Exploring generative adversarial networks for controllable musical audio synthesis

Javier Nistal Hurlé

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAT009

Thèse de doctorat

Exploring Generative Adversarial Networks for Controllable Musical Audio Synthesis

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 9/03/2022, par

JAVIER NISTAL HURLÉ

Composition du Jury :

Axel Roebel Research Director, IRCAM (SAS), Paris, France	Président
Xavier Serra Professor, Universitat Pompeu Fabra (MTG), Barcelona, Spain	Rapporteur
Sølvi Ystad Research Director, CNRS (PRISM), Marseille, France	Rapporteuse
Mathieu Fontaine Associate Professor, Télécom Paris, IP Paris, France	Examinateur
Gaël Richard Professor, Télécom Paris, IP Paris, France	Directeur de thèse
Stefan Lattner Associate Researcher, Sony CSL, Paris, France	Co-directeur de thèse
Lonce Wyse Associate Professor, NUS (CNM), Singapore	Invité
Jean-Baptiste Rolland Research Senior Software Developer, Steinberg Media Technologies, Germany	Invité

Acknowledgement

First and foremost, I am extremely grateful to my supervisors, Prof. Gaël Richard and Dr. Stefan Lattner, for counting on me for this project and their invaluable advice, continuous support, and patience during my Ph.D. study. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would especially like to appreciate the implication of Stefan in any hurdle along the way of this project, you name it: writing, coding, inspiration, bureaucracy, existential... Stefan has spent very long deadline nights and has been there for any problem I could have. I would like to specially mention our late-night inspirational discussions at Outland bar, resulting in most of the crazy ideas behind the work described here and many others to be carried out. I'm looking forward to our future projects.

I want to thank all the staff at Sony CSL for making this Ph.D. project possible. It is their kind help and support that have made my study and life in Paris a wonderful time. I would like to send a special thanks to some colleagues and former colleagues:

- Emmanuel Deruty, for supporting my work and providing unlimited resources to make it possible (e.g., data, GPUs, promotion). The impact, reach, and, ultimately, the success of this project has significantly been possible thanks to his work. I would also like to appreciate his trust and confidence in my work and autonomy.

- Cyran Aouameur, for his fellowship and continuous support, in and outside the lab. Cyran created DrumGAN's VST interface and has continuously supported me with any matter (e.g., writing, presentations, coding, bureaucracy, well-being). It's been an absolute pleasure working with him, and I hope we'll continue doing so for long!

- Michael Turbot, for his implication and confidence in my work, and specifically in DrumGAN. He was responsible for creating a promotional video teaser about my work, and he has contributed to the reach and impact of this project.

- Michael Anslow, for his friendship and support and, also, our many inspirational conversations.

- Dr. Gaëtan Hadjeres, Dr. Leopold Crestel, Dr. Maarten Grachten, and Theis Bazin for their always thoughtful input and support and for their work's inspiration.

- Dr. Matthias Demoucron, for his implication in DrumGAN and his support with many legal and bureaucratic issues.

- Amaury Delourt, for helping me with any data-related matter, and his interest in my work.
- Pratik Bhoir, for his patience and support with any IT matter. Thanks to him, connecting to servers, using Sony's internal services, and configuring my computer has been a piece of cake.
- Dr. Stephane Rivaud, for his excellent lectures on GANs and the inspirational talks at the beginning of my Ph.D.
- Ithan Velarde, for his work on DrumGAN's encoder.
- Jeremy Uzan, for his interest in my work and his support outside the lab.
- Sophie Boucher and Cristina Nunu, for supporting and helping me with any bureaucratic and legal issues I could have.

I would like to thank all the members of the ADASP group at Telecom and the MIP-Frontiers network. I would like to specially thank my colleagues Giorgia Cantisani, Karim Ibrahim, Kilian Schultz, and Ondrej Cifka for their support throughout the project.

Finally, I would like to express my gratitude to my parents, my brother and my friends. Without their tremendous understanding and encouragement in the past three years, it would have been impossible for me to complete this work in one piece.

This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068 (MIP-Frontiers).

Sony CSL

Abstract

Audio synthesizers are electronic musical instruments that generate artificial sounds under some parametric control. While synthesizers have evolved since they were popularized in the 70s, two fundamental challenges are still unresolved: 1) the development of synthesis systems responding to semantically intuitive parameters; 2) the design of "universal," source-agnostic synthesis techniques. This thesis researches the use of Generative Adversarial Networks (GAN) towards building such systems. The main goal is to research and develop novel tools for music production that afford intuitive and expressive means of sound manipulation, e.g., by controlling parameters that respond to perceptual properties of the sound and other high-level features.

Our first work studies the performance of GANs when trained on various common audio signal representations (e.g., waveform, time-frequency representations). These experiments compare different forms of audio data in the context of tonal sound synthesis. Results show that the Magnitude and Instantaneous Frequency of the phase and the complex-valued Short-Time Fourier Transform achieve the best results.

Building on this, our following work presents DrumGAN, a controllable adversarial audio synthesizer of percussive sounds. We demonstrate that intuitive control can be gained over the generation process by conditioning the model on perceptual features describing high-level timbre properties. This work results in developing a VST plugin generating full-resolution audio and compatible with any Digital Audio Workstation (DAW). We show extensive musical material produced by professional artists from Sony ATV using DrumGAN.

The scarcity of annotations in musical audio datasets challenges the application of supervised methods to conditional generation settings. Our third contribution employs a knowledge distillation approach to extract such annotations from a pre-trained audio tagging system. DarkGAN is an adversarial synthesizer of tonal sounds that employs the output probabilities of such a system (so-called "soft labels") as conditional information. Results show that DarkGAN can respond moderately to many intuitive attributes, even with out-of-distribution input conditioning.

Applications of GANs to audio synthesis typically learn from fixed-size two-dimensional spectrogram data analogously to the "image data" in computer vision; thus, they cannot generate sounds with variable duration. Our fourth paper addresses this limitation by exploiting a self-supervised method for learning discrete features from sequential data. Such features are used as conditional input to provide step-wise time-dependent information to the model. Global consistency is ensured by fixing the input noise $\mathbf{z}$ (characteristic in adversarial settings). Results show that, while models trained on a fixed-size scheme obtain better audio

quality and diversity, ours can competently generate audio of any duration.

One interesting direction for research is the generation of audio conditioned on preexisting musical material, e.g., the generation of some drum pattern given the recording of a bass line. Our fifth paper explores a simple pretext task tailored at learning such types of complex musical relationships. Concretely, we study whether a GAN generator, conditioned on highly compressed MP3 musical audio signals, can generate outputs resembling the original uncompressed audio. Results show that the GAN can improve the quality of the audio signals over the MP3 versions for very high compression rates (16 and 32 kbit/s).

As a direct consequence of applying artificial intelligence techniques in musical contexts, we ask how AI-based technology can foster innovation in musical practice. Therefore, we conclude this thesis by providing a broad perspective on the development of AI tools for music production, informed by theoretical considerations and reports from real-world AI tool usage by professional artists.

Résumé

Les synthétiseurs audio sont des instruments de musique électroniques qui génèrent des sons artificiels sous un certain contrôle paramétrique. Alors que les synthétiseurs ont évolué depuis leur popularisation dans les années 70, deux défis fondamentaux restent encore non résolus: 1) le développement de systèmes de synthèse répondant à des paramètres sémantiquement intuitifs; 2) la conception de techniques de synthèse «universelles», indépendantes de la source à modéliser. Cette thèse étudie l'utilisation des réseaux adversariaux génératifs (ou GAN) pour construire de tels systèmes. L'objectif principal est de rechercher et de développer de nouveaux outils pour la production musicale, qui offrent des moyens intuitifs et expressifs de manipulation du son, par exemple en contrôlant des paramètres qui répondent aux propriétés perceptives du son et à d'autres caractéristiques.

Notre premier travail étudie les performances des GAN lorsqu'ils sont entraînés sur diverses représentations de signaux audio (par exemple, forme d'onde, représentations temps-fréquence). Ces expériences comparent différentes formes de données audio dans le contexte de la synthèse sonore tonale. Les résultats montrent que la représentation magnitude-fréquence instantanée et la transformée de Fourier à valeur complexe obtiennent les meilleurs résultats.

En s'appuyant sur ce résultat, notre travail suivant présente DrumGAN, un synthétiseur audio de sons percussifs. En conditionnant le modèle sur des caractéristiques perceptives décrivant des propriétés timbrales de haut niveau, nous démontrons qu'un contrôle intuitif peut être obtenu sur le processus de génération. Ce travail aboutit au développement d'un plugin VST générant de l'audio haute résolution et compatible avec les Stations de Travail Audio Numériques (STAN). Nous montrons un vaste matériel musical produit par des artistes professionnels de Sony ATV à l'aide de DrumGAN.

La rareté des annotations dans les ensembles de données audio musicales remet en cause l'application de méthodes supervisées pour la génération conditionnelle. Notre troisième contribution utilise une approche de distillation des connaissances pour extraire de telles annotations à partir d'un système d'étiquetage audio pré-entraîné. DarkGAN est un synthétiseur de sons tonaux qui utilise les probabilités de sortie d'un tel système (appelées « étiquettes souples ») comme informations conditionnelles. Les résultats montrent que DarkGAN peut répondre modérément à de nombreux attributs intuitifs, même avec un conditionnement d'entrée hors distribution.

