisely, we take a set of actual features from a batch of the training data and then modify the descriptor to be evaluated to 20, 50, or 80 percent of the min/max value for the said descriptor.

More mathematically, this means that for a sample x from the dataset, we estimate the descriptors values $D(x) = (d_{brightness}(x), d_{depth}(x), d_{warmth}(x))$. The descriptor d to be evaluated is then changed to a value d_{target} such that :

$$\frac{d_{target} - \min_X d(x)}{\max_X d(x) - \min_X d(x)} \in \{0.2, 0.5, 0.8\}$$

Since the network is trained on data, it will learn to reproduce similar features as the actual data, which means only a part of the possible combinations. When using descriptors simultaneously for control, unseen combinations may happen. While these combinations are not problematic if they can be interpolated, extrapolation to unseen data can lead to issues. This behavior is shown in Fig. 4.4, 4.5a and 4.5b where the fidelity is strong within the dataset limits with minimal dependence on the training dataset statistics, but the extrapolation capabilities are limited.

This method's results are shown in Table 4.10.

Descriptor \ Features	E1	E2	E3
Brightness	1.0	1.0	1.0
Depth	1.0	1.0	0.99
Warmth	0.98	0.59	0.97

Table 4.10: Ordering accuracy for multiple descriptors using Multi-dimensional Descriptor Controls with StyleWaveGAN (higher is better)

Table 4.10 shows some interesting and strong results. First, the brightness descriptor achieves full ordering accuracy on all three target values. Almost the same can be said for the depth descriptor where E3 lowers to 99%, i.e., between low and mid targets. The warmth descriptor performs well on E1 and E3, i.e., low to high and low to mid, when the E2 value is low. Given that the two other values are higher than 97%, there is an issue with the higher values in the dataset, but not enough to disturb the low to high accuracy. We already know there is a low of training data in the E2 region, as shown on the histogram in Fig. 4.5b, and this is most likely the reason for the poor performance here.

Mean Absolute Error Metric As shown in Tabl. 4.10, training the descriptors with the proposed differentiable error function produces a network following controls with a precision such that the ordering criterion proposed in [Ramires et al., 2020] and used in [Nistal et al., 2020] is no longer sufficient to evaluate the control precision. As a criterion based on ordering rather than accuracy, it is limited in its ability to capture the complexity and nuances of the problem fully. Because Tabl. 4.10 shows that all descriptors, except for one, achieve an accuracy of over 97% in ordering, a new metric is necessary to more precisely capture the limitations of the timbral control method that was tested and to investigate the behaviors that decrease the criterion. In the following, we propose a refined evaluation criterion that allows evaluating control precision with more precision, not considering ordering but errors and linearity.

This criterion will be using the Mean Absolute Error (MAE) between the target values and the output values on three regions based on quantiles of the dataset values :

- F1: MAE evaluated using only the target descriptor values within the 20th and 50th quantiles
- F2: MAE evaluated using only the target descriptor values within the 50th and 80th quantiles

- F3: MAE evaluated using only the target descriptor values within the 20th and 80th quantiles

First, the interest in working with quantiles is that we expect to cover the same amount of dataset values each time while avoiding extreme values.

The results are shown in Tabl. 4.11. The values in the table are not percentages or relative to the descriptor values; they are absolute errors. We also note that these errors have the same unit as the described descriptors.

In Table 4.11, the lines labeled *single* show the results using networks with only one descriptor, and the lines labeled *combined* show the results when the descriptor of interest is set. Still, the others are taken from an actual sound from the training dataset. The lines labeled "*combined, dataset*" show the results when all the descriptors values are taken from the training dataset.

Configuration \ Features	F1	F2	F3
NeuroDrum (brightness, single)	7.22	10.40	8.81
Brightness (single)	0.83	1.06	0.98
Depth (single)	1.06	1.15	1.10
Warmth (single)	1.15	1.01	1.08
Brightness (combined)	0.97	1.36	1.17
Depth (combined)	1.33	1.50	1.41
Warmth (combined)	1.29	3.31	2.33
Brightness (dataset, combined)	0.75	0.95	0.85
Depth (dataset, combined)	0.99	1.03	1.00
Warmth (dataset, combined)	1.42	1.37	1.39

Table 4.11: Mean absolute error for several configurations (lower is better)

StyleWaveGAN performs significantly better than NeuroDrum in every tested configuration. The combination of descriptors tends to worsen the MAE slightly.

Impact of control loss on control precision We will start the evaluation of the effectiveness of the timbre control for the different timbre features by means of studying the difference obtained when using two different loss functions. The two loss functions we have studied as control loss are the $L1$ and $L2$ norm of the deviation between the target and generated timbre features.

The $L2$ loss performs better than the $L1$ loss when using brightness and depth with values from the dataset as seen in Tabl. 4.12. Outside of the dataset, the interpolation capabilities generally suffer from worse performances overall than the $L2$ norm, except for the depth descriptor. Results with warmth are considerably inferior when using the $L2$ loss in both cases. An offset can explain the problem with the warmth descriptor and $L2$ norm as the control loss. This phenomenon can be seen in Fig. 4.6b when compared to Fig. 4.6a, which shows the results with $L1$ loss. Red lines show the limit values of the dataset, and the green vertical lines show the position of the 20th, 50th, and 80th quantiles.

Configuration \ Features	F1	F2	F3
Brightness ($L1$, combined)	0.97	1.36	1.17
Depth ($L1$, combined)	1.33	1.50	1.41
Warmth ($L1$, combined)	1.29	3.31	2.33
Brightness ($L2$, combined)	1.16 (+19.6%)	1.73 (+27.20%)	1.45 (23.93%)
Depth ($L2$, comb.)	1.21 (-9.02%)	1.29 (-14.00%)	1.26 (-10.63%)
Warmth ($L2$, comb.)	4.96 (284.5%)	2.49 (-24.00%)	3.69 (58.37%)
Brightness ($L1$, dataset, combined)	0.75	0.95	0.85
Depth ($L1$, dataset, comb.)	0.99	1.03	1.00
Warmth ($L1$, dataset, comb.)	1.42	1.37	1.39
Brightness ($L2$, dataset, comb.)	0.66 (-12.00%)	0.85 (-10.52%)	0.76 (-10.59%)
Depth ($L2$, dataset, comb.)	0.77 (-22.22%)	0.54 (-47.00%)	0.65 (-35.00%)
Warmth ($L2$, dataset, comb.)	6.92 (387.32%)	6.28 (358.39%)	6.60 (374.82%)

Table 4.12: Absolute and relative Mean Absolute Error for the $L2$ loss compared to the $L1$ loss (lower is better). For the $L2$ loss, values between brackets show the relative difference between the MAE with $L1$ and $L2$ losses.

Due to time constraints, we could not perform further studies to determine why we observed an offset when evaluating the model trained with 3D perceptual controls using $L2$ loss. It is unclear whether this is due to a local minimum that we can solve through retraining with different weight initialization or if it indicates a systematic issue with the loss weighting, which we could address by increasing the weight of the warmth descriptor error in the loss function. Unfortunately, conducting systematic evaluations of numerous training runs is time-consuming, and we could not conduct further studies.

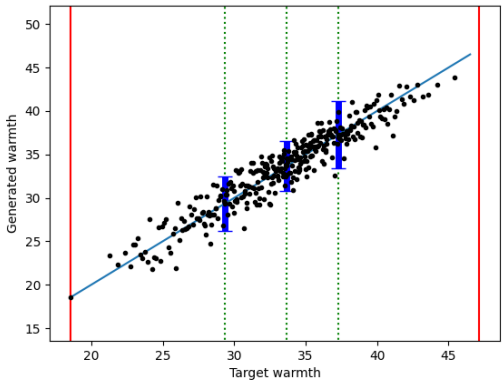
The overall worse performance with the $L2$ loss confirms our original implementation choice of using $L1$ loss.

Linearity Since we improved significantly over our baseline, this motivates us further to study the behavior of our timbral control method. Suppose the control method works correctly and produces a perfect output that matches the control input exactly. In that case, we should observe a strong linear correlation between the output and the input. To evaluate this, we will compute a linear least-square regression on the domain bound by the 20th and 80th quantiles and use its determination coefficient R^2 as a metric of good linearity. In this case, R^2 is equal to :

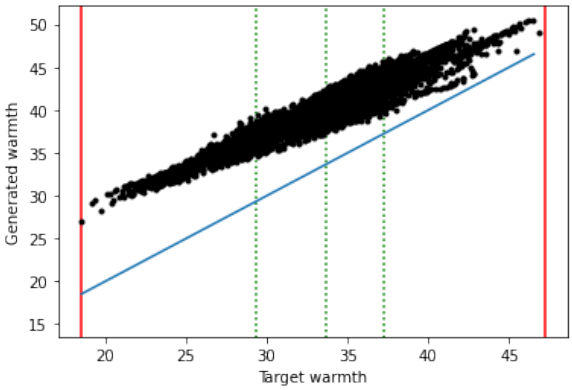
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.10}$$

where n is the number of samples, y_i is the output value of the i-th measure, $\hat{y}_i$ the corresponding predicted value and $\bar{y}$ the average of the measured values. The results are compiled in Tabl. 4.13.

Apart from a significantly better fit than NeuroDrum, we can see that the determination coefficient is generally quite satisfying except for the warmth when used with values



(a) Generated values for warmth when synthesizing snare drum sounds with control values obtained from the snare drum sounds in ENST-AUG using StyleWaveGAN trained with the $L1$ loss for the control error and combined descriptors. The blue error bars show the just-noticeable difference at the points of interest obtained with our psychometric experiment described in Sect. 4.3.5



(b) Generated values for warmth when synthesizing snare drum sounds with control values obtained from the snare drum sounds in ENST-AUG using StyleWaveGAN trained with $L2$ loss for the control error and combined descriptors.

Figure 4.6: Effect of loss function on generated descriptors

outside the dataset. The determination coefficient, also known as R-squared (R^2), is a measure that can be interpreted as the percentage of variance explained by the linear model. In this study, the R-squared values were generally above 50%, indicating that a significant portion of the variability in the data can be explained by the linear relationship between the descriptor control and its target, in our case, the timbral descriptors. Therefore, we can conclude that the behavior of the descriptor control is mostly linear with respect to its target, which is far from being the case for NeuroDrum on our dataset. This result is illustrated in Figure 4.7c, where there is a bend in the output value, where the warmth descriptor values are taken in an evenly spaced interval between 0 and 100, hence with combinations that are not present in the dataset.

This bend is due to the distribution of descriptor values in the dataset. Indeed, for high warmth values, the values of the other two descriptors remain confined to a narrow range (a variation of fewer than 5 points around 50 for brightness and 66 for depth, these values being already quite rare in the dataset). So, when the control inputs of brightness and depth cover the full range of descriptor values available in the dataset as our evaluation method does, the warmth value has to be extrapolated by the network since such a combination was not seen during training.

However, this behavior is not shown when evaluating control values from the dataset ($R^2 = 0.08$ when using a continuum of values becomes $R^2 = 0.45$ when using a combination seen during training). For the other descriptors, the linearity remains satisfying whatever evaluation method is used, on the dataset or using a continuum.

Overall, for unseen combinations falling within the range of the training set, the error stays below 4 points and has a good linearity, indicating a promising outcome. However, for combinations outside the dataset, the results are less conclusive. A saturation phe-

Configuration \ Features	R^2
NeuroDrum (brightness)	0.03
Brightness (single)	0.75
Depth (single)	0.70
Warmth (single)	0.76
Brightness (combined)	0.47
Depth (combined)	0.67
Warmth (combined)	0.08
Brightness (dataset, combined)	0.72
Depth (dataset, combined)	0.62
Warmth (dataset, combined)	0.45

Table 4.13: Determination coefficient for several configurations (higher is better)

nomenon occurs quickly in these areas, leading to noisy estimations. It is worth noting, however, that values outside the dataset range tend to be coherent, with values higher than the maximum value of the dataset generally remaining higher and vice versa for values lower than the minimum value. These considerations can be seen in Fig. 4.7a to 4.7c.

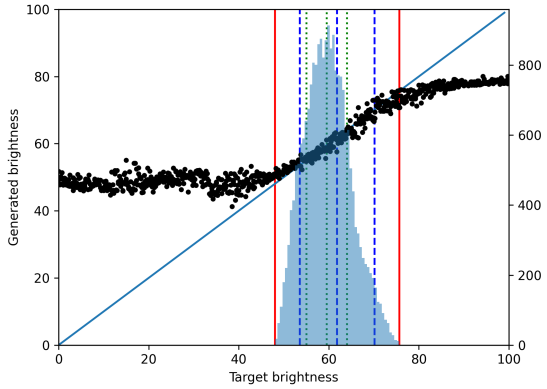
To conclude, we have demonstrated that our method for timbral controls works better than the state-of-the-art almost everywhere in the min/max values of the training dataset and can extrapolate further than the min/max values available in the dataset while retaining the overall signification of the features and interpolate between unseen combinations of descriptors within the dataset limits.

4.3.5 Subjective evaluation

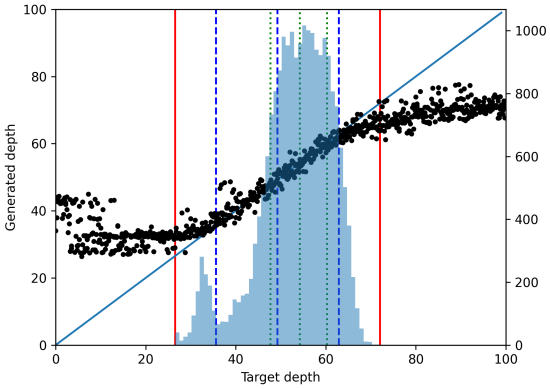
This section presents the methodology and the results of our subjective evaluation of the control error. Since we now know that our descriptor control method works with an error limited to a few percent, the main question changed from "Can we control the network with differentiable descriptors?" to "is the control error perceivable by a human listener?".

Going back to the objectives of this project, this question about the perception of the control error is key to knowing if we could use such a control method in an actual music production context. Indeed, if the error can be perceived but remains small, the control method would still be usable because most users will not listen to the drum alone. This means that in a real music production context, the control error may not be heard even if it would be perceived when isolated from the rest of the production. On the other hand, if the control error is not perceivable, this becomes more interesting. In that case, we can choose a less sophisticated control approach, simplifying the synthesizer even more. Moreover, this would imply that we don't need to improve our timbral control technique significantly. In any case, showing the potential of deep hybrid learning in the context of deep neural network control will be an outstanding achievement.

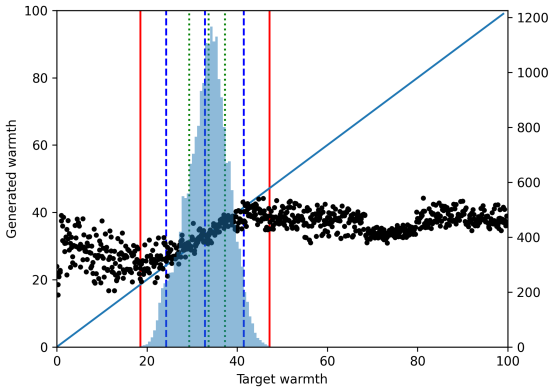
The following results are taken from [Lavault et al., 2022c].



(a) Target brightness vs. Generated brightness with combined descriptors



(b) Target depth vs. Generated depth with combined descriptors



(c) Target warmth vs. Generated warmth with combined descriptors

Figure 4.7: Target vs. Output for the combined descriptors

Subjective evaluation for control quality evaluation

We use one of the psychophysical methods described in [Giordano et al., 2012] to measure the absolute perception threshold. Our experiment uses the constant stimulus method. With the constant stimulus method, two sounds are played, one after the other, one being the reference generated with descriptor value v obtained from the dataset and the other being generated with a descriptor value $v + \Delta$. The test subjects are then asked if the stimuli were perceived as identical or different. Following [Giordano et al., 2012], we determine the differential threshold as the psychometric function's slope at the 50% threshold.

Due to time constraints, we only used a subset of snare samples in this experiment from dataset SWG-CQ. We choose to use the measurement points corresponding to the ones we used for the MAE metric described in Sect. 4.3.4. By doing so, we hope to be able to compare the subjective equality points to the MAE and draw conclusions about whether or not the deviation due to the size of the error is perceived. The descriptor values v are then computed to match the 20th, 50th, and 80th quantiles of the descriptor of interest.

The offset Δ is selected from the set

$$D = \{x | x = 0.25z \text{ with } z \in \mathbb{Z} \text{ and } -8 \leq x \leq 8\}. \tag{4.11}$$

Note that the AudioCommons descriptor values are normalized and range from 0 to 100 so that the variations used in the test always cover ±8% of the full range of the descriptor. We chose this range of variation since the MAE results in [Lavault et al., 2022b] and reproduced in Tabl. 4.11 is always lower than 4, which makes the chosen range of variation the double of a global upper bound of the MAE metrics.

The order in which we play the two samples is essential, and both orderings are in the test. Since we chose the sample pairs randomly, this ensures that all orderings are equally present. A fade-out is applied to each sample to avoid noisy tails impacting the evaluation.

Since subjective equality measures are only valid for a single stimulus intensity, having multiple target values will allow us to extrapolate subjective equality points over a broader range of values.

Descriptor	Ordering	Steps	Total
Brightness	2	65	130
Depth	2	65	130
Warmth	2	65	130

Table 4.14: Summary of absolute threshold experiments data set

Table 4.14 summarizes the dataset's content of pairs of samples from SWG-CQ used in this experiment.

Listeners and Test conditions

Like in the sound quality experiment, we provide the participants' listening devices and age distribution in Tabl. 4.15. And the evaluation methodology described in Sect. 4.2.5 is reused here as well.

Listening device	Number of Participants
Studio Speakers	2
Headphones	27
Earbuds	2

Age Group	Number of Participants
Age 0-17	0
Age 18-25	13
Age 26-40	9
Age 41-65	9
Age 66+	0

Table 4.15: Listening devices and age groups for the control quality perceptual test

Results of subjective evaluation of control error perception

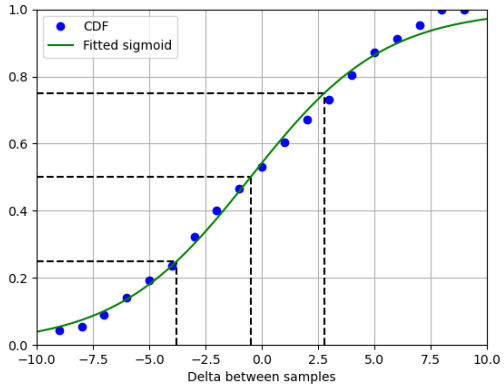
To interpret our results from the subjective evaluation of control error, we need to study different thresholds on the estimated probability distribution of the data. We are interested in the just noticeable difference, the differential threshold in this case.

We calculate an estimated Cumulative distribution function (CDF) and choose the sigmoid function as our psychophysical function. Mathematically speaking, this means we try to fit the following function to the CDF:

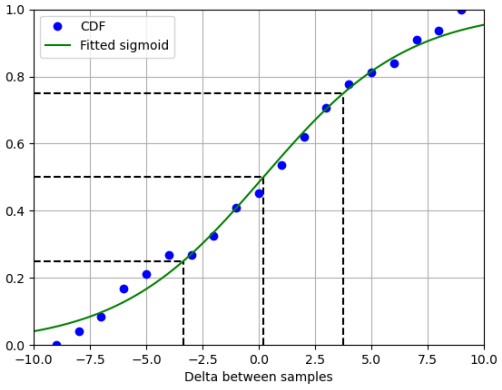
$$\psi(x, b, d) = \frac{1}{1 + \exp(-bx + d)} \tag{4.12}$$

where b and d are real parameters estimated with a least-square estimator.

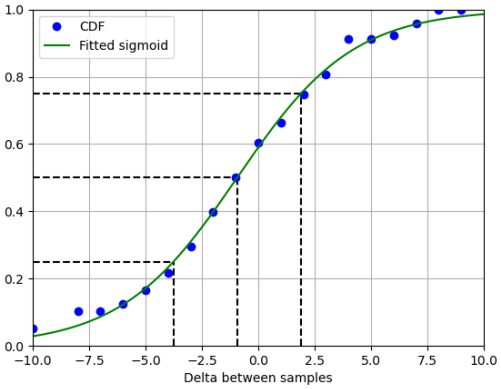
Figures 4.8a to 4.8c present the cumulative probability of detecting a difference across all the different values.



(a) Psychometric results for brightness and for the descriptor value corresponding to the 80th quantile for differential threshold estimation



(b) Psychometric results for depth and for the descriptor value corresponding to the 20th quantile for differential threshold estimation



(c) Psychometric results for warmth and for the descriptor value corresponding to the 50th quantile for differential threshold estimation

Figure 4.8: Psychometric results for differential threshold estimation

In this context, the differential threshold is the metric we are interested in. It is measured

as the slope around 50% of the psychometric function and is obtained as the difference of the 25% and 75% values, as described in Sect. 1.6.3.

These results are shown in Tabl. 4.16.

Descriptor	Q20	Q50	Q80
Brightness	2.63 (28%)	2.55 (37%)	3.29 (26%)
Depth	3.55 (28%)	2.53 (40%)	3.79 (26%)
Warmth	3.13 (45%)	2.83⋆ (48%)	3.91 (35%)

Table 4.16: Summary of differential experiments for every 3 measurement points with the fitted sigmoid. Between parenthesis, the ratio between the MAE on the dataset and the differential threshold in percentages. Values marked with ⋆ are below the MAE metric when considering values outside the dataset.

Our main goal with this experiment was to measure whether the error between our network's control and the output is perceptible. Our MAE metric measures the mean error on a segment. We chose the measurement points for the psychophysical experiment to be the endpoints of the segments of our MAE metric to be able to make some conclusion. For further illustration, we added blue error bars on Fig. 4.6a at the different measurement points. The error bars show the points of subjective equality for each measurement point. This figure shows the generated sample output values are mostly within the limits of the differential threshold on the segments of interest. Regarding the comparison with the MAE metric, we can say that it is significantly lower than the point of subjective equality, as shown in Table 4.16 where the MAE is always lower than 50% of the differential threshold.

We claim that the average error is imperceptible. Moreover, the overall error is almost always lower than the differential threshold, hence not perceivable, mainly when control values are taken from the training set (Figure 4.6a). The MAE metric is even lower than the differential threshold for values from the dataset and slightly above it in only one instance for interpolated values. Simply put, the average error is imperceptible when using values from the dataset and almost imperceptible when interpolating (except for one outlier).

Based on these strong findings, we can draw some conclusions. Firstly, the error should only be noticeable in some cases. Specifically, generating different sounds with the same descriptor controls should not result in noticeably different timbral properties. Additionally, we discovered that control is not necessarily perceived symmetrically; the same variation with a different sign could result in different points of subjective equality. Lastly, the positive results of our perceptual experiment suggest that a discrete control approach could be practical, where a step of 1 unit should be imperceptible in nearly all cases. By employing a discrete control approach with a step of 1 unit, precision would be reduced, but the advantage would be a greater convenience for professional audio production, and we now have the guarantee that it will not be perceivable.

This strong result shows how powerful deep hybrid learning can be. Combining the strengths of both deep learning and classical signal processing, deep hybrid learning can revolutionize how we approach complex control problems and enable us to achieve performance and precision previously thought inaccessible, in our case, for intuitive music

synthesizer controls.

4.4 StyleWaveGAN with trainable oscillator bank

4.4.1 Introduction

Given the encouraging results of StyleWaveGAN in terms of objective and subjective evaluations of sound quality, we set out to improve the quality even further with minimal changes to the discriminator architecture.

From the subjective evaluation we conducted, we gathered the feedback of a few expert listeners on sound quality. As stated in Sect. 4.2.5, the main problem was the lack of strong note perception in the shell sounds. This problem is understandable and somewhat expected, as generating sine waves with neural networks in the time domain can be challenging, notably for long-term stable oscillations, which are difficult to learn without additional supervision or architectural constraints.

Given the success of Differentiable Digital Signal Processing (DDSP) [Engel et al., 2020] in musical instrument synthesis, we thought of adding network-controlled oscillators to help the synthesis by providing the hard-to-generate sine waves and partials. This oscillator bank is the first step toward a hybrid synthesis combining classical oscillators and GANs.

4.4.2 StyleWaveGAN-OSC

We will describe the results obtained using more traditional synthesis concepts integrated into StyleWaveGAN.

Additive Synthesis and Differentiable Oscillators

With the release of Differentiable Digital Signal Processing (DDSP) [Engel et al., 2020], using oscillators with neural networks was possible and showed great capabilities in terms of synthesis when presented with harmonic content. A notable example of this application on the piano [Renault et al., 2022] showed that the DDSP approach made high-quality instrument synthesis and performance modeling possible with minimal networks. While the latter is not interesting to us, the former is a great inspiration for the following contribution. By using non-harmonic series (even if slightly non-harmonic), Renault et al. could generate convincing piano performances with DDSP.

What is of interest for our application is to dissociate the stable resonances, which an oscillator bank would then generate, from the residual part, which the StyleWaveGAN network would then generate. The style-based generator of StyleWaveGAN would be used as a generator for the oscillator parameters, taking the part of the controller network in the original DDSP model. The underlying premise is that StyleWaveGAN's discriminator will discern the less stable oscillations compared to real-world drum signals, prompting the generator to favor oscillators.

To help the discriminator in its task, a method akin to the constraints applied to the harmonic components of a classical additive model (described in Sect. 1.4.1 and eq. (1.2)) will be employed. Alongside this, an additional goal is to restrict the parameter space of the oscillators as much as possible. By doing so, the challenge for the discriminator is expected to be diminished, especially considering that the discriminator is already operating at its limits. It's worth noting a discriminator based on time-domain signals, like ours, is not entirely equivalent to a loss function based on spectral magnitudes like the one in DDSP. This is particularly relevant when dealing with phase since the spectral loss is independent of phase, while the time-domain discriminator is not.

Oscillator constraints Our hypotheses for the oscillators are the following:

- The amplitude is constant,
- The decay of the instant frequency follows an exponential decay,

Because StyleWaveGAN utilizes envelopes in its output stage, the inner layers operate on a nearly constant energy representation. Therefore, if we employ these oscillators before the envelope, it makes sense to use a constant amplitude. We added an exponential decay to the instant frequencies to match the pitch glide effect on drum shells.

We can justify the exponential decay of the instant frequency by the physical study of membrane modeling [Fletcher and Bassett, 1978; Avanzini et al., 2012]. In the first referenced paper ([Fletcher and Bassett, 1978]), the authors present an initial assessment indicating that drums experience a downward frequency shift caused by membrane deflection. They approximated this frequency shift with a quadratic decay model. However, the most significant finding from their study was that the relative (percentage) frequency variation was consistent across all the drum overtones. The second cited paper ([Avanzini et al., 2012]) expanded on this research and revealed a pitch glide that closely resembled exponential decay.