Les applications des GAN à la synthèse audio apprennent généralement à partir de données de spectrogramme de taille fixe, de manière analogue aux «données d'image» en vision par ordinateur; ainsi, ils ne peuvent pas générer de sons de durée variable. Dans notre quatrième article, nous abordons cette limitation en exploitant une méthode auto-supervisée pour l'apprentissage de caractéristiques

discrètes à partir de données séquentielles. De telles caractéristiques sont utilisées comme entrée conditionnelle pour fournir au modèle des informations dépendant du temps par étapes. La cohérence globale est assurée en fixant le bruit d'entrée z (caractéristique en GANs). Les résultats montrent que, tandis que les modèles entraînés sur un schéma de taille fixe obtiennent une meilleure qualité et diversité audio, les nôtres peuvent générer avec compétence un son de n'importe quelle durée.

Une direction de recherche intéressante est la génération d'audio conditionnée par du matériel musical préexistant, par exemple, la génération d'un motif de batterie compte tenu de l'enregistrement d'une ligne de basse. Notre cinquième article explore une tâche prétexte simple adaptée à l'apprentissage de tels types de relations musicales complexes. Concrètement, nous étudions si un générateur GAN, conditionné sur des signaux audio musicaux hautement compressés, peut générer des sorties ressemblant à l'audio non compressé d'origine. Les résultats montrent que le GAN peut améliorer la qualité des signaux audio par rapport aux versions MP3 pour des taux de compression très élevés (16 et 32 kbit/s).

En conséquence directe de l'application de techniques d'intelligence artificielle dans des contextes musicaux, nous nous demandons comment la technologie basée sur l'IA peut favoriser l'innovation dans la pratique musicale. Par conséquent, nous concluons cette thèse en offrant une large perspective sur le développement d'outils d'IA pour la production musicale, éclairée par des considérations théoriques et des rapports d'utilisation d'outils d'IA dans le monde réel par des artistes professionnels.

Contents

List of Figures

List of Tables

List of publications

⟨**EUSIPCO2019**⟩
Javier Nistal, Stefan Lattner, and Gaël Richard. Comparing Representations for Audio Synthesis Using Generative Adversarial Networks In *Proceedings of the 28th European Signal Processing Conference (EUSIPCO 2019)*, pages 161-165, Amsterdam, The Netherlands, November 2019. EUSIPCO. doi: 10.23919/Eusipco47968.2020.9287799. `https://ieeexplore.ieee.org/document/9287799`.

⟨**ISMIR2020**⟩
Javier Nistal, Stefan Lattner, and Gaël Richard. DrumGAN: Synthesis of Drum Sounds with Timbral Feature Conditioning Using Generative Adversarial Networks. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, pages 590–597, Montreal, Canada, October 2020. ISMIR. URL `http://archives.ismir.net/ismir2020/paper/000255.pdf`.

⟨**ISMIR2021**⟩
Javier Nistal, Stefan Lattner, and Gaël Richard. DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis With GANs. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Virtual, November 2021. ISMIR. `https://hal.archives-ouvertes.fr/hal-03349492/document`.

⟨**WASPAA2021**⟩
Javier Nistal, Cyran Aouameur, Stefan Lattner, and Gaël Richard. VQCPC-GAN: Variable- Length Adversarial Audio Synthesis using Vector-Quantized Contrastive Predictive Coding. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2021, New Paltz, United States. `https://hal.telecom-paris.fr/hal-03413460/document`.

⟨**MDPI2021**⟩
Stefan Lattner, and Javier Nistal. Stochastic Restoration of Heavily Compressed Musical Audio using Generative Adversarial Networks. Electronics 10, no. 11: 1349. MDPI. `https://doi.org/10.3390/electronics10111349`.

⟨**TISMIR2022**⟩
Emanuel Deruty, Martin Grachteen, Stefan Lattner, Javier Nistal, and Cyran Aouameur. On the development and practice of AI technology for contemporary popular music production. *Transactions of the International Society for Music Information Retrieval (TISMIR)*

Notation

We generally use bold symbols to indicate vectors (lowercase) and matrices (uppercase). Uppercase symbols may be also used to indicate constants. When theory is applicable to either scalars, vectors or n-dimensional algebraic objects (tensors), we employ lowercase symbols. We indicate probability distribution by $p(\cdot)$. We often add a subscript to probability densities, e.g. $p_\theta(x)$, to indicate the distribution of random variables x with distributional parameters θ. The symbol $\mathrm{E}(\cdot)$ represents the expected value operator. Finally, we represent the sampling or simulation of variates x from a distribution $p(x)$ using the notation $x \sim p(x)$.

Abbreviations

AE	Autoencoder
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CPC	Contrastive Predictive Coding
CPM	Contemporary Popular Music
CQT	Constant-Q Transform
CV	Computer Vision
DAW	Digital Audio Workstation
DL	Deep Learning
DNN	Deep Neural Network
DSP	Digital Signal Processing
FAD	Fréchet Audio Distance
FFT	Fast Fourier Transform
FM	Frequency Modulation
GAN	Generative Adversarial Network
GNN	Generative Neural Network
GP	Gradient Penalty
GUI	Graphical User Interface
IAF	Inverse Autoregressive Flow
IS	Inception Score
IIS	Instrument Inception Score
KD	Knowledge Distillation
LFO	Low-Frequency Oscillator
LSTM	Long Short-Term Memory
MIDI	Musical Instrument Digital Interface
MFCC	Mel Frequency Cepstrum Coefficients
ML	Machine Learning
NAM	Neural Autoregressive Model
NAS	Neural Audio Synthesizer
NF	Normalizing Flow
PGAN	Progressive Growing GAN
PIS	Pitch Inception Score
RNN	Recurrent Neural Network
STFT	Short-Time Fourier Transform
VAE	Variational Autoencoder
VST	Virtual Studio Technology
VQ	Vector Quantization
WGAN	Wasserstein GAN

Chapter 1

Introduction

> "I dream of instruments obedient to my thought and which with their contribution of a whole new world of unsuspected sounds, will lend themselves to the exigencies of my inner rhythm"
>
> — Edgard Varèse, 1917

Audio synthesizers are electronic musical instruments that generate artificial sounds under some parametric control. Popularized during the 70s, these devices, now ubiquitous in most of the music we listen to, have since reshaped music production, giving birth to new music genres and novel paradigms for musical interaction and expression.

While synthesizers have evolved since they first appeared, two fundamental challenges are still faced. One is the development of genuinely accessible and user-driven synthesis systems, responding to semantically intuitive parameters. Fig. 1.1 shows the interface of Sonicbits' Exakt Lite, an "intuitive and user-friendly FM synthesizer plugin."[1] Waveform, filter, frequency modulation, etc.: the vast and complex parameter space that synthesizers afford is an unquestioned source of inspiration for a few, yet, for the many, they pose an obstacle that slows down the creative process. This process may seem analogous to any other musical instrument: one must train hard to fully unveil its sonic possibilities, e.g., performing vibrato or finger-tapping in a guitar to achieve different timbres requires expertise. However, anyone can understand the guitar's interaction protocol, i.e., the set of mechanic rules to obtain a specific sound. Synthesizers, on the contrary, require strong signal processing knowledge to guide the system towards a specific sound purposely. The main barrier resides in the semantic gap between the synthesis parameters and the musician's cognitive factors driving creative thoughts. The synthesizer's language is signal processing, whereas the artist's language concerns abstract properties such as emotions, perception, experiences, or musical aesthetics, to name a few. Under current systems, it is the user's responsibility to translate this unstructured context into the synthesizer's parameter space and not the opposite; this is the synth incorporating parameters that can respond to the musician's language. An overview of this process is further illustrated in Fig. 1.2.

The second challenge is designing "universal," source-agnostic synthesis techniques that can approximate any timbre and offer generic workflows. As we

[1] https://www.sonicbits.com/exakt-lite.html

Figure 1.1 – Graphical User Interface (GUI) of SONICBITS Exakt Lite FM synthesizer.

will see in Chapter 3, many different techniques exist for audio synthesis. Each technique understands sound differently, conferring specific characteristics to the generated sound and providing specific means of control. Therefore, some synthesizers may be better suited than others for generating specific sounds or for specific musical purposes.

From the above-described context we can identify four main challenges and limitations that music producers may face when working with synthesizers: 1) the need for dedicated software or hardware for specific musical purposes; 2) mastering of different synthesis techniques and workflows; 3) the need for sample libraries due to the unavailability of technology for modeling specific sound sources; 4) the creative barrier that current synthesizers impose through their obscure terminology and workflows. The question arises as to how we can design synthesis techniques exposing intuitive parameters that respond to acoustic (e.g., source, space), musical (e.g., harmony, genre), or perceptual (e.g., pitch, timbre) properties of the sound and with rich timbral capabilities.