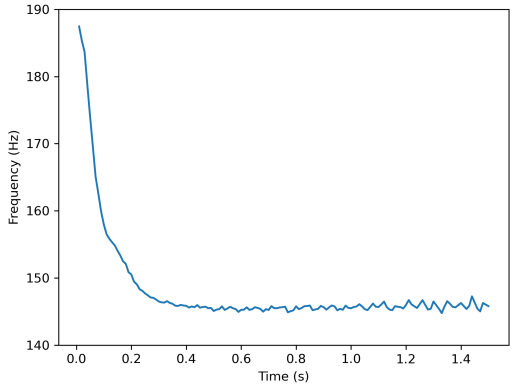
We can also show this behavior on real drum sounds compared to synthetic glides to further this information. To this end, we use a pitch tracker (CREPE [Kim et al., 2018]) to follow the pitch glide's evolution. Figure 4.9a shows the output of a pitch tracker on a real tom sound. This figure shows a clear 45Hz pitch glide, equivalent to 3 semitones. The decay shown of the fundamental in this figure appears to be close to exponential, as expected from [Fletcher and Bassett, 1978; Avanzini et al., 2012]. To further justify this claim, we compare two models of frequency evolution. The first one, we will call "linear decay," can be expressed as, for $t \geq 0$:

$$f_{linear}(t) = f_{stop} + \delta \max(-\lambda t + \delta, 0) \tag{4.13}$$

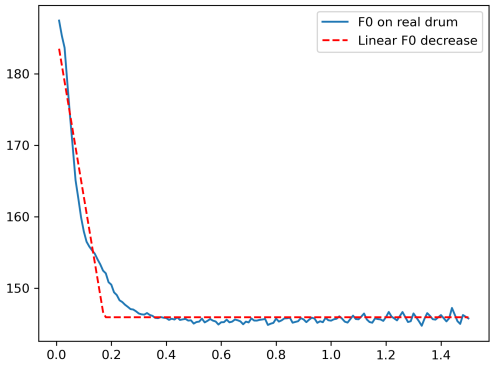
The second, we will call "exponential decay," can be expressed as, for $t \geq 0$:

$$f_{exponential}(t) = f_{stop} + \delta \exp(-\lambda t) \tag{4.14}$$

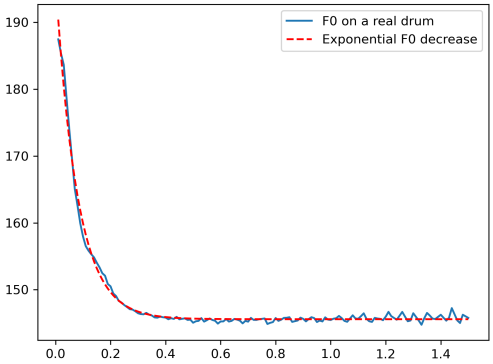
We will first compare the real frequency evolution to the linear one, as shown in Fig. 4.9b. Here, the solid blue line shows the real drum pitch estimation using CREPE, and the



(a) Pitch measurement of real tom sound (Yamaha Stage Custom 10") using CREPE



(b) Pitch measurement of a synthesized sine wave with linear pitch glide against a real drum sound pitch measurement using CREPE.



(c) Pitch measurement of a synthesized sine wave with exponential pitch glide against a real drum sound pitch measurement using CREPE.

Figure 4.9: Comparison of pitch measurements between real tom sound and synthetic sine with pitch glide using CREPE

dashed red line shows the fitted linear decay model. The linear model does not match the real drum's frequency evolution, making us discard it for further experiments.

However, when comparing Fig. 4.9a and Fig. 4.9c, the similarities are much more evident, and the synthetic model matches the overall shape of the decay. Please note, the solid blue line shows the real drum pitch estimation, and the dashed red line shows the fitted exponential decay model, just like the Fig. 4.9b.

Overall, it appears safe to assume the pitch glide of a real drum can be modeled by a decreasing exponential between two frequencies or a steady-state frequency and a pitch interval, as proposed in eqs. (4.13) and (4.14). To convert a frequency+delta model to a two-frequencies one, one should consider that $f_{stop} = f_{stop}$ and $\delta = f_{start} - f_{stop}$ with f_{start} the frequency at $t = 0$. The same experiment can be done with other tom sounds, with similar results.

Deriving oscillator phase When working with sinusoids, the argument of the sinusoid is the phase, denoted as $P(t)$. The instantaneous frequency of the sinusoid can be obtained as the derivative of the phase with respect to time, i.e., $\frac{dP}{dt}$. Suppose we want a certain frequency evolution for the sinusoid. In that case, we must integrate the desired frequency evolution to obtain the corresponding phase evolution. Specifically, if we denote the desired frequency evolution as $f(t)$, the phase evolution can be obtained as the integral of f with respect to time: $P(t) = \int_0^t f(u)du$.

When the desired frequency evolution is a decreasing exponential, the frequency decay is of the form $f(u) = a\exp(-\lambda u) + b$ with a and b two positive constant parameters. In this case, we end up with the phase P:

$$P(t) = \int_0^t f(u)du = \int_0^t a\exp(-\lambda u) + bdu = \frac{a}{\lambda}(1 - \exp(-\lambda t)) + bt \tag{4.15}$$

If we set $a = f_{start} - f_{stop}$ and $b = f_{stop}$, and $\lambda = \frac{1}{\tau}$ we end up with the instantaneous frequency:

$$f(t) = (f_{start} - f_{stop})\exp(-\frac{t}{\tau}) + f_{stop} \tag{4.16}$$

and the overall phase

$$P(t) = \tau(f_{start} - f_{stop})(1 - \exp(-\frac{t}{\tau})) + f_{stop}t \tag{4.17}$$

An oscillator of such a bank can be defined as:

$$o(t, f_{start}, f_{stop}, \tau_{decay}, \varphi_0, A) = A\cos\left(2\pi P(t, f_{start}, f_{stop}, \tau_{decay}) + \varphi_0\right) \tag{4.18}$$

We can note the instantaneous frequency $\frac{dP}{dt}$ is indeed a decreasing exponential between f_{start} and f_{stop}

$$f(t, f_{start}, f_{stop}, \tau_{decay}) = f_{stop} + (f_{start} - f_{stop})\exp\left(-\frac{t}{\tau_{decay}}\right) \tag{4.19}$$

We then have 4 parameters to control per oscillator: amplitude, start frequency, stop frequency, and decay. The phase is an additional parameter setting the initial conditions at $t = 0$ for P.

These oscillators are controlled by the style mechanism from [Karras et al., 2019, 2020] and described in Sect. 2.2.3. In other words, starting from an intermediary representation w, we end up with eq. (4.19) with:

$$\tau_{decay} = ReLU(S_{decay} \times w + b_{decay}) \tag{4.20}$$
$$f_{start} = ReLU(S_{start} \times w + b_{start}) \tag{4.21}$$
$$f_{stop} = ReLU(S_{stop} \times w + b_{stop}) \tag{4.22}$$

where the S_x are learned matrices transforming the mapped intermediary latent w into actual values and b_x a bias vector. We use the ReLU on the frequencies and the decay as a

constraint, so the parameters have meaningful and positive values. Finally, Fig. 4.10 shows how the oscillator bank is inserted inside StyleWaveGAN. The output of the synthesis generator is summed with the signal generated from the oscillator bank, and the sum is processed through the envelopes described in Sect. 4.2.2.

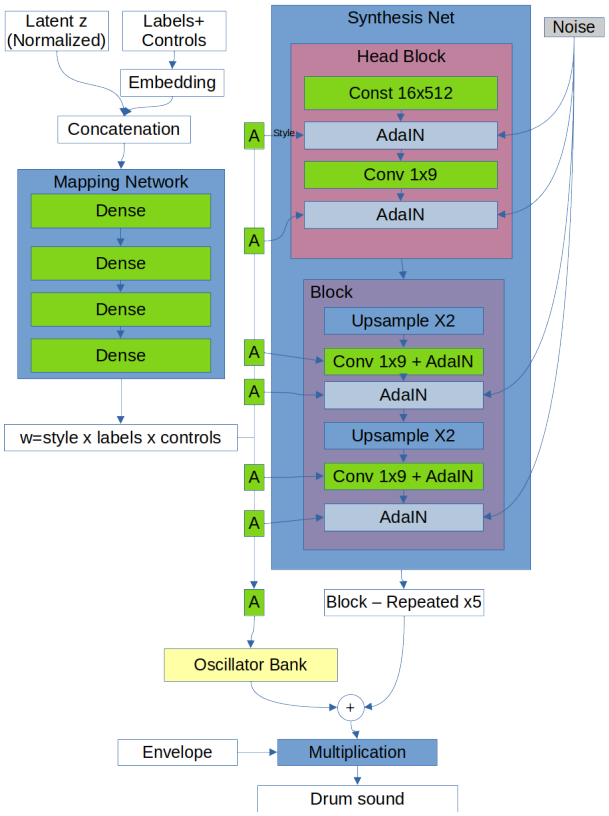


Figure 4.10: StyleWaveGAN generator with oscillator bank added.

Simplifying the oscillator model In our preliminary experiments, it turned out that despite the constraints of the oscillator parameter space, the discriminator could not properly adapt the oscillator parameters. Frequency crossing was the main problem we encountered in these preliminary experiments. We speculated that such crossings did appear due to the low-frequency modulations of the sinusoids observed in the generated sounds. Given the oscillator amplitudes are constant, these oscillatons appear difficult to explain otherwise. This problem will put the network in a position that most likely represents a local minimum. Therefore, gradient descent will not allow for resolving the situation. To ensure oscillator frequencies cannot cross and to further simplify the parameter space by means of removing permutation ambiguity, we introduce two further constraints:

- frequency decay time τ and frequency decay step δ are shared by all oscillators.
- start and stop frequencies are ordered so that the first frequency is always the lowest.

Sharing the same frequency decay step for all oscillators can be expressed as $f^i_{start} = (1+\delta) f^i_{stop}$ with $\delta \geq 0$ for the i-th oscillator. Such constraint will keep the frequency ratio relationship between the oscillators. This is reasonable from a physical standpoint, as it

follows the observations from [Fletcher and Bassett, 1978], where the frequency variation ratio was found to be the same for all the partials. The same can be said about the frequency decay τ. Furthermore, by having such a simplified model, we are avoiding crossing between the oscillators. In other words, eq. (4.19) would be simplified to:

$$f(t, \delta, f_{stop}, \tau) = f_{stop}\left(1 + \delta \exp\left(-\frac{t}{\tau}\right)\right) \tag{4.23}$$

where δ and τ are shared among all oscillators.

Moreover, it is essential to highlight the importance of ensuring frequency ordering, as it notably simplifies the generator's task. The generator can more easily generate appropriate amplitudes for each frequency component by organizing the frequencies in a specific order, such as from low to high, the lowest having generally the highest amplitude. When using multiple oscillators, we use the cumulative sum of the frequencies to ensure the frequencies are arranged in ascending order. In other words, for a set of positive numbers f_1 to f_n, the cumulative sums give the following:

$$(f_1, \cdots, f_n) \rightarrow (f_1, f_1 + f_2, \cdots, \sum_{i=1}^{n} f_i) \tag{4.24}$$

The ascending order is guaranteed since all f_i are positive (because of the ReLU).

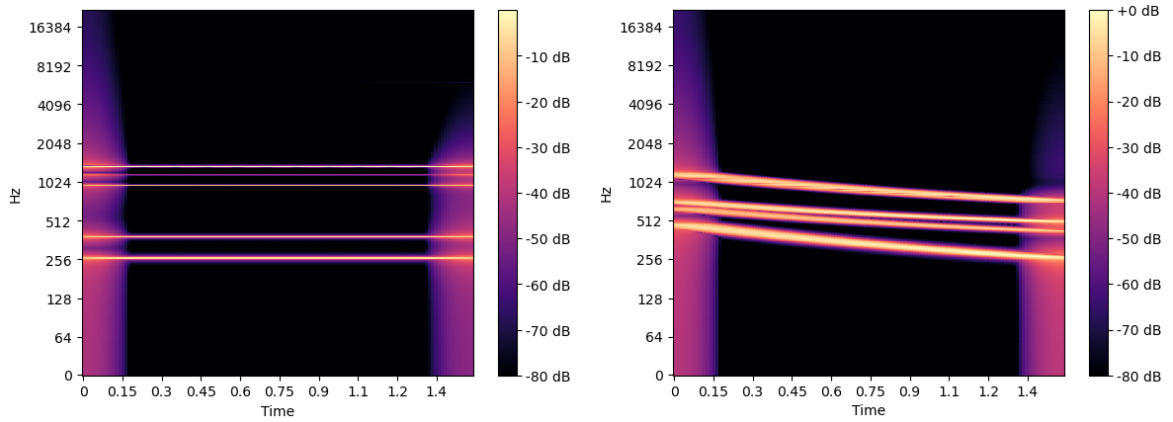
Lastly, the frequency should have an upper bound to avoid aliasing, such as not exceeding half of the sample rate. This can be enforced by clipping the values above the limit. However, the oscillator frequencies remain low in practice (below 2000Hz) for data at a 44100Hz sampling rate, which is below the Nyquist frequency of 22500Hz, making aliasing not happen in practice. To illustrate this point, we provide examples in Fig. 4.11 of the two classes that generate partials with the highest frequencies: the hi-hat. Given that the generated oscillations remained limited to lower frequency regions in all our experiments, we did not use the clipping strategy to enforce the anti-aliasing constraint.

Constant-energy representation in the discriminator

Using pre-computed envelopes at the output (as described in Sect. 4.2.2) significantly improved the FAD results for StyleWaveGAN, as seen in Sect. 4.2.5, The synthesis network works on an almost constant energy representation using these envelopes.

We hypothesize that such representation allows the synthesis network not to be biased towards some parts of the sound and facilitates the training process by reducing the time-dependent impact of the decay. Given the significant effect of such representation with the synthesis network, we hypothesize using it in the discriminator would bring low-level information up in volume, allowing the discriminator to provide better information to the generator. This is effectively a dynamic range compression technique.

There are two possible approaches to incorporating a "constant-energy" gain compensation in the discriminator. We either take the estimated energy envelope of the signal and inverse it, or we use the pre-computed average envelopes. By using the envelopes, we can force the discriminator to only work on transformed sounds following the constant-energy gain compensation described above. Since it is the actual internal representation of the



(a) Output of the oscillator bank for closed hi-hat.

(b) Output of the oscillator bank for an open hi-hat.

Figure 4.11: Examples of the oscillator bank output for closed and open hi-hat. The number of oscillators is 8, but the spectrograms show fewer partials, as some frequencies are extremely close to each other and can't be discriminated (the difference is < 0.001Hz in this case). Note these spectrograms were obtained with a version of SWG-OSC with the frequency ordering but different decay times. This result holds for the single decay version as well.

generator, making both networks work in the same compensated domain should benefit the overall generation process in terms of quality.

In any case, the gain compensation can be expressed as:

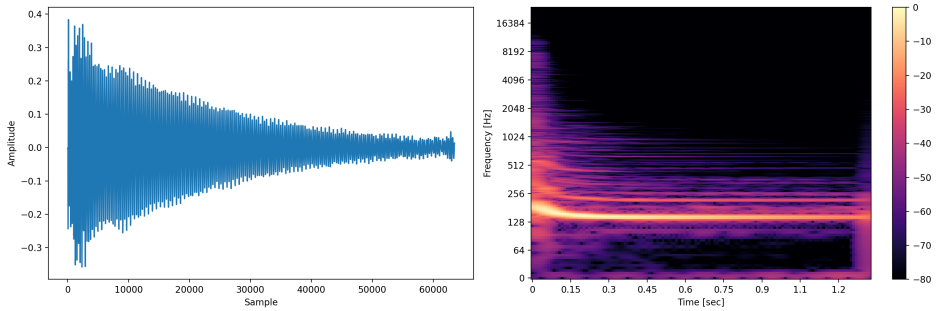
$$\hat{x} = \frac{x}{e} \tag{4.25}$$

In this context, $\hat{x}$ represents the transformed signal, x represents the original signal, and e represents the envelope selected from those described in Sect. 4.2.2, depending on the class label of x, although we could have opted for a more accurate envelope estimation using a windowed energy method. However, a challenge in using on-the-fly energy calculation involves introducing additional parameters for envelope generation to the discriminator. As we set out to modify the discriminator as minimally as possible, using pre-computed envelopes helps alleviate the parameter issue. Still, it comes at the cost of compromising the fit of the data. Figure 4.12 illustrates the effect of gain compensation using the pre-computed envelopes.

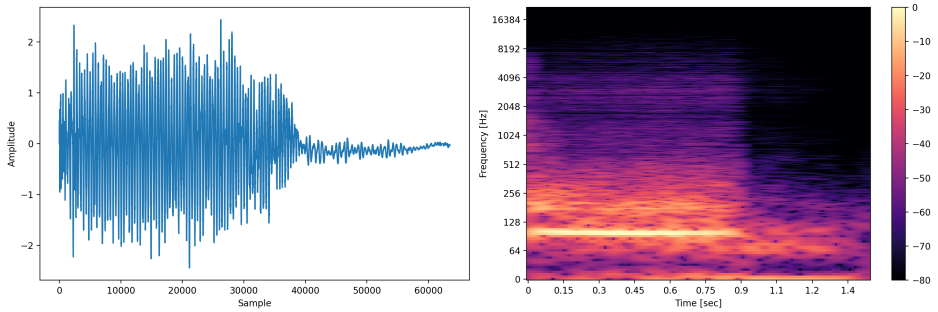
Label-dependent Style

We will introduce here a modified version of the style mechanism that establishes an explicit dependence on the class label.

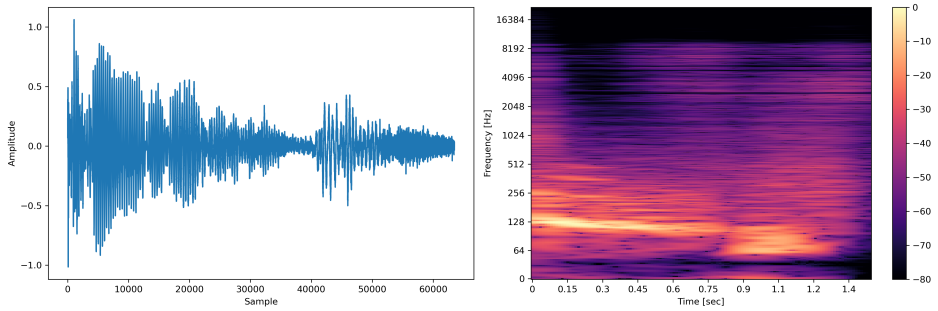
The style mechanism employed in StyleGAN and StyleWaveGAN, as described in Sect. 2.2.3, utilizes the AdaIN method to regulate the contributions of feature maps in the generation process (c.f. eq. (2.14)). However, this mechanism lacks explicit label dependency. Indeed, when considering regular StyleWaveGAN, the intermediary vector w already contains the



(a) Real tom sound (Yamaha Stage Custom, 10", Apeira Drums)



(b) Sound generated by original StyleWaveGAN



(c) Sound generated by StyleWaveGAN with oscillators

Figure 4.12: Examples of waveforms and spectrograms of real and synthesized sounds by StyleWaveGAN trained on ENST-AUG, with the constant-energy transform applied

one-hot encoded label, but there is no guarantee the network will use this information as part of the style mechanism as there is no constraint on this. Also, this entangles the label information with the intermediary representation w.

To address this problem and create an explicit dependency on the class label, we increase the dimension of the affine transform. Let us consider a case with n labels. Instead of producing a vector of dimension $2N$, the first N being for the linear transformation and the second part being the bias, we generate a style y of dimension $n2N$ to have $2N$ per class (a linear transform and a bias for each class label). In that case, the label dependency of the style is explicit.

In this scenario, we have:

$$y = (y_{c_i})_{1 \leq i \leq n} \tag{4.26}$$

where each y_{c_i} is of size $2N$.

Let us consider a mapping from the intermediate latent space w to the label-dependent

style, denoted as y_c. Consequently, the AdaIN transform becomes:

$$AdaIN(x_i, \underbrace{y_c}_{=(y_{s,c}, y_{b,c})}) = y_{s,c_i} \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,c_i} \tag{4.27}$$

where $y_c = (y_{s,c}, y_{b,c})$ is the label-specific style.

We will employ this mechanism with the oscillators to acquire distinct value sets depending on the label, while the intermediary representation w remains unchanged.

4.4.3 Objective evaluation

The goals of our experiments are the following:

- Measuring the difference in FAD compared to the original StyleWaveGAN
- Checking if the network uses the oscillators effectively, with or without labels.
- Getting insights on the internal representation of the generator and its effect on the discriminator.

To answer these different points, we will consider the following setups

- a StyleWaveGAN baseline, without the oscillators
- two versions of StyeWaveGAN with oscillators, with and without label-dependent style control
- A version of one of the networks with constant-energy gain compensation

The FAD comparison between the configurations would be a good indication of the overall contribution of the oscillators and gain compensation for the generation quality. To further observe if the oscillators are used effectively, we will have to study some spectrograms of generated sounds and compare the presence of clear sinusoids or not.

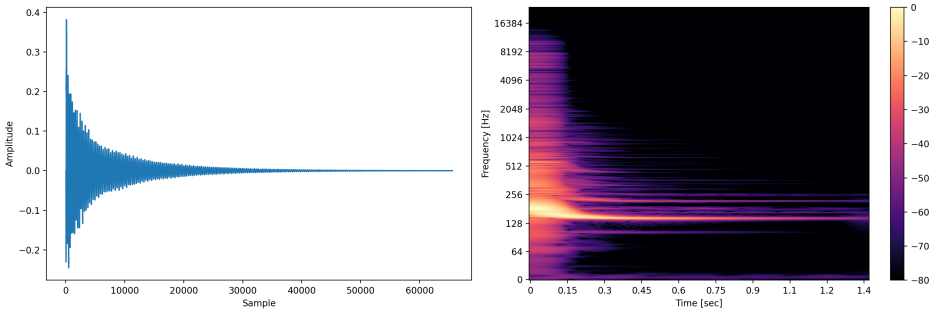
All figures for this subsection use the best version of the network we could obtain. In all experiments with oscillator banks, the affine transform has its parameters initialized following a Gaussian distribution $\mathcal{N}(0,\ 1)$.

Observations on the harmonic contents of the generated drum sounds for SWG-OSC

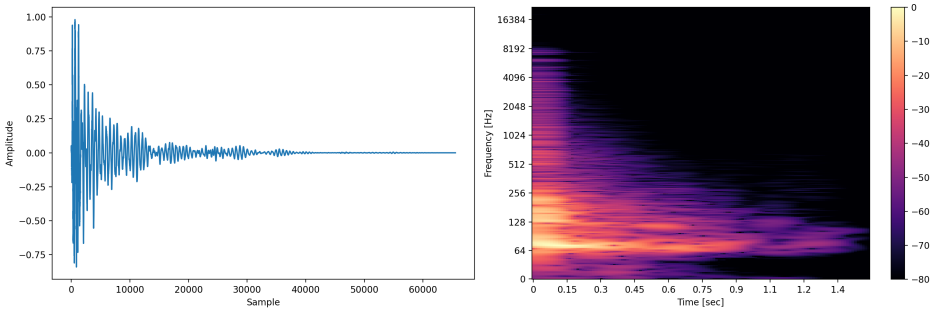
We can observe Fig. 4.13 to answer the point about the effective use of the oscillator bank. We can observe the thin horizontal lines characteristic of clear sine waves on the spectrogram of Fig. 4.13c. Compared to Fig. 4.13b, the original StyleWaveGAN sample is noisier and does not present strong harmonic content. Regarding waveforms, StyleWaveGAN with oscillators shows a shorter decay time than the non-oscillator StyleWaveGAN.

Finally, compared to an actual tom recording in Fig. 4.13a, the oscillator-based sample shows inconsistencies in terms of the decay of the harmonics. Similarly, the decay is not as smooth in Fig. 4.13b and 4.13c as in Fig. 4.13a. It should be noted that the oscillator-based StyleWaveGAN shows a shorter decay than the oscillator-less version.

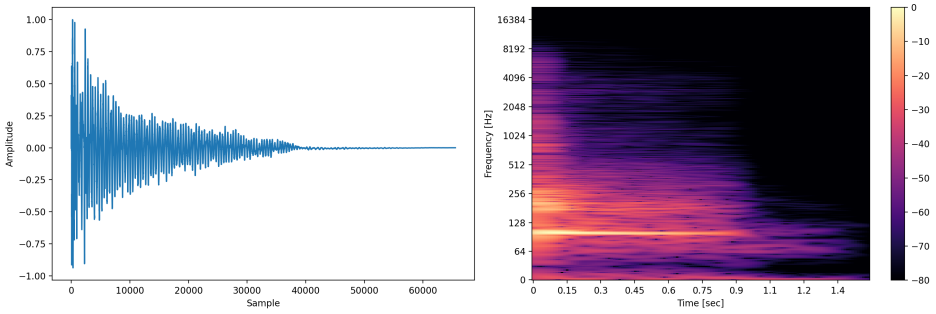
Where the real samples have high frequencies and harmonics decaying rapidly (≈0.2s), the synthesized samples have inconsistent decays of the harmonics, especially with a longer decay for the higher frequencies compared to the real and oscillator-less samples. This can be explained by the network trying to add high-order partials through the noise and convolution layers instead of the oscillator bank, which is limited to 8 oscillators in our experiment.



(a) Real tom sound (Yamaha Stage Custom, 10", Apeira Drums)



(b) Sound generated by original StyleWaveGAN



(c) Sound generated by StyleWaveGAN with oscillators

Figure 4.13: Examples of waveforms and spectrograms of real and synthesized tom sounds with StyleWaveGAN trained on ENST-AUG

Measurements with Fréchet Audio Distance for SWG-OSC

In terms of FAD, we have several networks to compare. In Table 4.17, we use the following abbreviations to describe the tested variations of StyleWaveGAN:

SWG-OG: The "original" StyleWaveGAN, as seen in Sect. 4.4

SWG-OSC: StyleWaveGAN with an oscillator bank

SWG-OSC-PL: StyleWaveGAN with per-label oscillator bank, as described in Sect. 4.4.2

SWG-EDISC: StyleWaveGAN with per-label oscillator bank and constant-energy discriminator

Dataset \ Model	SWG-OG	SWG-OSC	SWG-OSC-PL	SWG-EDISC
Full dataset	3.62	4.21	**3.60**	8.02
Tom	6.27	6.13	**5.75**	17.56
Closed Hi-hat	4.23	3.7	5.57	**3.04**
Open Hi-hat	**4.12**	5.13	5.09	8.12
Snare	4.29	6.09	**3.08**	4.31
Kick	**3.58**	3.86	4.13	5.60

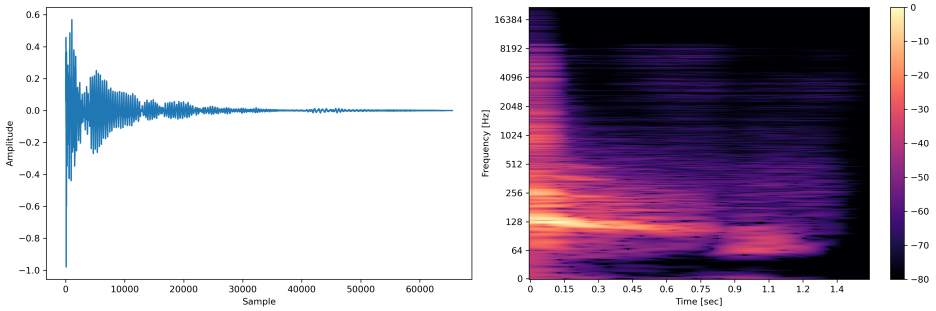
Table 4.17: FAD for tested variations of StyleWaveGAN on ENST-Drums (lower is better)

The results compiled in Tabl. 4.17 are obtained by computing the FAD on 64000 generated samples. When a specific drum or cymbal is mentioned in the table, only samples with the correct label are generated. The number of tested samples remains at 64000.