1.1 Deep Learning Meets Audio Synthesis

The field of Deep Learning (DL) offers new approaches to synthesizing audio that may pave the way towards building such systems. Generative Neural Networks (GNNs) are biologically-inspired computer algorithms that utilize statistical rules to learn models from some training data (a more formal introduction is given in Chapter 2). A neural audio synthesizer refers to a GNN trained on a sound dataset. Once trained, GNNs can generate new data without being explicitly programmed to do so. They can also be conditioned on preexisting information to gain control over specific features, or, they can even learn by themselves to find meaningful features in the data. This is in contrast to expert models —such as synthesizers— which rely on static and explicitly stated models of sound built

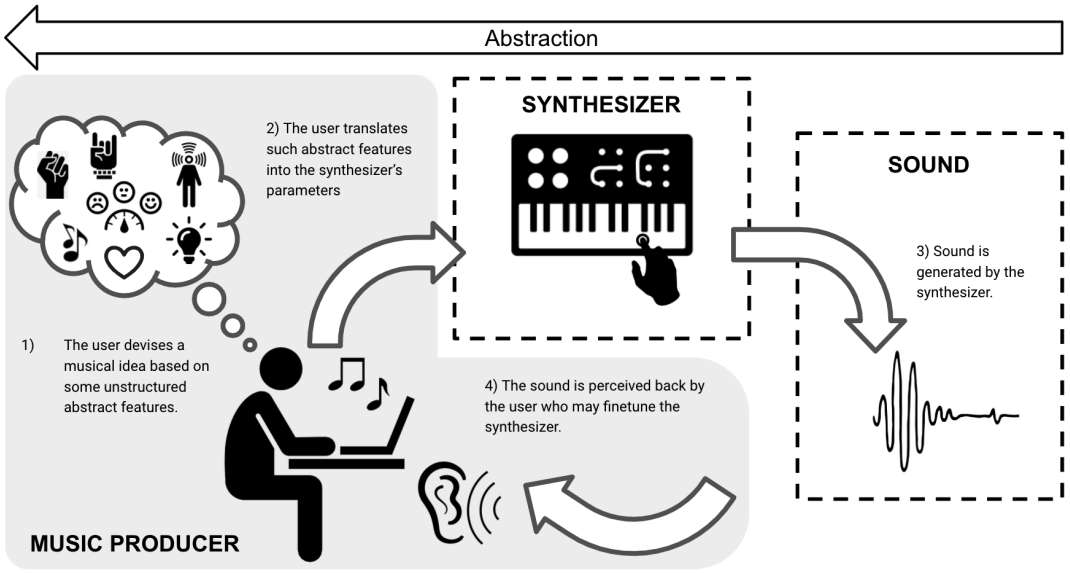


Figure 1.2 – Problem overview.* The user starts by devising a musical idea that can integrate various information, including mood, emotional state, a melody, preexisting musical content (e.g., a pre-recorded bass line), and more. We call these high-level features because of their high degree of abstraction. The user must then adjust the parameters of the synthesizer to obtain a specific sound. If no clear sound is targeted, one can wander around through the parameter space. The sound produced by the synth is perceived back by the user, who may incorporate this information to make new adjustments.

* Icon acknowledgements can be found in Appendix A.

upon prior knowledge of the domain. As we will mention in Sec. 3.2, devising controls responding to abstract sound properties in expert systems requires an in-depth study of such specific properties an their relationship with low-level features observed in the signal [Ystad, 1998]. Further, in many cases the relationship of such attributes with the audio signal is an ill-defined problem with no unique solution. Music is a design task where no single algorithm exists to transform some abstract initial state into an underdefined goal state, e.g., there is no single way of composing a bassline for some pre-recorded content or given some abstract description. Following this paradigm, in Fig. 1.3, we illustrate how interaction with a synthesizer could look like under the lens of deep learning. Even though complete adaptation to the user will still require fundamental developments in, e.g., active learning, representation learning, machine listening, or adaptive user interfaces, to name but a few fields, the work contained in this thesis contributes to building such systems.

DL has led to remarkable breakthroughs in Computer Vision (CV) due to fundamental developments such as Generative Adversarial Networks (GAN). GANs are powerful generative neural networks capable of synthesizing photo-realistic face images [Karras et al., 2018] or rendering high-definition images from a sketched landscape [Park et al., 2019]. In the field of audio, most of the work has been oriented towards speech synthesis for human-machine translation tasks [Sotelo et al., 2017, Ping et al., 2017, Shen et al., 2018]. Research on musical audio is scarce mainly due to the high-quality standards of musical applications cou-

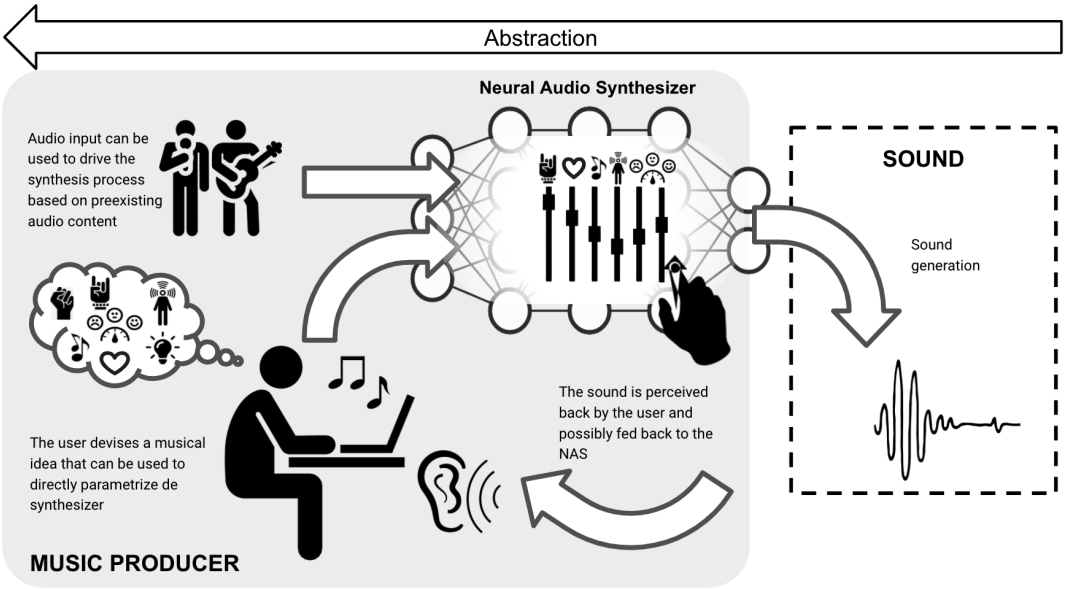


Figure 1.3 – Diagram of a DL-driven synthesizer.* The workflow is not any more linear as in Fig. 1.2. The synthesizer can be directly operated based on high-level controls and, additionally, it can be controlled based on preexisting audio content. Once the synthesizer is configured and sound is produced, the generated sound could be fed back to the synthesizer for its automated fine-tuning.

* Icon acknowledgements can be found in Appendix A.

pled with the complexity of music when rendered in its raw form. Music relies heavily on repetition to build structure and meaning at very different scales. Self-reference occurs not only on multiple timescales but frequency scales, from motifs to phrases to entire sections of a music piece, or even harmonic structure in the frequency domain across the different instruments. Thus, generative models of audio require large representational capacity distributed in time [Dieleman et al., 2018]. Most of the work applying neural networks to music generation has been devoted to symbolic representations such as MIDI or scores as these capture musical information in a more concise way [Briot et al., 2017, Pachet, 2002, Huang et al., 2019a, Hadjeres et al., 2017, Simon and Oore, 2017]. These representations, though, limit considerably the extent to which models can learn musically relevant nuances. For example, information about micro-timing variations, timbre, and precise dynamics (expressiveness) is harmed when music is represented as a score or a MIDI sequence, while audio waveforms retain all these relevant aspects. Models trained on such raw representations are also more general and can be applied to recordings of any set of instruments and non-musical audio signals such as speech. Previous work on music modeling in the raw audio domain [Donahue et al., 2019, Ai et al., 2018, van den Oord et al., 2016a] has shown that capturing local structure (such as timbre) is feasible. Recent work showed that capturing longer term structure (e.g., form, style) is also possible at the expense of model size, training data and generation time [Dhariwal et al., 2020].

1.2 Scope and Contributions

This thesis researches a broad set of applications of GANs to musical sound synthesis tasks. The main goal is to study and develop novel tools for music production that can offer the user intuitive and, simultaneously, inspiring means of sound manipulation, e.g., by controlling parameters that respond to perceptual properties of the sound or other high-level features. Further motivated in Chapter 2, an adversarial scheme is preferred over other generative modeling strategies (e.g., autoregressive, variational) as GANs evidence a good compromise between generation time, sample quality, and diversity. These considerations are of great importance in building commercially viable solutions that can run on a conventional computer while meeting the audio quality standards and real-time performance required in music production.

In order to steer the synthesis process, we are interested in conditioning the GAN on information describing musical or perceptual properties of the sound, namely features describing timbre (e.g., brightness, boominess), instrument categories (e.g., violin, piano), or sound event categories (e.g., sonar, mantra). Such annotations are obtained from 1) pre-existing hand-labeled information, 2) human-engineered feature extractors, or 3) representations learned using pre-trained automatic audio tagging systems. A sound synthesizer built upon such a model would have many applications and speed the music production workflow tremendously while making it more intuitive and user-driven.