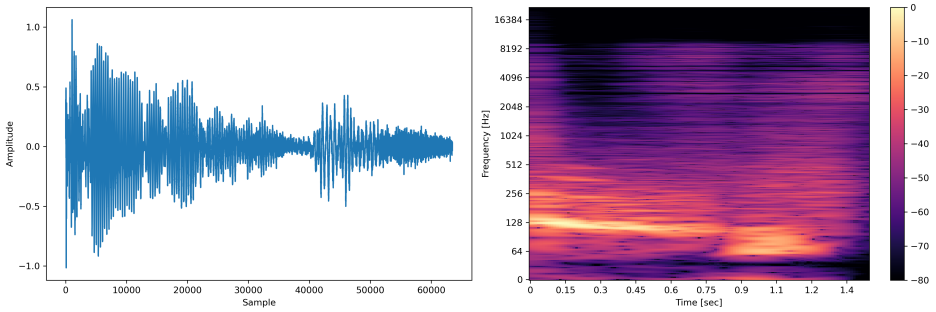
We slightly improved FAD compared to the original StyleWaveGAN using a per-label oscillator bank (SWG-OSC-PL). More importantly, the performance on the toms and snare is quite improved, but we have to concede worse results for the cymbals. This partial degradation in measured FAD is evidence that adding the oscillators helps the generation of resonant drums like toms and snares, but hinders the synthesis of the cymbals due to their chaotic behavior. However, the network did not turn down the oscillator amplitude to zero, which makes us hypothesize the network uses the oscillators but not in a perceivable or perceptually relevant way. Concerning the shells, informal listening tests showed improved resonances on toms and a noticeable pitch-bend effect, typical of such drums. For snare and kick drums, the generated sounds appeared to have the resonances found lacking during the subjective evaluation in Sect. 4.2.5, but the overall sound quality was slightly lower.

Effect of the constant-energy representation

The other notable result comes from the constant-energy discriminator (SWG-EDISC). First, let us observe the influence of the transform on drum sounds, especially the toms. Figure 4.12 shows the results when using a sample from one of the datasets of real drum sounds, a sample from the original SWG, and a sample from the per-oscillator variation. The first observation is the reduced dynamic range, which we expected from such a transformation. In particular, for Fig. 4.12c, this is the output of the generator network before the envelope. This figure shows a low-frequency beating phenomenon effectively perceptible by a human listener, as observed in informal listening tests. Such low-frequency artifact is most likely due to problems during the up-sampling process in the first layers of the generator.



(a) Tom sample by StyleWaveGAN trained with a constant-energy discriminator, without constant energy compensation applied



(b) Tom sample with constant energy transformation applied

Figure 4.14: Tom sample by StyleWaveGAN trained with a constant-energy discriminator, with constant energy compensation applied at the input

From Fig. 4.14a and 4.14b, we can see the samples are considerably noisier than their counterpart from Fig. 4.12c and 4.13c. Also, the beating phenomenon observed above is even more pronounced here. This beating is notably apparent in the transformed version Fig. 4.14b.

If the discriminator functions correctly, the generator is not motivated to introduce additional noise since all relationships remain constant. The fact the generated sounds are far noisier may suggest that the discriminator is not powerful enough for this application and can be fooled by a noisy generation.

While we must recognize that the discriminator needs improvement, addressing this issue within the thesis was not feasible given the time constraints.

4.5 Velocity control

4.5.1 Introduction

Although we now have significant command over the timbral aspects of the generated output, as measured by objective and subjective criteria, as shown in Sect. 4.3, StyleWaveGAN is not yet suitable for musical production or performance. One of the most fundamental attributes of musical performance is dynamics, which includes the varying hit velocity applied to the diverse components of a drum kit. Along with the capacity to control timbral properties, StyleWaveGAN requires dynamics control to ensure its

viability in a music production setting.

Velocity estimation is common for automatic transcription tasks [Gillick et al., 2019; Callender et al., 2020], but these datasets are unsuitable for one-shot drum synthesis. These datasets comprise full performances on electronic drum kits, taking the form of MIDI files and stereo output of the performance. In this case, there is no separation between the sound sources (i.e., the different drums and cymbals), as only one sound file of the whole drum kit per performance can be found. Moreover, even if the sound sources were separable, the electronic drum sound module used to record the performances would lack the variability of real performances on real drums. On the other hand, no one-shot sound database is available with MIDI velocity annotation, especially with real drum sounds. Given the non-existence of adequate datasets, we created the Apeira Drums dataset, as described in Sect. 3.4.

To add dynamics control to StyleWaveGAN, we could leverage the prior knowledge from timbral controls Sect. 4.3. This means we would consider a velocity target at the input, just like the timbral features, and we would then use a penalty loss between generated and real samples using an external differentiable estimator. In this section, we will study two methods for velocity estimation: one using neural networks and another using a differentiable signal descriptor based on energy. More precisely, we will use the global dynamics indication used during the recording of Apeira Drums (Sect. 3.4) for the neural velocity descriptor and the velocity estimation obtained from the Senstroke as a ground truth also found in Apeira Drums for the signal-based descriptor.

4.5.2 Neural Velocity Estimator

Description of the Neural Velocity Descriptor

Contrary to timbral features, there is no absolute and well-defined metric for velocity. Moreover, velocity differs from gain, making it even harder to create a data-agnostic feature. It is important to understand that, in this context, the term "data-agnostic" refers to an estimator unaffected by the recording conditions, particularly the recording gain. For instance, a sound from ENST-Drums and one from Apeira Drums might have the same peak value/energy but represent different dynamic levels (due to different preamplifier gain and recording conditions).

We can leverage neural networks' capabilities and expressive power here. Instead of crafting a signal-based descriptor like [Peeters, 2004; Pearce et al., 2016], we can automatically utilize a neural network to generate the descriptor. In our situation, this means the neural velocity descriptor is trained before the synthesizer.

This neural velocity descriptor will have as its objective to minimize the mean square error between the output of the network and the dynamic level. More precisely, it aims to estimate the dynamic level as a dynamics annotation. Since we have 4 dynamic levels in the Apeira Drum dataset, we can build an estimator using this information as the target while having a peak-normalized input. In our case, 1 for pianissimo to 4 for fortississimo.

During training, we added a random noise taken from a Gaussian distribution with a standard deviation of 0.17 to the label to avoid over-fitting as a form of soft labeling.

Choosing such a standard deviation allows us to generate 99% of the randomized values between $\approx \pm\frac{1}{2}$ through numerical calculation with the Gaussian distribution. The same neural velocity estimator without soft labeling is also trained for comparison purposes, and we will refer to it as the "discrete label" velocity estimator. Finally, a version of the neural velocity estimator without any input normalization is trained to measure the impact of the peak normalization.

Using the mean squared error (MSE) as the training loss should allow us to build Gaussian models around the target values, retaining a notion of similarity that we would lose in a pure discrete classification context. We hypothesize that doing so will help with the interpolation when used to control a neural synthesizer.

Regarding implementation and training, we use a random 90/10 dataset shuffle split to measure the mean square error on a validation set. The peak normalization should prevent a dependency between the peak level of the input and the dynamic level, contributing to the descriptor's data agnosticism.

The network is a convolutional neural network (CNN) whose input is a peak-normalized version of the first 250ms of the input sound samples. We then transform the audio input into a Mel-spectrogram before going through the different layers. This is motivated by [Bous and Roebel, 2023], where the authors provided a method to disentangle the Mel spectrogram from its level on human voice signals and also provided a voice level estimator using this representation. Another argument in favor of the Mel spectrogram is the compactness of its representation, which is expected to lead to small and efficient models, making it a viable candidate for a descriptor.

The network has 6 1D convolutional layers :

- Layer 1: filter size 5, 220 filters
- Layers 2-3: filter size 5, 200 filters
- Layers 4-5: filter size 3, 100 filters
- Layer 6: filter size 3, 50 filters

For this task, we trained the network on 8192 epochs with a batch size of 32. The optimizer used here is Adam [Kingma and Ba, 2015].

Experimental Results of the Neural Velocity Estimators on Apeira Drums

This section presents the performance of neural velocity estimators based on the velocity information found in Apeira Drums.

We must create this baseline, since we have no baseline or state-of-the-art to compare ourselves. We will reuse the Apeira Drums dataset (AD) for this evaluation experiment and check if the neural velocity estimator has good generalization capabilities.

Neural Velocity Descriptor on Apeira Drums The first measurement of the quality of our different configurations of neural velocity estimators is a measure of their discriminatory power. We expect our estimators to, at least, predict the subjective velocity instruction given to the drummer.

To measure the capacity of the velocity estimator to work well in most cases, we use the validation set used during training to evaluate the network's performance. We then do the validation process against the true values, i.e., those unaffected by the soft-labeling process.

Method \ Dynamic	*pp*	*mp*	*f*	*fff*
Shells, soft-labeling (SL)	0.06	0.08	0.07	0.10
Shells, Soft-labeling and no normalization (SLNN)	0.08	0.10	0.07	0.12
Shells, Discrete values (DV)	0.02	0.07	0.03	0.08
Cymbals (SL)	0.21	0.23	0.13	0.24
Cymbals (SLNN)	0.23	0.24	0.12	0.24
Cymbals (DV)	0.15	0.12	0.13	0.31
Cymbals + Shell (SL)	0.11	0.11	0.08	0.21
Cymbals + Shell (SLNN)	0.10	0.11	0.11	0.28
Cymbals + Shell (DV)	0.11	0.18	0.08	0.15

Table 4.18: Root mean squared error of the neural velocity estimator (validation split, close mics only, lower is better)

Method \ Dynamic	*pp*	*mp*	*f*	*fff*
Shells (SL)	1.00	0.99	1.00	0.99
Shells (DV)	1.00	0.99	1.00	0.99
Cymbals (SL)	0.95	0.93	0.99	0.93
Cymbals (DV)	0.97	0.98	0.98	0.92
Cymbals + Shell (SL)	0.98	0.98	0.99	0.96
Cymbals + Shell (DV)	0.97	0.98	0.99	0.97

Table 4.19: Accuracy of the neural velocity estimator (validation split, close mics only, higher is better)

The results of the validation split of the training dataset are acceptable. The Root Mean Squared Errors (RMSE) shown in Tabl. 4.18 for the soft-label networks are close to the ones obtained with discrete ones. The version where the input was not peak normalized (as defined in eq. (3.1)) shows slightly worse RMSE than other versions, especially at higher levels. The worse performance of the non-normalized version leads us to discard it for other measurements, shown in Tab. 4.19 to 4.21.

We can define an accuracy metric by using rounding of the output. We will consider the prediction accurate if the rounded output equals the target. We show the results for this metric in Table 4.19. The accuracy is overall greater than 95% for all cases. With such performance, the network can safely be added to control the velocity of a neural synthesizer, as long as the training data is used to train the synthesizer.

Method \ Dynamic	*pp*	*mp*	*f*	*fff*
Shells (SL)	1.16	0.62	0.77	0.90
Shells (DV)	1.23	0.60	0.73	0.96
Cymbals (SL)	1.38	0.67	0.24	0.83
Cymbals (DV)	0.81	0.34	0.58	1.29
Cymbals + Shell (SL)	1.02	0.69	0.72	0.94
Cymbals + Shell (DV)	1.24	0.59	0.52	0.91

Table 4.20: Root mean squared error of the neural velocity estimator on private test dataset (lower is better)

Method \ Dynamic	*pp*	*mp*	*f*	*fff*
Shells (SL)	0.30	0.74	0.41	0.59
Shells (DV)	0.14	0.81	0.45	0.50
Cymbals (SL)	0.00	0.42	0.95	0.18
Cymbals (DV)	0.07	0.82	0.60	0.17
Cymbals + Shell (SL)	0.56	0.56	0.56	0.46
Cymbals + Shell (DV)	0.17	0.66	0.65	0.37

Table 4.21: Accuracy of the neural velocity estimator on private test dataset (higher is better)

Generalization Capabilities The same evaluation was conducted on a private dataset of drum sounds recorded with the same protocol as the Apeira Drums to further the validation split results and explore the estimation network's generalization capabilities. This dataset was recorded with the same microphones and drummer but with different drums and cymbals. From Tab. 4.20 and 4.21, we can see that the proposed estimation network has difficulties generalizing to the test data while performing well on the validation split. The first hypothesis would be an overfitted neural velocity estimator on the training set. This would take the form of high accuracy and low MSE on the training set but results similar to Tab. 4.20 and 4.21. And it is indeed the case, as accuracy and MSE on Tab. 4.18 and 4.19, are above 90% and less than 0.31 respectively when it drops below 50% in general and with MSE up to 1.38 while Tab. 4.18 and 4.19 show an example of overtraining. However, it should be noted the metrics are not uniformly worse and that the most impacted dynamics levels are the extremes (i.e., *pp* and *fff*).

Given the recording setups were the same, as per Apeira Drums recording protocol, another hypothesis would be that the drummer did not manage to keep the hit dynamics coherent over the few weeks the recording session took place. To study this hypothesis, we could use the trigger sensor data. Assuming that the energy of the trigger sensor correlates with the hit strength, we could compare the mean and variance of trigger energy between the proposed dataset and the test data. The energy of the trigger signal is computed on a 682ms frame by summing the square of the signal on said frame.

Results are compiled in Tabl. 4.22. There is a noticeable difference in the energy levels of the trigger signals between the two datasets (training and test). This raises the question

Model	Mean	Standard deviation
Training, *pp*	0.56	0.47
Test, *pp*	1.21	3.92
Training, *mp*	13.30	10.44
Test, *mp*	7.07	4.16
Training, *f*	38.07	25.75
Test, *f*	32.95	12.46
Training, *fff*	167.81	71.11
Test, *fff*	172.45	54.36

Table 4.22: Mean and standard deviation of signal energy on trigger sensors for snare drums on different datasets (assuming normal distribution)

whether the velocity estimator would be better trained using trigger energy instead of raw waveforms. However, it is important to consider that piezoelectric triggers can experience physical saturation at higher velocities, which appear as clipping on the recorded signals, as shown in Fig. 3.4d. While using the piezoelectric triggers might be a dead end, using energy to estimate velocity may be a promising direction for further exploration.

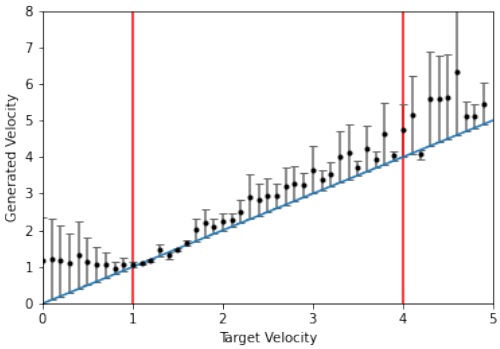
Application to StyleWaveGAN of Neural Velocity control

Class \ Dynamic	*pp*	*mp*	*f*	*fff*
Kick	0.35	0.43	0.07	0.15
Snare	0.01	0.08	0.27	0.32
Tom	0.05	0.00	0.03	0.21
Closed HH	0.02	0.01	0.03	0.08
Half-opened HH	3.96	1.01	0.02	0.15
Open HH	0.21	0.06	0.06	0.11
Crash	0.10	0.06	0.10	0.10
Ride	0.11	0.07	0.06	0.14
Special	0.14	0.09	0.04	0.18

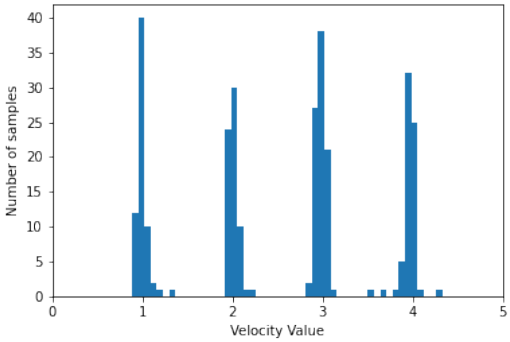
Table 4.23: Mean squared error computed using the estimated velocity of samples generated by StyleWaveGAN with velocity control (lower is better)

This subsection presents the results of integrating a velocity control with StyleWaveGAN. This study uses a single estimator for all shells and cymbals. We used the soft-label estimator, as its RMSE is close to the discrete version shown in Tabl. 4.18. More importantly, its accuracy is more consistent than the discrete version, as shown in Tab. 4.19 and 4.21.

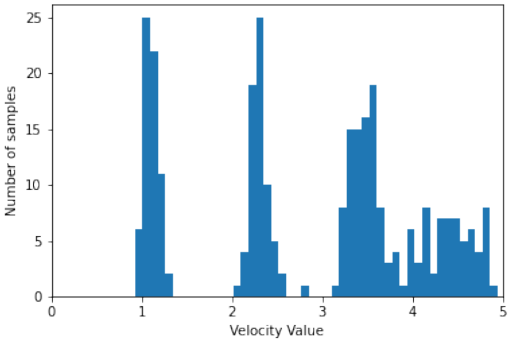
To establish a loss for the velocity control, we pass the generated samples through the estimator, calculate the MSE between the estimated velocity and the control input, and use the result as the velocity control loss. Note that the velocity model used for providing the velocity loss when training the synthesizer has been trained on the same training dataset. Therefore, the generalization problem revealed with the test dataset in Sect. 4.5.2 should not pose a problem here. We used the non-discrete velocity model in our experiment.



(a) Target velocity vs. estimated velocity for snare drum synthesis with StyleWaveGAN



(b) Histogram of estimated velocity values from snare drum samples taken from the training dataset



(c) Histogram of estimated velocity values for generated snare drum samples (target values from the training dataset)

Figure 4.15: Histogram of target and estimated velocity values for generated snare drum samples using the neural velocity descriptor

Table 4.23 shows the MSE computed between batches of generated samples with set descriptor values (at 1,2,3 and 4 to represent the different dynamic levels). First, we can conclude that the velocity control error remains significantly smaller than the velocity step for nearly all instruments and all velocity levels. There is an exception with a significant error with the half-opened hi-hat for *pp* and *mp* velocities. A second problem is the *pp* and *mp* velocities of the kick. Finally, we note a general tendency to increase control error, especially for the highest dynamics level (*fff*). This increase is particularly marked for the snare and tom drums. We can further study the case of the snare drum. From Fig. 4.15a, we can see that the network functions somewhat correctly regarding control accuracy. In this figure are plotted the error bars for a set of target velocities in black, a blue line representing the identity function, and vertical red lines showing the limit of the target values for the dataset. The error around the target values and the average increases for higher dynamic levels. The synthesizer tends to generate samples with higher dynamic levels than requested for the snare drum. Figures 4.15b and 4.15c show the difference in terms of distribution between computed velocity values on the snare drum subset and the values calculated on the generated snare drum samples with these values as targets. The distribution for the lowest dynamic levels is similar but slightly offset towards higher values. We can, however, note the deviation and offset increase at higher dynamic levels.

This offset may be partly due to the higher-level outlier values, which we can see in Fig. 4.15b between target values of 3 and 4.

4.5.3 Signal-based Velocity Descriptor

Following the results we showed in Sect. 4.3 of the differentiable descriptors with StyleWaveGAN compared to neural estimation as in DrumGAN or NeuroDrum, we can expect good performance from signal-based descriptors of velocity.

Contrary to the neural descriptor, we do not aim to provide a data-agnostic velocity descriptor. However, we will rely on the fact that we recorded the Apeira Drums dataset to provide equal recording gain for all samples and soft-label on velocity using dynamics indications. The special and robust recording protocol used for Apeira Drums is essential here, as we can use the energy of the recorded sounds without risking errors due to mismatched gains and recording protocol.

We will use the following energy estimation formula as our velocity feature:

$$P(x) = \sum_{n} x[n]^2 \tag{4.28}$$

where $x[n]$ is the n-th element of a sampled signal x.

While we could employ a more complex descriptor using, for instance, filtered parts of the signal, such a simple descriptor should provide enough information on the viability of using the energy as a velocity descriptor and the viability of a signal-based descriptor compared to a neural one.

Velocity information is part of a standard MIDI message, as it is a fundamental component of the MIDI specification. This linkage is particularly relevant in the context of automatic transcription datasets, such as those used for drums and piano transcription [Hawthorne et al., 2019; Gillick et al., 2019; Callender et al., 2020]. However, these datasets are limited to one instrument [Hawthorne et al., 2019] or use sample-based synthesizers as their output [Gillick et al., 2019; Callender et al., 2020], which limits either the variety (one instrument) or the expressivity (by limits of samplers).

This section will show how we can leverage the data available in Apeira Drums, especially the MIDI velocity estimate from the Senstroke sensor, to create a general velocity feature. A general velocity feature compatible with MIDI velocity would be important for our drum synthesizer. First, such a velocity feature would make StyleWaveGAN velocity-aware. Second, it would make the integration with MIDI controllers easy, hence increasing the usability of StyleWaveGAN in a music production context.

Energy and Senstroke MIDI velocity

The Senstroke sensor, whose firmware was developed by Apeira Technologies and used during the cymbal recordings of Apeira Drums, outputs a velocity value proportional to the angular velocity at the moment of impact, as described in eq. (3.2). Please note

that the outputted value is a floating-point number, contrary to the expectations of the MIDI standard, which expects an integer between 0 and 127. The scaling for the velocity estimation was done so that a hit on a Roland electronic drum kit outputs a similar MIDI velocity as the accelerometer. Even if such measurements were informal and are highly dependent on the electronic drum used, the commercial success of the Senstroke is enough of an argument to justify a good fit between its output and MIDI velocities, as described in Sect. 3.4.

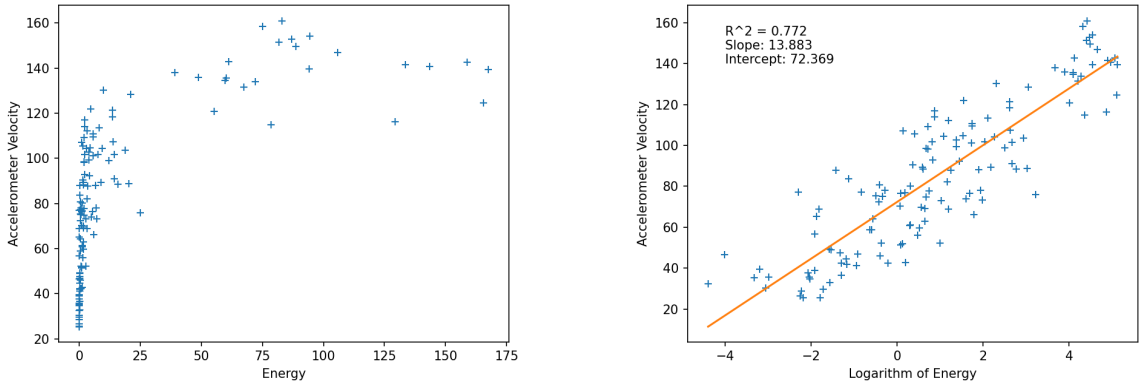
Our first task here will be to compare the energy of the cymbal recording to the estimated MIDI velocity, as shown in Fig. 4.16a. We know that the cymbal microphone was placed at 1m of the recorded cymbal and pointed at the cymbal's bell (which corresponds to the mounting point of the cymbal stand) with an angle of 30 degrees. We measured the distance with a laser telemeter and fixed the angle for the recording process. In other words, the recording conditions are repeatable.

From the observation of Fig. 4.16a, we can expect a non-linear relationship between angular velocity and energy that appears to be logarithmic. To this end, we also studied the relationship between the logarithm of energy and estimated velocity, as shown in Fig. 4.16b.

More precisely, we are testing the following model:

$$\hat{V}(s) = a \log(P(s)) + b \tag{4.29}$$

where $\hat{V}$ is the estimated MIDI velocity of the signal s, $P(s)$ is the energy of the signal s and a, b are coefficients obtained through a linear regression.



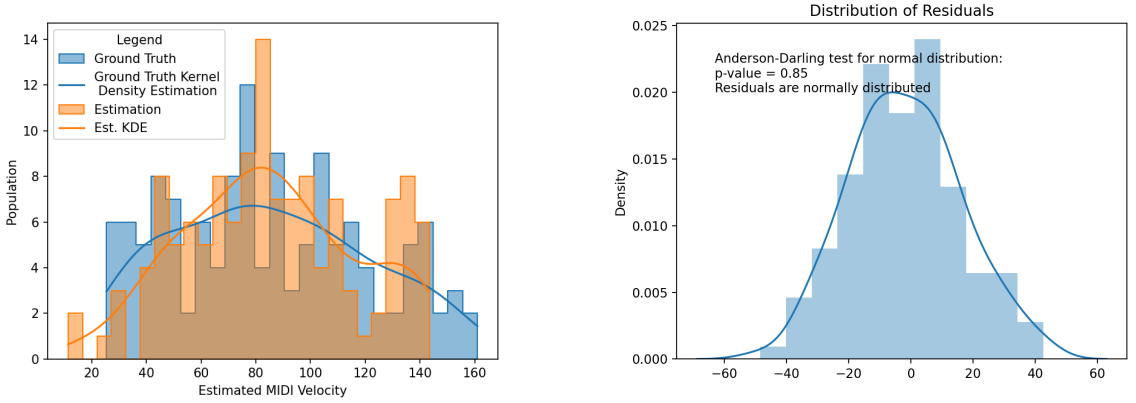
(a) Senstroke MIDI Velocity Estimation vs. Energy ((eq. (4.28))) for Cymbals

(b) Senstroke MIDI Velocity Estimation vs. log-Energy (eq. (4.29)) for Cymbals

Figure 4.16: Comparison of Senstroke MIDI Velocity Estimation vs. Energy Estimation for Cymbals from dataset AD-SS

Upon analyzing Fig. 4.16a and 4.16b, we can first observe that the velocity estimated with the accelerometer exceeds the maximum value of 127 permitted by the MIDI 1.0 standard. Please note that the values can be scaled and rounded to fit within the permitted MIDI range *a posteriori* to address this issue.

We can further study this model by working on the model's residuals and checking the estimated probability distribution.



(a) Distributions of real and estimated velocities on AD-SS

(b) Histogram of residuals from the model estimation on the dataset AD-SS

Figure 4.17: Histogram and Residuals for the log-energy to velocity estimator

The results from the inverted model are shown in Fig. 4.17a. The blue histogram is the ground truth, and the orange one is the histogram of the MIDI velocities estimated from log-energy. The smooth lines with colors corresponding to the histogram bars are the Kernel Density Estimation on the data [Rosenblatt, 1956; Parzen, 1962], with the kernel bandwidth obtained using Scott's rule [Scott, 1992]. We can learn from this figure that the velocity estimator has an acceptable performance.

To ensure that our estimator follows the normality of the error hypothesis of linear regression, we also provide Fig. 4.17b. In this figure, we plot the histogram of residuals and check its normality with the Anderson–Darling test [Anderson and Darling, 1952]. It tests the null hypothesis that a sample is drawn from a population that follows a normal distribution with unknown mean and variance. In this case, the normality of the residuals is not rejected by the test with a confidence level of 95%, since the p-value is 0.85.

Since the model is derived from the model described eq. (4.29), we have a guarantee that the ordering is correct so that we can check the distributions directly. We can then utilize the Kolmogorov-Smirnov test as an additional metric to test the hypothesis that the descriptors' distribution conforms to the empirical velocity distribution. In our analysis, we set a confidence level of 95%, meaning that we will only reject the null hypothesis in favor of the alternative if the p-value is less than 0.05. Upon performing the Kolmogorov-Smirnov test, we obtained a p-value of 0.54, above our threshold of 0.05. As a result, we cannot reject the null hypothesis, indicating that the two distributions are a good fit for one another.

We have established a linear relationship between equivalent MIDI velocity and the logarithm of the energy. We have determined that energy is suitable for velocity estimation, with a minimal skew in the estimation.

In turn, this means we can use the logarithm of the energy of a signal to obtain an estimate of the MIDI velocity. These findings are noteworthy and will drive our future work in this area.

Application to StyleWaveGAN of Signal-Based Velocity Estimator

To incite StyleWaveGAN to learn dynamics control, we will use the log-energy as an additional input feature, in the same way as the timbral descriptors were integrated, as described in Sect. 4.3. We choose to use the logarithm of the energy as the velocity feature instead of the MIDI velocity. The idea here is to use the energy as the global descriptor for any drum or cymbal, but afterward use an instrument-dependent scaling to obtain the MIDI velocity. Since the model described in eq. (4.29) is invertible, it can go from velocity to energy. This means we can move from a MIDI velocity to a target energy. The network is trained in the same configuration as Sect. 4.3, i.e., supervised StyleWaveGAN with the additional velocity feature. The feature is integrated through the loss function described in eq. (4.9). In this case, $F_T = V$ described in eq. (4.29).