The contributions of this thesis can be summarized in the following points:

1. We provide insights on the performance of several audio signal representations (e.g., raw waveform, spectrogram) for musical audio synthesis with GANs.

2. We study a variety of conditional generation tasks with GANs, exploring different sources of conditional information in order to provide to the user creative means of sound manipulation.

3. We demonstrate the capability of a single GAN architecture to model a wide variety of sound sources, from percussive and pitched instruments to chainsaw sounds and music.

4. A framework for generating audio with variable duration is proposed by conditioning the GAN on sequential features learned through self-supervised methods.

5. As a result of our research, we build two VST plugins for synthesizing high resolution (i.e., 44.1 kHz sample-rate) sounds such as drums in DrumGAN, or chainsaw sounds, in ChainsawGAN.

6. We perform a user study with professional musicians who created music with various in-house ML-driven tools for music production. We report their feedback and conclude thereby.

1.3 Ethical Considerations

Automation is increasingly becoming part of standard technological solutions and services (e.g., smartphones, cars, social networks), providing these with intelligence to plan and initiate actions autonomously. It is vital to raise awareness about the implications that such solutions have in creative activities like music. Deep learning-based generative models fall into one of two control principles: one where the user directly controls all aspects of the synthesis, akin to playing an instrument, and another whereby the system takes complete control. The deployment of fully automated audio generation systems can severely obscure and conceal the artist's role in the music creation process. While such systems are interesting from a scientific perspective to establish limits on technology, we believe that they do not add any value from a music innovation point of view. The author of this thesis is sensitive to the music community. This work does not hereby seek or claim to replace musicians in any way. On the contrary, we ought to develop tools that can democratize music production and help artists focus on creative aspects of music rather than technical ones.

Another critical aspect to be considered is the carbon footprint of training large-scale deep learning systems. We estimate to have emitted an average of 18 kg CO2 per model, assuming a standard carbon efficiency of the grid. This is equivalent to 63 Km driven by an average car or to 7.8 Kgs of coal burned. One of our aims for future work is to train efficient, compact models that require less amount of training time and that can run on a personal computer.

1.4 Document Organisation

The first three chapters of this thesis are dedicated to presenting the background and related work. Chapters 5 to 10 comprise a collection of six articles listed at the beginning of this document. The first four of these (chapters 5 to 8) constitute the main contribution of the thesis and cover several audio synthesis tasks with GANs. Chapters 9 and 10 are collaboration journal articles focusing respectively on: (i) audio enhancement of MP3-music using GANs and (ii) a critical perspective on AI-centered musical research in the context of musical innovation in contemporary popular music. More in detail, the rest of this thesis is organized as follows:

Chapter 2: Background. In this chapter we provide some basics on different deep learning strategies and various audio signal representations that are required for the proper understanding of this thesis.

Chapter 3: Related work. This chapter overviews all the relevant literature on neural audio synthesis as well as techniques prior the deep learning era.

Chapter 4: Methodology. This chapter describes the common methodologies followed throughout the experiments, describing in detail the GAN architecture, the datasets and the evaluation metrics.

Chapter 5: Comparing Representations for Audio Synthesis Using GANs. This chapter presents the results of our first work comparing representations for adversarial audio synthesis of tonal instrument sounds.

Chapter 6: DrumGAN: Synthesis of Drum Sounds with Timbral Feature Conditioning Using GANs. In this chapter we present DrumGAN, a GAN synthesizer of drum sounds that can be controlled based on perceptual features describing timbre. We also introduce a VST plugin implementation of DrumGAN capable of generating audio that meets music production standards in terms of quality.

Chapter 7: DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis with GANs. In this chapter we present DarkGAN, a framework for learning high-level feature controls in a GAN synthesizer by distilling knowledge from a pre-trained audio-tagging system.

Chapter 8: VQCPC-GAN: Variable-length Adversarial Audio Synthesis using Vector-Quantized Contrastive Predictive Coding. This chapter presents VQCPC-GAN, an adversarial framework for synthesizing variable-length audio by exploiting a self-supervised learning technique called Vector-Quantized Contrastive Predictive Coding (VQCPC).

Chapter 9: Stochastic Restoration of Heavily Compressed Musical Audio using GANs. In this chapter we describe a GAN that restores heavily compressed MP3 music to its high-quality, uncompressed form.

Chapter 10: On the Development and Practice of AI Technology for Contemporary Popular Music Production. This chapter formulates Sony CSL music team's vision on how to conduct music technology research in practice, involving the artist in the process and by releasing commercially viable music as a means for implicit validation. To this end, we report on our collaborations with professional musicians, in which we harmonize the use of AI-based tools with their music production workflow.

Chapter 11: General Conclusion. Finally, in this chapter some general conclusions are drawn, and, ultimately, we suggest directions for future research.

Chapter 2

Background

This thesis explores the generation of musical sounds using Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] by exploiting different sources of conditional information. Our goal is to provide insights into musically controllable adversarial audio synthesis and, ultimately, implement tools that can help professional artists in music production settings to enhance creativity while optimizing their workflows.

This chapter provides some ground knowledge on the various topics upon which this thesis builds. First, Section 2.1 briefly describes the main approaches to generative modeling (autoregressive, variational autoencoders, adversarial, and flow-based), paying special attention to GANs and some standard techniques such as *progressive growing* [Karras et al., 2017] or the *Wasserstein objective* [Arjovsky et al., 2017] (see Sec. 2.1.4). Section 2.2 introduces Knowledge Distillation (KD) [Hinton et al., 2015] and Dark Knowledge [Hinton et al., 2014] (KD is employed in Chapter 7 to perform data-free learning of semantically meaningful parameters in DarkGAN [Nistal et al., 2021b]). Next, Section 2.3 provides some background on self-supervised learning of sequences and introduces Vector-Quantized Contrastive Predictive Coding (VQCPC), which is used in Chapter 8 to address the problem of variable-length audio generation in GANs. Finally, in Section 2.4, we give an overview of some common representations of audio that will be compared in the context of audio synthesis with GANs (see Chapter 5).

2.1 Generative Neural Networks

Generative Neural Networks are a family of generative modeling strategies that employ neural networks to model the distribution $p_{\mathcal{X}}(\mathbf{x})$ of some random process producing observations from a dataset $\mathcal{X}$ with samples $\mathbf{x} \in \mathcal{X}$. Specifically, we focus our attention on likelihood-based models that learn via the principle of Maximum Likelihood Estimation (MLE): learning the model's parameters $\boldsymbol{\theta}$ so that the likelihood of observing the data $\boldsymbol{x} \in \mathcal{X}$ is maximized. This process is formulated as

$$\boldsymbol{\theta} := \arg\max_{\boldsymbol{\theta}} \sum_{\boldsymbol{x} \in \mathcal{X}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{2.1}$$

where $p_{\boldsymbol{\theta}}(x)$ is the likelihood or, in other words, the probability of $\boldsymbol{x} \in \mathcal{X}$ under the model with parameters $\boldsymbol{\theta}$. Note that this is done in the log space for

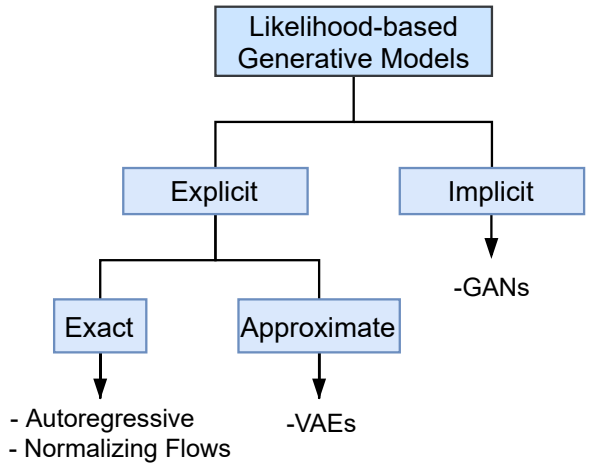


Figure 2.1 – Taxonomy of Generative Neural Networks [Goodfellow, 2017]. Methods differ in how they represent or approximate the likelihood. *Explicit* density estimation methods provide means to directly maximize the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Among these, the density may be computationally tractable, as in Autoregressive or Flow-based models, or it may be intractable, as in VAEs, meaning that it is necessary to make some approximations to maximize the likelihood. In contrast, *implicit* models do not explicitly represent a probability distribution over the data space. Instead, the model provides some way of interacting less directly with this probability distribution, typically by learning to draw samples from it. For example GANs can generate a sample $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ but cannot directly compute $p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

computational simplicity (i.e., products become additions) and numerical stability. This can also be thought as minimizing the KL-Divergence between the data distribution $p_X(x)$ and the model distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ [Goodfellow, 2017]. Once trained, generative models can be used to draw new samples $\hat{\boldsymbol{x}} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ as if they came from the training distribution $p_{\mathcal{X}}(\boldsymbol{x})$ (i.e., $p_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx p_{\mathcal{X}}(\boldsymbol{x})$). In order to gain control over the samples we draw from the generative model, we can feed a conditioning signal $\boldsymbol{c}$, containing side information about the kind of samples we want to generate. The model is then trained to fit the conditional likelihood distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{c})$ instead of $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. For simplicity, we refer to the unconditional distribution in the theoretical descriptions that follow this section.

Generative modeling strategies differ in the way they represent or approximate the likelihood (see Fig. 2.1). Two main approaches exist:

- **Explicit density estimation** models provide means of computing $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and can explicitly maximize the likelihood as formulated in (2.1). Among these, two different strategies exist to define a tractable expression. By carefully designing the neural network architecture, *exact* methods can define $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ so that it is computationally tractable. Popular examples of these are Neural Autoregressive Models (see 2.1.1) or Normalizing Flows (see 2.1.2). Other methods *approximate* the likelihood by e.g., maximizing a lower-bound as in Variational Autoencoders (see 2.1.3).

- **Implicit density estimation** models do not explicitly define the likelihood and, instead, offer indirect ways of interacting with $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. For example, Generative Adversarial Networks [Goodfellow et al., 2014] can be used to

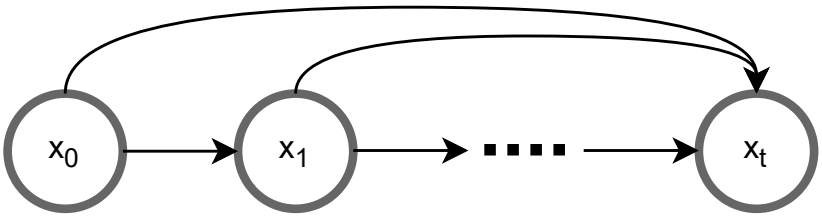


Figure 2.2 – Schematic of an autoregressive model. Each sample x_t depends on all the past samples $x_{<t}$

produce new samples imitating the dataset but cannot be used directly to infer the likelihood of an example.

These approaches exhibit specific run-time, diversity, and architectural trade-offs. For example, explicit models can be highly effective at capturing the diversity in the data since they directly optimize the log-likelihood, i.e., they have a mode-covering behavior [Dieleman, 2020]. However, they can be very slow to sample from, as in Autoregressive models, or produce blurred samples like in VAEs. In contrast, GANs can produce precise samples —potentially— at the expense of diversity, i.e., they have a mode-seeking behaviour [Dieleman, 2020]. In the following sections, we will deepen into these trade-offs as we briefly overview each generative strategy. It is not in the scope of this thesis to do an exhaustive review of generative methods. We recommend the following sources for a more in-depth overview: [Goodfellow, 2017, Briot et al., 2017, Dieleman, 2020, Bond-Taylor et al., 2021, Ji et al., 2020, Huzaifah and Wyse, 2020].

2.1.1 Neural Autoregressive Models

One of the challenges in explicit generative modeling is building expressive models that are also computationally tractable [van den Oord et al., 2016c]. Autoregressive approaches address this problem by treating $\boldsymbol{x} \in \mathcal{X}$ as a sequence $\boldsymbol{x} = (x_0, ..., x_t)$ (see Fig. 2.2). Then, the joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be decomposed into a product of conditional distributions using the probabilistic chain-rule as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod p_{\boldsymbol{\theta}}(x_t | x_0, ..., x_{t-1}), \tag{2.2}$$

where x_t is the t^{th} variable of $\boldsymbol{x}$ and $\boldsymbol{\theta}$ are the parameters of the neural autoregressive model. The conditional distributions are usually modelled with a neural network that receives $x_{<t}$ as input and outputs a distribution over possible x_t.

This approach seems to be a natural choice for time-series data such as audio signals, where each item x_t in the sequence corresponds to a specific amplitude value that the waveform takes at that specific (discrete) time step. Some popular neural networks employing this type of generative strategy on audio are WaveNet [van den Oord et al., 2016a], by using causal dilated convolutions, or SampleRNN [Mehri et al., 2017], which, instead, uses RNNs. Other approaches apply the autoregressive principle on other forms of data that are not naturally sequential

such as images [van den Oord et al., 2016c,b]. In Chapter 3 we review these and other approaches in detail.

Autoregressive models are very precise methods and can accurately capture correlations between the elements x_t in the sequential data. They also allow for fast inference (i.e. computing $p_{\boldsymbol{\theta}}(x_t|x_{<t})$). However, due to the sequential scheme, autoregressive models can only generate one sample at a time, becoming very slow to sample from (e.g., WaveNet [van den Oord et al., 2016a] can take minutes to generate just one second of audio). Also, autoregressive models can suffer from the exposure bias problem, i.e., the discrepancy between the conditional samples $x_{<t}$ used at training time, which come from the dataset, and those used for inference, which are generated by the model [Bengio et al.]. As a result, at generation time the error increases over time as the generated samples are fed back into the model.

2.1.2 Normalizing Flows

Other approaches for explicit density estimation that also provide an exact definition of the likelihood are Normalizing Flows (NF) [Rezende and Mohamed, 2015]. NFs are a family of procedures for learning flexible posterior distributions through an iterative procedure. The general idea is to start from an initial random variable $\boldsymbol{z_0}$ following a simple base distribution with known and computationally cheap probability density function, typically a standard Gaussian distribution. Then, a cascade of invertible and differentiable transformations $g : f_T \circ f_{T-1} \circ ... \circ f_1 \circ f_0$ is applied using the change of variables formula to produce a sample from the dataset as $\boldsymbol{x} = g(\boldsymbol{z_0})$ (see Fig. 2.3). The log-likelihood can then be expressed as

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log p_{\boldsymbol{z_0}}(\boldsymbol{z}) - \sum_{t=1}^{T} \log \left| \det \frac{\partial f_t}{\partial f_{t-1}} \right| = \log p_{\boldsymbol{z_0}}(\boldsymbol{z}) - \log \left| \det J_g(\boldsymbol{z_0}) \right|, \quad (2.3)$$

where the Jacobian $J_g(\boldsymbol{z_0})$ is a matrix of all partial derivatives of g w.r.t. $\boldsymbol{z_0}$. The density $p_{\boldsymbol{\theta}}$ is tractable if the density $p_{\boldsymbol{z}}$ and the determinant of the Jacobian of g are tractable. We can think of this process as follows: the density of the base distribution $\boldsymbol{z_0}$ gets molded by each transformation in g in order to produce an increasingly richer output distribution from which to sample $\boldsymbol{x}$.

Many types of NFs exist satisfying the invertibility and tractability requirements for g and J_g respectevely, e.g., Planar Flows (PFs), Masked Autoregressive Flows (MAFs), Inverse Autoregressive Flows (IAFs). Each formulation exhibits different inference-sampling time trade-offs, e.g., IAFs can generate samples fast although computing the likelihood of new data points is slow [Bond-Taylor et al., 2021]. Also, the invertibility requirement for g enforces the variables $\boldsymbol{z_0}$ to have the same dimensionality as $\boldsymbol{x}$, constraining the model's architecture and parameter efficiency. As a result, flow-based models require rather deep architectures to be effective.

2.1.3 Variational Autoencoders

To overcome some of the disadvantages imposed by the design requirements of models with tractable density functions (e.g., slow sampling time in NAMs, pa-

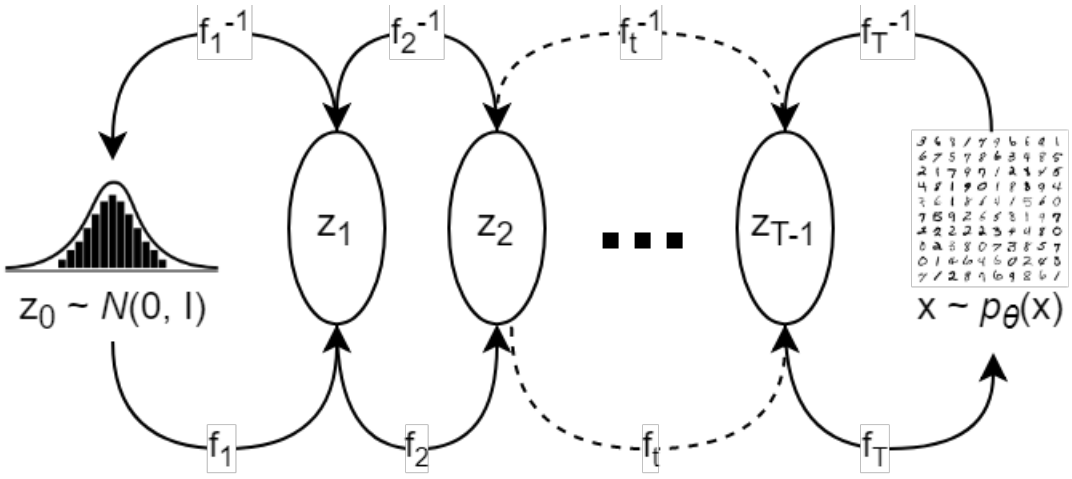


Figure 2.3 – Schematic depiction of Normalizing Flows. The term 'flow' refers to the stream followed by samples $\boldsymbol{z_0} \sim \mathcal{N}(0, \mathbf{I})$ as they are molded by the sequence of transformations $f_1, ..., f_T$. The term 'normalizing' refers to the fact that the probability mass is preserved throughout the transformations. Note that the last iterate in g results in a more flexible distributions of $\boldsymbol{z_{T-1}}$ over the values of the data $\boldsymbol{x}$ (being $\boldsymbol{z_T} = \boldsymbol{x}$ and $\boldsymbol{z_t} = f_t(\boldsymbol{z_{t-1}})$).

rameter inefficiency in NFs) and still not run into intractability issues, some models use some approximations to maximize the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Variational methods define a lower bound $\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq \log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and provide an analytical approximation of the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ to perform inference.