To illustrate how this would be implemented in a MIDI-compatible synthesizer, we provide Fig. 4.18. The key idea is to separate the MIDI control from the synthesizer control.

Since the energy distribution differs from instrument to instrument, having a fixed velocity scale would be a problem, as some instruments would not span the whole allowed scale. On the other hand, even if every instrument sees its equivalent MIDI velocity scaled in the range of 0 to 127, its energy would remain unchanged.

Having the MIDI velocity as an input feature is necessary from a user standpoint. This allows the end-user to manipulate well-known quantities while retaining a meaningful feature for the synthesizer. However, the network should work on the energy as it is not class-dependent, contrary to the MIDI velocity similarly to the timbral descriptors from Sect. 4.3.

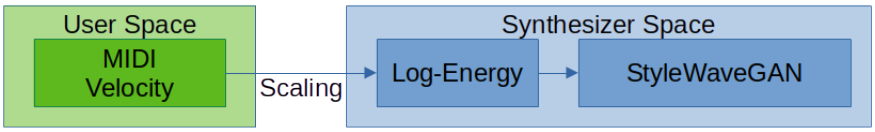


Figure 4.18: Implementation of velocity to energy scaling in StyleWaveGAN

Figure 4.19a shows the target and estimated velocity values for generated snare drum samples with the signal-based velocity estimator scaling from 0 to 127, the log-energy lying in a different interval. The scaling is possible because we showed a linear relationship between log-energy and Senstroke MIDI velocity ground truth in eq. (4.29), which can be summarized as MIDI velocity $\propto \log(P)$ with P being the energy of the signal. This applies the principle shown in Fig. 4.18.

In orange are the target values, and in blue are the estimated velocity from the generated audio samples. It should be noted that the log-energy being the target, the velocity estimation shown in Fig. 4.19a is simply an affine scaling such that the lowest energy sound in the dataset has velocity 1 and the highest energy sound in the dataset has velocity 127. The same experiment was carried out with a linear interpolation between the min value on the snare drum dataset in log-energy and +33% of the maximum value on the dataset with 250 steps. This experiment aims to study the network's capacity to handle values outside the training dataset and its interpolation capabilities. Figure 4.19b shows the results of this experiment, with the same scaling as in Fig. 4.19a.

Regarding objective results, the best-case scenario would be similar to the brightness feature control described in Sect. 4.3. In other words, the network could interpolate within the dataset's limits and be capable of extrapolating unseen combinations beyond the limits to some extent.

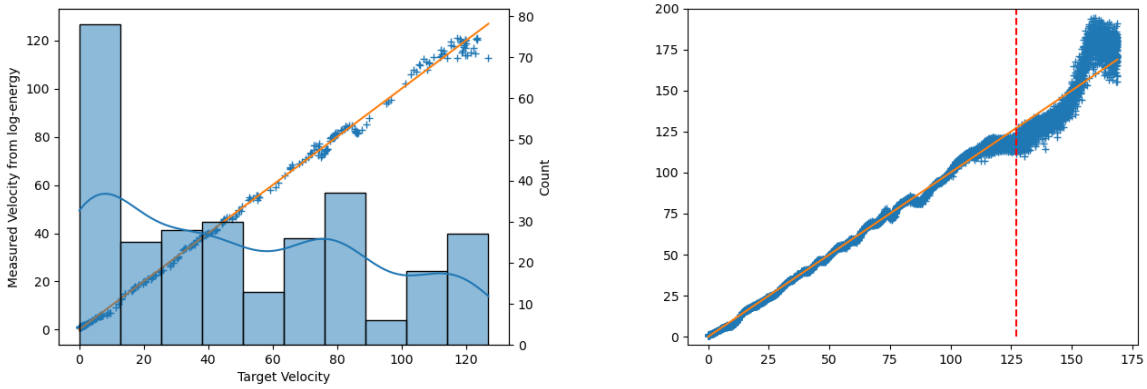
To give a more objective measurement than eyeballing a plot, we provide Tabl. 4.24 where we compute the mean absolute error and the root mean squared error.

Just like Sect. 4.3, we also provide a linearity metric by using the determination coefficient of linear regression on the data given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.30}$$

Case	Mean Absolute Error	Root Mean Squared Error	R^2
Values from dataset	1.37	2.04	0.99
Values Interpolated	1.68	2.47	0.99
Values Extrapolated	10.48	12.36	0.86

Table 4.24: Control Errors and Determination Coefficient with the signal-based velocity descriptor, given in velocity units



(a) Values with target values from the dataset. Note the axes are in velocity units, with 127 corresponding to the maximum log-energy on the dataset.

(b) Velocity using a continuum of values. Note the axes are in velocity units, with 127 corresponding to the maximum log-energy on the dataset, indicated by a dashed red line.

Figure 4.19: Velocity Control with StyleWaveGAN, using values from the dataset and a continuum of values within the dataset, snare drum only. Note the axes are in velocity units, with 127 corresponding to the maximum of log-energy on the dataset

The results are shown in Fig. 4.19a and 4.19b: these are promising and are accompanied by satisfactory metrics in Tabl. 4.24.

Note that the metrics are from the scaled log-energy to MIDI velocity. Figures 4.19a and 4.19b show a case where instead of using the energy directly for control, we scaled the energy so that the values from the dataset max out at 127, with the minimum value

from the dataset being mapped to 1. This velocity scaling means we now have a MIDI-compatible estimator for this instrument (here, snare drums in the same recording conditions) through scaling the control before the network, as described in Fig. 4.18. It is worth noting, however, that the network will continue to operate on the energy descriptor and not the MIDI-scaled version.

As for objective metrics, as shown in Tabl. 4.24, the fidelity achieved on the dataset is satisfying, with an MAE of only 1.37 MIDI units and an RMSE of 2.04 MIDI units. This implies the measure is close to its target with minimal deviation and coherent with Fig. 4.19a where the values are close to the target. The interpolated case, whose error values are found in the "Values Interpolated" row, is based on a set of values ranging from 1 to 127 with a fixed step and aims to evaluate the network's capability to interpolate between the values in the training dataset. The MAE and RMSE remain close to the case where values were taken from the dataset (1.68 vs. 1.37 and 2.47 vs. 2.04). The results are worse than the values from the dataset case but remain rather close. From this, we can safely assume the network can follow the velocity control on the velocity range from the dataset. This is coherent with Fig. 4.19b, where the interpolated case corresponds to the values at the left of the red dashed line. The overall shape of the generated values follows the one from Fig. 4.19a with good fidelity, and in fact, Fig. 4.19b appears to be simply a smoothed version of Fig. 4.19a on this range from 1 to 127.

The extrapolated case, whose error values are found in the "Values Extrapolated" row, describes the behavior when the velocity value exceeds 127. Figure 4.19b shows clearly that the values extrapolated are not as precise and close to the target, and also suffer from increased deviation. However, the values remain relatively consistent in terms of the order. This behavior is also described in Tabl. 4.24 where the MAE and RMSE significantly increase compared to the training set of the interpolated case.

In terms of linearity, the training set evaluation and the interpolated case provide excellent determination coefficients (0.99 for both), which means most of the behavior can be explained by a linear model. While not as good with an R^2 of 0.86, the extrapolated case performs better than the timbral control on that point, as shown in Tabl. 4.13.

This finding underscores the effectiveness and strength of the signal-based descriptor approach.

These results suggest that the signal-based descriptor is a powerful and valuable approach for achieving accurate and scalable drum synthesis, and should be reasonably extensible to other musical instruments.

4.5.4 Discussion

On Neural Velocity Descriptor

Even if the performances shown in the results were not outstanding in any way, we were able to prove that it is indeed possible to use dynamic information as part of the control of a neural synthesizer.

When examining the half-opened hi-hat case, it becomes evident that when comparing

the actual audio with the generated audio for this category, the training dataset's audio samples exhibit either an absence or a subtle and momentary attack at the lowest levels. However, regardless of the velocity input, the generator produces a clear attack transient. Therefore, the velocity estimator interprets these sounds as having higher velocity values than desired. Here the velocity control loss may not be sufficient to counter the weight of the overwhelming majority of examples seen by the discriminator, and all have a strong attack transient. Conditioning the discriminator on the drum type and the velocity may help here.

Concerning the problems with tom and snares, we hypothesize that these more significant errors are because the generator does not produce drum sounds with the same signal properties as the real-world drum sounds shown in the subjective evaluation in Sect. 4.2.5. A particular problem are the resonances produced by the drums mentioned above: the generator replaces most of these using noise. This mismatch in signal properties is particularly problematic when training with an estimator trained only on authentic drum sounds. Due to the mismatch, the estimator may not produce a coherent gradient; accordingly, the loss will not work as expected. We note that the *fff* error remains relatively small for instruments with fewer strong resonances, like most cymbals.

On Signal-Based Velocity Descriptor

The signal-based estimator displayed excellent performance compared to its neural network counterpart. Our results were comparable to those obtained with the timbral control of StyleWaveGAN shown in Sect. 4.3. The same conclusion holds: reliable and deterministic losses are preferable to neural-based solutions if they exist. Expertly designed features are beneficial, especially in audio, given the extensive state-of-the-art research on perception and signal features [McAdams et al., 1995; Peeters, 2004; Peeters et al., 2011; Pearce et al., 2016; Lavault et al., 2022b] but will necessitate some preliminary engineering to make them compatible with deep-learning frameworks. While our experiment with StyleWaveGAN used only the raw energy, we can use the scaling obtained in Sect. 4.5.3 to convert input velocity to target energy. We also introduce an example of per-instrument scaling to ensure the velocity on the training set is within preset bounds for MIDI compatibility, while showing interpolation on the training dataset range is of great quality. At the same time, the extrapolation outside the limits remains acceptable.

Chapter 5

Conclusion

5.1 General Conclusion

With the advances of deep learning and especially deep generative models, a human-machine interface responding to complex human perception will transform how music and sound can be produced. This document presented how Generative Adversarial Networks (GAN) can build a neural drum synthesizer with intuitive and musically relevant controls.

Starting from a review of the existing state-of-the-art methods and their limitations, we built upon these by choosing one specific generative model. We augmented its capabilities to provide improved perceived quality and high-level control features. The neural synthesizer we present in this document, StyleWaveGAN, is a modified image generator built to generate waveforms and accept different control inputs, allowing for good inference performance while being able to generate longer sounds compared to the state-of-the-art.

From there on, we showed that the proposed model demonstrates improved synthesis quality compared to other specialized drum generation models for both objective metrics (FAD) and the perceived quality as measured in perceptual tests. This is substantial progress on the way toward high-quality neural synthesizers.

In addition to sound generation quality, the project's other objective was to create a controllable synthesizer with high-level features. By implementing differentiable variants of commonly used timbre descriptors and using these to establish a differentiable feature control loss, we significantly improved the control precision achieved with the proposed model compared to the existing approaches from the state of the art. We observed notable enhancements in our system's performance. This encouraged us to adopt a refined metric for measuring control precision based on absolute errors at particular quantile-based regions since the previous metric used to assess control quality was relatively weak. To establish a baseline, we conducted psychophysical experiments to gauge the perception threshold around the values used in this new metric. Our findings indicate that the control error with our method is not noticeable except in the most extreme cases.

Finally, we experimented with adding different oscillators and velocity control to StyleWaveGAN to improve the generation quality and go even further with drums' synthesis control based on the specialized sensor data from a new drum sound dataset.

All in all, this document presents the results of the use of generative adversarial networks for the synthesis and control of drum sounds, where we achieved the following:

- improvement over state-of-the-art neural drum synthesizers while not achieving fully realistic synthesis (section 4.2.5)
- a sufficiently fast model to be capable of being integrated into real-time applications,
- Basic and more advanced control over the timbral properties of the sounds 4.3 and velocity 4.5

5.2 Limitations and Future Works

As it stands now, StyleWaveGAN achieves good performances, but we cannot claim it can generate realistic samples in all cases. Even if the worst examples can be used in a musical context with some processing added, the quality of certain types of drums can be seen as uneven. These results are mostly due to a potentially weak discriminator that could benefit from working in the frequency domain, making it better at discriminating unstable sinusoids and then classifying them as fake.

Regarding velocity and dynamics, we confirmed the superiority of a signal-based descriptor over neural network-based ones, as we have already shown with timbral descriptors. However, this descriptor only loosely relates to standard MIDI velocity control and necessitates fine-tuning per instrument to make it fit the MIDI velocity range.

Finally, the timbral features StyleWaveGAN uses do not cover the entire AudioCommons feature set. While this was justified by doing an informal survey among drummers and musicians to know which were the features to implement in priority, having the whole set reimplemented and compatible with deep-learning frameworks would make the results of this thesis even more substantial when it comes to differentiable timbral features and their integration to the network control.

In terms of future works, the list is still long. First, the neverending quest for improved sound quality is followed closely by extending the results to both public and private parts of Apeira Drums. While recording as many samples as the augmented ENST-Drums seems impossible, an extended Apeira Drums coupled with a revised augmentation pipeline could provide a dataset similar in size to the augmented ENST-Drums presented in this document while not being subject to a perceivable degradation in quality, as was ENST-AUG. From this augmented Apeira Drums dataset, we would have to reintroduce the timbral features and expand the list of available timbral features from the AudioCommons package to not only the three prioritized ones in this document.

Bibliography

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, A., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. (2016). YouTube-8M: A Large-Scale Video Classification Benchmark. *ArXiv*, abs/1609.08675.

Anderson, T. W. and Darling, D. A. (1952). Asymptotic Theory of Certain "Goodness of Fit" Criteria Based on Stochastic Processes. *The Annals of Mathematical Statistics*, 23(2):193–212.