Variational Autoencoders [Kingma and Welling, 2014] learn two neural networks jointly: an inference model or encoder and a generative neural network or decoder. The encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is a neural network with parameters $\boldsymbol{\phi}$ that maps $\boldsymbol{x}$ into a compressed representation $\boldsymbol{z}$ and approximates the true posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. The decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is a neural network with parameters $\boldsymbol{\theta}$ that regenerates an approximation $\hat{\boldsymbol{x}}$ from the encoding. In plain Autoencoders (AEs), the latent variable $\boldsymbol{z}$ follows an unknown probability distribution and, the computation of the generative models' true posterior density $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is intractable as a result of the combinatorially wide $\boldsymbol{z}$ space. VAEs simplify this by restricting $\boldsymbol{z}$ to follow some prior distribution $\boldsymbol{z} \sim p(\boldsymbol{z})$ with a known density function. The Evidenced Lower Bound (ELBO) to be maximized is formulated as

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) &\geq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \mathbf{x}) \\ &= -D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\big(\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\big). \end{aligned} \tag{2.4}$$

Here, $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \mathbf{x})$ is the variational lower bound to optimize and D_{KL} stands for the Kullback–Leibler divergence (KLD). The prior over the latent variables $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is usually set to be the centred isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix. The usual choice of $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is $\mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \boldsymbol{\sigma^2}(\mathbf{x}) * \mathbf{I})$, so that $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}))$ can be calculated in closed form. In practice, $\mu(\mathbf{x})$ and $\boldsymbol{\sigma^2}(\mathbf{x})$ are learned from the observed data via the encoder neural networks. The expectation term accounts for the reconstruction loss in (2.4), where the role of the decoder is to transform latent variables $\boldsymbol{z}$ to reconstruct $\hat{\boldsymbol{x}}$.

The main drawback of VAEs is that the gap between the ELBO and the true

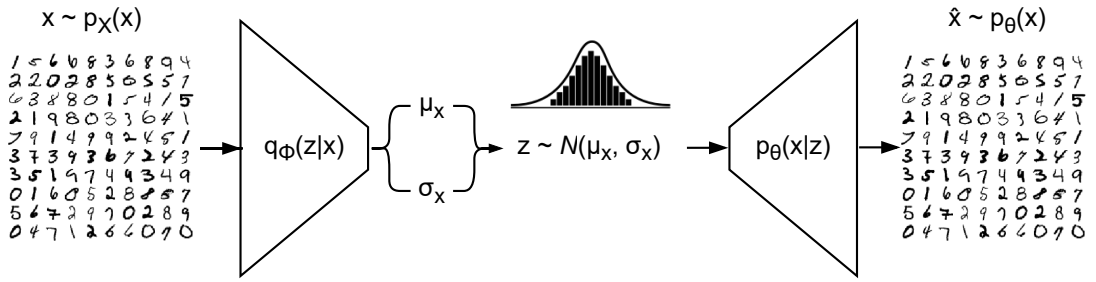


Figure 2.4 – Schematic depiction of Variational Autoencoders (VAEs).

likelihood can result in poor quality samples if the posterior or prior distributions are too simple. Also, if the decoder network is too powerful, VAEs can suffer from mode failure, where the decoder may ignore the latent codes $\boldsymbol{z}$ and generate outputs arbitrarily. In general, VAEs often obtain good likelihood and can perform inference precisely, yet, in practice, they produce lower-quality samples than other methods.

2.1.4 Generative Adversarial Networks

In this section, we present the standard adversarial formulation of Generative Adversarial Networks (GANs) and some important subsequent versions: the Wasserstein GAN and the Progressive Growing GAN. Additionally, we review some standard techniques for training GANs such as *mini-batch standard deviation*, *equalized learning*, and *pixel-wise feature normalization*.

The basic GAN framework

Generative Adversarial Networks (GAN) are a family of training procedures inspired by game theory that circumvent the difficulty of having to approximate intractable probabilistic computations aroused in maximum likelihood methods. In the adversarial framework, a generative model competes against a discriminative adversary that learns to distinguish whether a sample is real or fake [Goodfellow et al., 2014]. The generative network, or Generator (G), implicitly models a distribution $p_{\mathcal{X}}$ over some real data $\boldsymbol{x} \in \mathcal{X}$, which we will refer to as p_r, by learning the push-forward mapping of an input noise $p_{\boldsymbol{z}}$ to data space as $G_{\boldsymbol{\theta}}(\boldsymbol{z})$, where $G_{\boldsymbol{\theta}}$ is a neural network implementing a differentiable function with parameters $\boldsymbol{\theta}$. Inversely, the discriminator $D_{\boldsymbol{\beta}}(\boldsymbol{x})$, with parameters $\boldsymbol{\beta}$ is trained to output a single scalar indicating whether the input comes from the real distribution p_r or from the generated distribution $G_{\boldsymbol{\theta}}(\boldsymbol{z}) \sim p_g$. Simultaneously, $G_{\boldsymbol{\theta}}$ is trained to produce samples that are identified as real by the discriminator. Competition drives both networks until an equilibrium point is reached and the generated examples are indistinguishable from the original data. In other words, $D_{\boldsymbol{\beta}}$ and $G_{\boldsymbol{\theta}}$ play the following two-player minimax game with value function $V(G_{\boldsymbol{\theta}}, D_{\boldsymbol{\beta}})$ [Goodfellow et al., 2014]:

$$\min_{G_{\boldsymbol{\theta}}} \max_{D_{\boldsymbol{\beta}}} V(D_{\boldsymbol{\beta}}, G_{\boldsymbol{\theta}}) = E_{\boldsymbol{x}\sim p_r}[\log D_{\boldsymbol{\beta}}(\boldsymbol{x})] + E_{\boldsymbol{x}\sim p_g}[1 - \log D_{\boldsymbol{\beta}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))], \qquad (2.5)$$

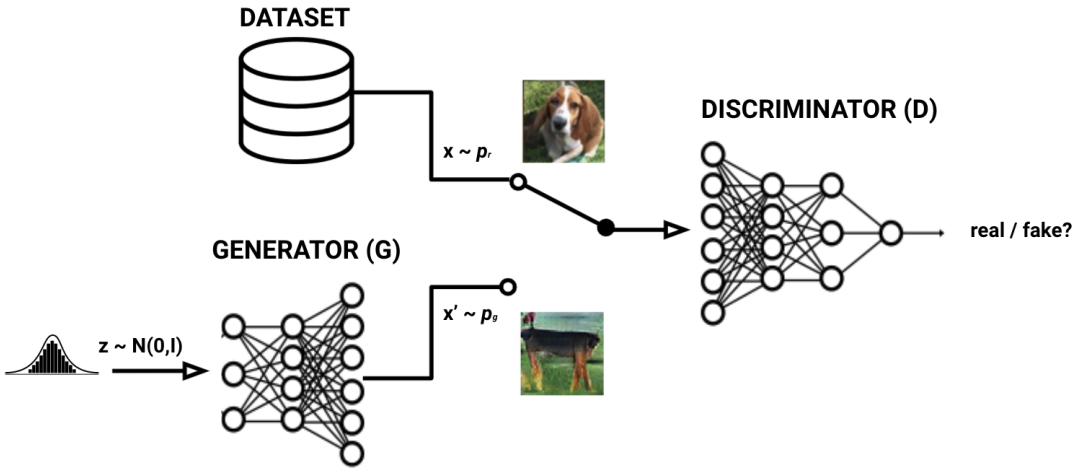


Figure 2.5 – GAN framework

where $\min_{G_{\boldsymbol{\theta}}} \max_{D_{\boldsymbol{\beta}}} V(D_{\boldsymbol{\beta}}, G_{\boldsymbol{\theta}})$ indicates that the parameters of $G_{\boldsymbol{\theta}}$ are optimized to minimize this loss, and the parameters of $D_{\boldsymbol{\beta}}$ are optimized to maximize it. Note that the optimization of $G_{\boldsymbol{\theta}}$ only affects the second term in (2.5), resulting in a maximization of $D_{\boldsymbol{\beta}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))$.

Wasserstein GANs

One of the main drawbacks of the original GAN setting, where the objective of $D_{\boldsymbol{\beta}}$ is a binary classification problem, is that the cost function is potentially not continuous with respect to the generator's parameters, leading to training difficulty. Instead, Arjovsky et al. [2017] propose the Earth-Mover or Wasserstein-1 distance $W(p_g, p_r)$, which is informally defined as the minimum cost of transporting mass in order to transform the distribution p_g into the distribution p_r, where the cost is mass times transport distance. Under mild assumptions, $W(p_g, p_r)$ is continuous everywhere and differentiable almost everywhere. The Wasserstein GAN (WGAN) value function is constructed using the Kantorovich-Rubinstein duality [Villani, 2008] to obtain

$$\min_{G_{\boldsymbol{\theta}}} \max_{D_{\boldsymbol{\beta}}} \Gamma(D_{\boldsymbol{\beta}}, G_{\boldsymbol{\theta}}) = E_{\boldsymbol{x}\sim p_r}[D_{\boldsymbol{\beta}}(\boldsymbol{x})] - E_{\boldsymbol{x}\sim p_g}[D_{\boldsymbol{\beta}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))], \tag{2.6}$$

where $D_{\boldsymbol{\beta}}$ is the set of 1-Lipschitz functions and p_g is once again the model distribution implicitly defined by $\hat{\boldsymbol{x}} = G_{\boldsymbol{\theta}}(\boldsymbol{z})$, with $\boldsymbol{z} \sim p(\boldsymbol{z})$. In that case, under an optimal discriminator or critic[1] [Arjovsky et al., 2017], minimizing the value function with respect to the generator parameters minimizes $W(p_r, p_g)$. The WGAN value function results in a critic function whose gradient with respect to its input is better behaved than its GAN counterpart, making optimization of the generator easier. Empirically, it was also observed that the WGAN value function appears to correlate with sample quality, which is not the case for traditional GANs [Goodfellow et al., 2014]. Enforcing the Lipschitz constraint on the critic

[1] $D_{\boldsymbol{\beta}}$ is not trained anymore in a binary classification task (i.e., real/fake) but to assign high and low Wasserstein distances to generated and real data, respectively. Therefore, the Discriminator is sometimes referred to as a *Critic*. For simplification, we still use the Discriminator term.

was originally accomplished by clipping the weights of the critic to lie within a compact space $[-c, c]$ [Arjovsky et al., 2017]. The set of functions satisfying this constraint is a subset of the k-Lipschitz functions for some k, which depends on c and the critic architecture. An alternative way to enforce the Lipschitz constraint is to constrain the gradient norm of the critic's output with respect to its input by means of Gradient Penalty (GP). GP introduces a penalty on D's gradient norm for random samples $\hat{\boldsymbol{x}} \sim p_g$ to circumvent tractability issues [Gulrajani et al., 2017]. Then, the GP-WGAN's objective is defined as

$$L = E_{\boldsymbol{x}\sim p_g}[D_{\boldsymbol{\beta}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))] - E_{\boldsymbol{x}\sim p_r}[D_{\boldsymbol{\beta}}(\boldsymbol{x})] + \lambda E_{\boldsymbol{x}\sim p_g}[(\|\nabla_{\hat{\boldsymbol{x}}} D_{\boldsymbol{\beta}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})\|_2 - 1)^2], \tag{2.7}$$

where λ is the penalty coefficient and it is typically set to 10, which was found to work well across a variety of architectures and datasets [Gulrajani et al., 2017]. Following the original GP-WGAN implementation, normalization methods that introduce correlations between the examples in the batch (e.g. batch normalization) are avoided in favour of layer-wise feature normalization as explained later in this section.

Progressive Growing of GANs

Progressive growing [Karras et al., 2017] is a training methodology for GANs where low-resolution data (e.g., down-sampled images or spectrograms) is used at the beginning of training and then progressively scaled up by adding convolutional and up-sampling layers to the networks (see Fig. 2.6). This incremental procedure allows the network to first discover large-scale structure in the data and then progressively shift attention towards finer-grain detail instead of having to learn the full resolution data directly. Generator and discriminator networks are commonly mirrored versions of each other and always grow synchronously. All existing layers in both networks remain trainable throughout the training process. When new layers are added to the networks, they are faded in smoothly, as illustrated in Figure 2.7. This reduces any possible perturbations to the already well-trained, smaller-resolution layers. Progressive training has many benefits, including improved training stability, generation diversity, and a reduced training time.

Mini-Batch Standard Deviation

GANs are prone to cover only a part of the training data variance. Mini-batch discrimination [Salimans et al., 2016] is a way of alleviating such a mode failure by providing D with additional information, namely statistics of the respective minibatch to simplify the discrimination of real and fake batches. To that end, on the last layers of D, we compute feature statistics across the batch dimension. First, the standard deviation for each feature map in each spatial location (i.e., the height and width dimensions of the convolutional tensor) is estimated over the minibatch and averaged over all the features and spatial locations to arrive at a single value. Then, this value is replicated and concatenated to all spatial locations and over the minibatch, yielding one additional (constant) feature map.

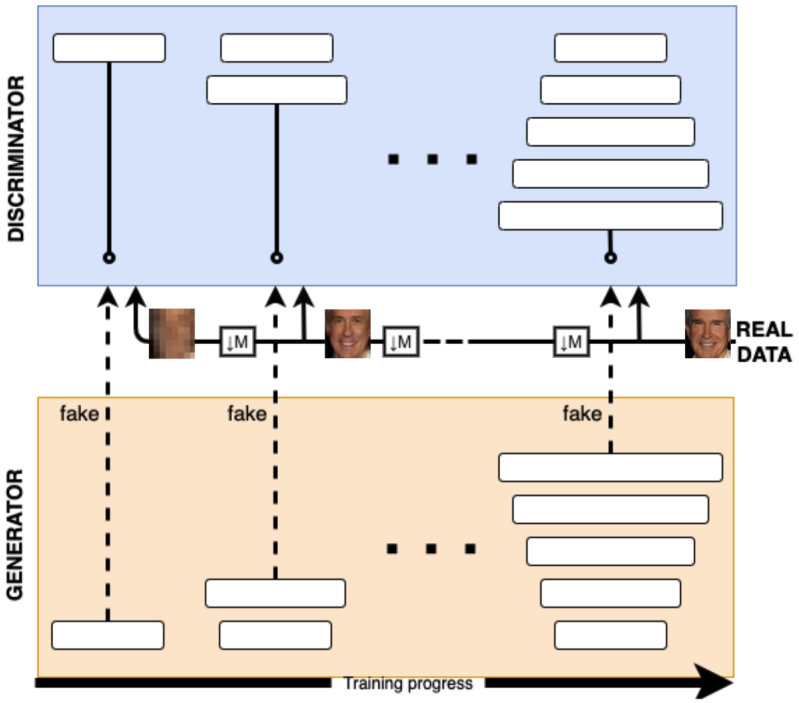


Figure 2.6 – Progressive Growing of GANs as illustrated in [Karras et al., 2017]. Training starts with both G and D having a low spatial resolution of 4×4 pixels and, as training progresses, new layers containing up-sampling blocks are added to G and D, increasing the spatial resolution of the generated images.

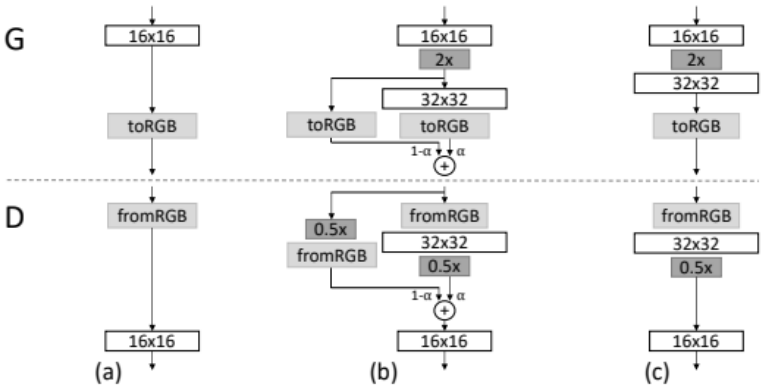


Figure 2.7 – Layer fading as illustrated by Karras et al. [2017]. The output of every new layer in G and D is interpolated by a factor α with the previous layer's output. This transition from low-resolution data, e.g., 16×16 pixel images (a), to high-resolution data, e.g., 32×32 pixel images (c), is illustrated in the transition (b), where the layers that operate on the new resolution are treated as a residual block with α increasing linearly from 0 to 1. Here $2\times$ and $0.5\times$ refer to doubling and halving the resolution using nearest neighbor up-sampling and average pooling, respectively, for G and D. *toRGB* is a 1×1 convolutional layer that projects feature maps to the data space (e.g., RGB channels of an image or magnitude, and phase components of a spectrogram) and *fromRGB* does the reverse. When training the discriminator on a *real* batch, the data is down-scaled to match the current resolution of the network.

The discriminator can use these feature statistics internally, encouraging G to generate image batches with the same statistics as the real data batches.

Equalized Learning-Rate

GANs are sensible to instabilities in the signal magnitudes as a result of unhealthy competition between G and D. In order to alleviate this problem, dynamic weight initialization was proposed in [Karras et al., 2017]. First, weights are initialized to $\mathcal{N}(0, 1)$ and then explicitly scaled at run-time as $w^i = w^i/c$, where w^i are the weights and c is the per-layer normalization constant from He's initializer [He et al., 2015]. The benefit of doing this dynamically instead of during initialization relates to the scale-invariance in adaptive stochastic gradient descent methods such as RMSProp and Adam. These methods normalize the gradient update by the estimated standard deviation, thus making the update independent of the scale of the parameter. As a result, those parameters exhibiting large dynamic range will take longer to adjust than others. Using an equalized learning-rate ensures that the dynamic range, and thus the learning speed, is the same for all weights.