Andresen, U. (1979). A New Way in Sound Synthesis. In *Audio Engineering Society Convention 62*.

ANSI/ASA S1.1-2013 (2020). Acoustical terminology. Standard, American National Standards Institute/Acoustical Society of America. https://webstore.ansi.org/standards/asa/ansiasas12013r2020.

Aouameur, C., Esling, P., and Hadjeres, G. (2019). Neural Drum Machine: An Interactive System for Real-time Synthesis of Drum Sounds. In *International Conference on Computational Creativity*.

Arjovsky, M. and Bottou, L. (2017). Towards Principled Methods for Training Generative Adversarial Networks. *arXiv preprint arXiv:1701.04862*.

Avanzini, F., Marogna, R., and Bank, B. (2012). Efficient synthesis of tension modulation in strings and membranes based on energy estimation. *The Journal of the Acoustical Society of America*, 131(1):897–906.

Barratt, S. and Sharma, R. (2018). A Note on the Inception Score. *arXiv preprint arXiv:1801.01973*.

Bilbao, S. (2012). Time domain simulation and sound synthesis for the snare drum. *The Journal of the Acoustical Society of America*, 131(1):914–925.

Bilbao, S. and Webb, C. J. (2013). Physical Modeling of Timpani Drums in 3D on GPGPUs. *J. Audio Eng. Soc*, 61(10):737–748.

Bonada, J., Serra, X., Amatriain, X., and Loscos, A. (2011). *Spectral Processing*, chapter 10, pages 393–445. John Wiley & Sons, Ltd.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294.

Bous, F. and Roebel, A. (2023). Analysis and Transformation of Voice Level in Singing Voice. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Brown, A. (1990). Modern Jazz Drumset Artistry. *The Black Perspective in Music*, 18(1/2):39.

Caillon, A. and Esling, P. (2021). RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. *arXiv preprint arXiv:2111.05011*.

Callender, L., Hawthorne, C., and Engel, J. (2020). Improving Perceptual Quality of Drum Transcription with the Expanded Groove MIDI Dataset.

Caspe, F., McPherson, A., and Sandler, M. (2022). DDX7: Differentiable FM Synthesis of Musical Instrument Sounds. *Proceedings of the 23rd International Society for Music Information Retrieval Conference*.

Chaigne, A., Touzé, C., and Thomas, O. (2005). Nonlinear vibrations and chaos in gongs and cymbals. *Acoustical science and technology*, 26(5):403–409.

Chamberlin, H. (1985). *Musical applications of microprocessors*. Hayden Books, Indianapolis, IN.

Chowning, J. M. (1977). The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. *Computer Music Journal*, 1(2):46–54.

Clark, J. J. (2003). Advanced programming techniques for modular synthesizers.

Collins, S. (2007). Amen to that: sampling and adapting the past. *M/C Journal*, 10(2).

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE.

Dhariwal, P. and Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc.

Dittmar, C. and Gärtner, D. (2014). Real-time transcription and separation of drum recordings based on NMF decomposition. In *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, Erlangen.

Donahue, C., McAuley, J., and Puckette, M. (2019). Adversarial Audio Synthesis. In *International Conference on Learning Representations*.

Dowson, D. and Landau, B. (1982). The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455.

Drysdale, J., Tomczak, M., and Hockman, J. (2020). Adversarial Synthesis of Drum Sounds. *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020),*, pages 24–30.

Drysdale, J., Tomczak, M., and Hockman, J. (2021). Style-based drum synthesis with GAN inversion. In *Extended Abstracts for the Late-Breaking Demo Session of the 22nd Int. Society for Music Information Retrieval Conference*.

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. (2019). GANSynth: Adversarial Neural Audio Synthesis. In *International Conference on Learning Representations*.

Engel, J., Hantrakul, L. H., Gu, C., and Roberts, A. (2020). DDSP: Differentiable Digital Signal Processing. In *International Conference on Learning Representations*.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. (2017). Neural audio synthesis of musical notes with WaveNet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077. PMLR.

Fletcher, H. and Bassett, I. G. (1978). Some experiments with the bass drum. *The Journal of the Acoustical Society of America*, 64(6):1570–1576.

Fletcher, N. H. and Rossing, T. D. (1998). *The Physics of Musical Instruments*. Springer New York.

Font, F., Roma, G., and Serra, X. (2013). Freesound Technical Demo. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, page 411–412, New York, NY, USA. Association for Computing Machinery.

Fréchet, M. (1957). Sur la distance de deux lois de probabilité. *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences*, 244(6):689–692.

Gillet, O. and Richard, G. (2006). ENST-Drums: An extensive audio-visual database for drum signals processing. *ISMIR 2006 - 7th International Conference on Music Information Retrieval*, pages 156–159.

Gillick, J., Roberts, A., Engel, J., Eck, D., and Bamman, D. (2019). Learning to Groove with Inverse Sequence Transformations. In *International Conference on Machine Learning (ICML)*.

Giordano, B. L., Susini, P., and Bresin, R. (2012). Experimental methods for the perceptual evaluation of sound-producing objects and interfaces. In *Sonic Interaction Design* :. MIT Press.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5768–5778.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778.

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. (2017). CNN Architectures for Large-Scale Audio Classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 131–135. IEEE Press.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 6627–6638.

Hjortkjær, J. and Walther-Hansen, M. (2014). Perceptual effects of dynamic range compression in popular music recordings. *Journal of the Audio Engineering Society*, 62(1/2):37–41.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Holmes, T. (2020). *Electronic and experimental music*. Routledge, London, England, 6 edition.

ISO 226 (2003). Acoustics — Normal equal-loudness-level contours. Standard, International Organization for Standardization, Geneva, CH. https://www.iso.org/fr/standard/34222.html.

ITU (1996). Methods for subjective determination of transmission quality. Recommendation P.800, International Telecommunication Union. https://www.itu.int/rec/T-REC-P.800-199608-I.

Jacques, C. and Roebel, A. (2018). Automatic drum transcription with convolutional neural networks. In *21th International Conference on Digital Audio Effects, Sep 2018, Aveiro, Portugal*.

Juanpere, E. M. and Välimäki, V. (2022). Realistic gramophone noise synthesis using a diffusion model. In *25th International Conference on Digital Audio Effects (DAFx20in22)*, pages 240–247.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

Karras, T., Laine, S., and Aila, T. (2019). 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.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

Katz, B. (2014). *Mastering Audio*. Focal Press, Oxford, England, 3 edition.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. (2019). Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2019-Septe:2350–2354.

Kim, J. W., Salamon, J., Li, P., and Bello, J. P. (2018). Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2021). Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Lavault, A., Roebel, A., and Voiry, M. (2022a). StyleWaveGAN: Style-based synthesis of drum sounds using Generative Adversarial Networks for higher audio quality. In *30th European Signal Processing Conference (EUSIPCO 2022)*, Belgrade, Serbia.

Lavault, A., Roebel, A., and Voiry, M. (2022b). StyleWaveGAN: Style-based Synthesis of Drum Sounds with Extensive Controls using Generative Adversarial Networks. In *19th Sound and Music Computing Conference (SMC 2022)*, Saint-Etienne, France.

Lavault, A., Roebel, A., and Voiry, M. (2022c). Subjective Evaluation of Sound Quality and Control of Drum Synthesis using StyleWaveGAN. In *25th International Conference on Digital Audio Effects (DAFx20in22)*, Vienna, Austria.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris*, pages 599–604.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Legge, K. A. and Fletcher, N. H. (1989). Nonlinearity, chaos, and the sound of shallow gongs. *The Journal of the Acoustical Society of America*, 86(6):2439–2443.

Liski, J., Mäkivirta, A., and Välimäki, V. (2021). Audibility of group-delay equalization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2189–2201.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares Generative Adversarial Networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802.

McAdams, S., Winsberg, S., Donnadieu, S., De Soete, G., and Krimphoff, J. (1995). Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes. *Psychological Research*, 58(3):177–192.

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for GANs do actually converge? In *35th International Conference on Machine Learning, ICML 2018*, volume 8, pages 5589–5626. International Machine Learning Society (IMLS). http://arxiv.org/abs/1801.04406.

Mirza, M. and Osindero, S. (2014). Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*.

Nistal, J. (2022). *Exploring Generative Adversarial Networks for controllable musical audio synthesis*. PhD thesis, Institut Polytechnique de Paris.

Nistal, J., Aouameur, C., Velarde, I., and Lattner, S. (2022). DrumGAN VST: A Plugin for Drum Sound Analysis/Synthesis With Autoencoding Generative Adversarial Networks. *arXiv e-prints*, pages arXiv–2206.

Nistal, J., Lattner, S., and Richard, G. (2020). DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks. In *ISMIR*.

Nistal, J., Lattner, S., and Richard, G. (2021). Comparing representations for audio synthesis using generative adversarial networks. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 161–165. IEEE.

Odena, A., , V., and Olah, C. (2016). Deconvolution and Checkerboard Artifacts. *Distill*, 1(10):e3.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional Image Synthesis with Auxiliary Classifier GANs. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651. PMLR.

Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076.

Pearce, A., Brookes, T., and Mason, R. (2016). Hierarchical ontology of timbral semantic descriptors. *AudioCommons - Deliverable D5.1*, pages 1–34.

Peeters, G. (2004). A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *CUIDADO Ist Project Report*, 54(0):1–25.

Peeters, G., Giordano, B. L., Susini, P., Misdariis, N., and McAdams, S. (2011). The Timbre Toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916.

Petzka, H., Fischer, A., and Lukovnikov, D. (2018). On the regularization of Wasserstein GANs. In *International Conference on Learning Representations*.

Pieslak, J. (2007). Re-casting Metal: Rhythm and Meter in the Music of Meshuggah. *Music Theory Spectrum*, 29(2):219–245.

Ramires, A., Chandna, P., Favory, X., Gomez, E., and Serra, X. (2020). Neural Percussive Synthesis Parameterised by High-Level Timbral Features. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2020-May, pages 786–790. Institute of Electrical and Electronics Engineers Inc.

Ramires, A., Juras, J., Parker, J. D., and Serra, X. (2022). A Study of Control Methods for Percussive Sound Synthesis Based On GANss. In *25th International Conference on Digital Audio Effects (DAFx20in22)*.

Renault, L., Mignot, R., and Roebel, A. (2022). Differentiable piano model for midi-to-audio performance synthesis. In *International Conference on Digital Audio Effects*.

Roads, C., Pope, S. T., Piccialli, A., and De Poli, G. (2013). *Musical Signal Processing*. Routledge.

Roads, C. and Strawn, J. (1996). *The computer music tutorial*. MIT Press.

Röbel, A. (2003). A new approach to transient processing in the phase vocoder. In *Proc. of the 6th Int. Conf. on Digital Audio Effects (DAFx03)*, pages 344–349.

Rodet, X. (1997). Sinusoidal+residual models for musical sound signals analysis/synthesis. *Applied Signal Processing*, 4:131–141.

Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837.

Rossing, T. D. (2000). *Science of Percussion Instruments*. World Scientific.

Rossing, T. D., Bork, I., Zhao, H., and Fystrom, D. O. (1992). Acoustics of snare drums. *The Journal of the Acoustical Society of America*, 92(1):84–94.

Rouard, S. and Hadjeres, G. (2021). CRASH: Raw Audio Score-based Generative Modeling for Controllable High-resolution Drum Sound Synthesis. In Lee, J. H., Lerch, A., Duan, Z., Nam, J., Rao, P., van Kranenburg, P., and Srinivasamurthy, A., editors, *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, pages 579–585.

Rumelhart, D. E. and McClelland, J. L. (1987). *Learning Internal Representations by Error Propagation*, pages 318–362. MIT Press.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sak, H., Senior, A. W., and Beaufays, F. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In *INTERSPEECH*, pages 338–342.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. (2016). Improved techniques for training gans. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Savitzky, A. and Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8):1627–1639.

Scott, D. W. (1992). *Multivariate Density Estimation*. Wiley.

Serra, X. and Smith, J. (1990). Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition. *Computer Music Journal*, 14(4):12–24.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Smith, J. O. (2010). *Physical Audio Signal Processing*. http://ccrma.stanford.edu/~jos/pasp/. online book, 2010 edition.

Smith, J. O. and Serra, X. (1987). Parshl: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation. In *Proceedings of the 1987 International Computer Music Conference, ICMC; 1987 Aug 23-26; Champaign/Urbana, Illinois.[Michigan]: Michigan Publishing; 1987. p. 290-7.* International Computer Music Conference.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France. PMLR.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. (2020). Learning to summarize with human feedback. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Stilson, T. S. and Smith, J. O. (1996). Alias-free digital synthesis of classic analog waveforms. In *International Conference on Mathematics and Computing*.

Su, K., Liu, X., and Shlizerman, E. (2020). Audeo: Audio generation for a silent performance video. In *Advances in Neural Information Processing Systems*, volume 2020-Decem. Neural information processing systems foundation.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2018). Wasserstein auto-encoders. In *International Conference on Learning Representations*.

Torrey, L. and Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Toulson, R. (2021). *Drum Sound and Drum Tuning*. Focal Press.

Toulson, R. and Hardin, M. (2020). Evaluating the accuracy of musicians and sound engineers in performing a common drum tuning exercise. In *Audio Engineering Society Convention 149*. Audio Engineering Society.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125. ISCA.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3915–3923. PMLR.

Vaserstein, L. N. (1969). Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research, ll (Dec)*, pages 3371–3408.

Wichmann, F. and Hill, N. (2001a). The psychometric function: I. Fitting, sampling, and goodness of fit. *Perception and Psychophysics*, 63:1290–1313.

Wichmann, F. and Hill, N. (2001b). The psychometric function: II. Bootstrap-based confidence intervals and sampling. *Perception and psychophysics*, 63:1314–29.

Wilbur, C. and Rossing, T. D. (1997). Subharmonic generation in cymbals at large amplitude. *Journal of the Acoustical Society of America*, 101:3144–3144.

Xiao, Z., Kreis, K., and Vahdat, A. (2022). Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. In *International Conference on Learning Representations*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.