Pixel-wise Feature Normalization

To further constrain the magnitudes in G and D and prevent signals from spiraling out of control, feature vectors are normalized in each location to unit length after each convolutional layer in G as

$$x = x_{ncwh} / \sqrt{C^{-1} \sum_{C} x_{ncwh}^2}, \tag{2.8}$$

where n, c, w and h are the batch, channel, width and height respectively and C is the total number of channels.

2.1.5 Discussion

As we introduced in Chapter 1, we find Generative Adversarial Networks (GANs) better suited than other generative strategies for the task under consideration. We highlighted some important prerequisites that a potential ML-driven audio synthesizer should meet: fast generation time and high audio quality. Some important points influencing the choice of GANs over other generative models are:

- Neural Autoregressive Models (NAMs) and Normalizing Flows (NFs) can produce very expressive and precise samples and provide an exact estimate of the likelihood of a sample. However, NAMs are slow at sampling time and NFs require very large models to capture rich dependencies in the data.

- Variational Autoencoders (VAEs) provide a more efficient and yet precise way to perform inference. However, due to the variational approximation, they produce blurred samples with lower quality than other approaches. Also, if the generative network is too powerful, VAEs can suffer from posterior collapse.

- GANs can be sampled in parallel, and they disregard the inference model. Therefore, they can be sampled faster and more efficiently than NAMs or NFs and generate samples with considerably higher quality than VAEs.

- GANs design of the generator function has very few restrictions as opposed to NAMs, which require autoregressive computations or NFs, which require the invertibility of the generator as well as require a latent code $\boldsymbol{z}$ with the same dimension as the data $\boldsymbol{x}$.

An important drawback of GANs is that they require large amounts of data to approximate the data distribution accurately. Also, they fail to capture rich variance given the mode-seeking behavior of the adversarial objective. Morever, GANs can be extremely difficult to train due to unhealthy competition between G and D. Nonetheless, we believe that the adversarial scheme is a promising approach to develop novel audio synthesizers complying with the generation time and audio quality standards in music production contexts. We hope to justify further this decision in Chapter 3, where we provide a broad review of generative neural networks applied to audio and music specifically.

2.2 Knowledge Distillation

High-performing models are often built upon classifier ensembles that aggregate their predictions to improve the overall accuracy. Despite having excellent performance, these models tend to be large and slow, impeding their use in memory-limited and real-time environments. Different methods exist for optimizing memory consumption and reducing the size of large models or ensembles, e.g., pruning, transfer learning, or quantization. Model compression allows to transfer the function learned by a teacher ensemble or a single large discriminative model into a compact, faster student model exhibiting comparable performance [Bucila et al., 2006]. Instead of training the student model directly on a hand-labeled categorical dataset, this method employs a pre-trained teacher model to re-label the dataset and then train the compact neural network on this teacher-labeled dataset, using the raw predictions as the target. This training framework was shown to yield efficient models which perform better than if they had been trained on the hand-labeled dataset in a variety of discriminative tasks [Bucila et al., 2006, Ba and Caruana, 2014, Li et al., 2014]. Model compression was further extended and formalized into the general Knowledge Distillation (KD) framework [Hinton et al., 2015]. This section provides a brief introduction to the knowledge distillation framework and the concept of *dark knowledge* that we employ in Chapter 7 as a means to learn interpretable controls in a GAN-driven synthesizer.

2.2.1 Multi-Label KD

Multi-label classifiers typically produce a probability distribution over a set of classes by using a *sigmoid* output layer that converts the so-called logit (the NN output before the activation function), z_i, computed for the ith class into a

probability q_i as

$$q_i = \frac{1}{1 + e^{-\frac{z_i}{T}}}, \tag{2.9}$$

where T is a temperature that is typically set to 1. In Knowledge Distillation (KD), knowledge is transferred to the distilled model by training it on the teacher-labeled data, using a higher temperature. By that, the distribution gets "compressed," emphasizing lower probability values. The same (higher) temperature is used while training the distilled model, but the temperature is set back to 1 after training. As for cost function, the binary cross-entropy is used as

$$H_s(q) = -\frac{1}{N}\sum_{i=1}^{N} p_i \log(q_i) + (1 - p_i)\log(1 - q_i), \tag{2.10}$$

where N is the number of attributes, p_i are the soft-labels predicted by the teacher, and q_i is the probability predicted by the student model for the $i - th$ class.

2.2.2 Dark Knowledge

In the seminal work on Knowledge Distillation (KD) [Hinton et al., 2015], the authors demonstrate that the improved performance of smaller models is due to the implicit information existent in the teacher's output probabilities (i.e., soft labels). As opposed to hard labels, soft labels contain probability values for all of the output classes. The relative probability values that a specific data instance takes for each class contain information about how the teacher generalized the discriminative task. This hidden information existent in the relative probability values was termed *dark knowledge* [Hinton et al., 2014]. An interesting observation by Hinton et al. [2015] is that the student model was able to gather information about categories that were not explicitly present in the transfer learning set.

Further on in this thesis, we employ this principle to transfer knowledge from a pre-trained audio neural network [Kong et al., 2020b] to a GAN synthesizer trained on tonal sounds from the NSynth dataset (see Chapter 7). This way, semantically meaningful controls can be learned on the GAN without the need for manual annotations. We also show in this work that the Dark Knowledge implicit in the teacher-labeled features indeed helps the GAN to learn consistent feature controls over abstract attributes that are not necessarily represented in the training data.

2.3 Self-Supervised Learning of Sequences

Representation learning is a framework for extracting general-purpose, useful information that explains the underlying factors of variation of data and can help improve in downstream tasks such as classification [Bengio et al., 2013]. Among unsupervised training schemes, in self-supervised learning, training is done via a proxy task, so-called pretext task, formulated directly on the learned representations and without requiring manually annotated labels. A prevalent self-supervised task is *contrastive learning*. This task relies on contrasting multiple,

slightly different versions of an example by using different sampling strategies. Recently, contrastive approaches have been used in audio-related tasks to learn transformations that map augmented versions of a given audio signal (e.g., reverb, additive noise) to the same latent space while pushing them away from different augmented audio signals [Verma and III, 2020, Spijkervet and Burgoyne, 2021]. Augmentation strategies can be circumvented by, instead, relying on similar signal pairs extracted from the same audio clip [Saeed et al., 2020]. Some works have employed this technique for learning representations that capture information from multiple audio formats [Wang and van den Oord, 2021].

In this thesis, we employ Vector-Quantized Contrastive Predictive Coding (VQCPC), a contrastive approach for learning discrete feature representations of sequences. VQCPC is employed in Chapter 8 to condition the GAN on such discrete features, enabling the generation of sounds with variable duration as well as the manipulation of local features. In what follows, we describe the building blocks that compose this technique.

2.3.1 Contrastive Predictive Coding

Contrastive Predictive Coding (CPC) is a self-supervised representation learning technique for extracting compact, low-dimensional sequences of latent codes from high-dimensional signals [van den Oord et al., 2018b]. Given an input sequence $\boldsymbol{x} = [x_1, ..., x_L]$ with length L, an encoder f_{enc} maps each element x_i into a real-valued embedding vector $\boldsymbol{z_i} = f_{\text{enc}}(x_i) \in R^{d_z}$. Next, an autoregressive model f_{ar} summarizes past and present context of the embeddings $\boldsymbol{z_{\leq t}}$ into a single context vector $\boldsymbol{h_t} = f_{\text{ar}}(\boldsymbol{z_{\leq t}}) \in R^{d_h}$.

The encoder and autoregressive model are trained to minimize the Information Noise Contrastive Estimation (InfoNCE) loss. Minimizing the InfoNCE loss is equivalent to maximizing the mutual information between the context vector $\boldsymbol{h_t}$ and future encodings $\boldsymbol{z_{t+k}} = f_{enc}(x_{t+k}), \forall k \in [1, K]$, where K is the number of future predictions [van den Oord et al., 2018b]. Formally, given an entry of the dataset $\boldsymbol{x}$, the model has to identify the encoding obtained from the true x_{t+k}, so-called *positive example*, from those obtained from a set of so-called *negative examples*, drawn from the dataset by following a specific negative sampling strategy. Defining $\mathcal{S}$ as the set containing $N-1$ negative examples, as well as the single positive example, the InfoNCE loss is defined as

$$\mathcal{L}_{\text{NCE}}(x_t) = -\sum_{k=1}^{K} \underset{\mathcal{S}}{E} \left[\log \frac{f_k(x_{t+k}, \boldsymbol{h_t})}{\sum_{s \in \mathcal{S}} f_k(s, \boldsymbol{h_t})} \right], \tag{2.11}$$

where $E[\cdot]$ denotes expectation and $f_k(a, b) := \exp(f_{\text{enc}}(a)^\intercal W_k b)$ is a simple log-bilinear model with the W_k being k trainable d x d matrices.

2.3.2 Vector Quantization

Vector Quantization (VQ) [van den Oord et al., 2017] consists in approximating the elements of a continuous vector space R^{d_c} by the closest element in a finite set of vectors or centroids $\mathcal{C} = \{\boldsymbol{c^1}, ..., \boldsymbol{c^C}\}$ lying in the same space R^{d_c}. Here, given a trainable set of codes $\mathcal{C}$, the quantization of an input vector $\boldsymbol{z}$ is given by

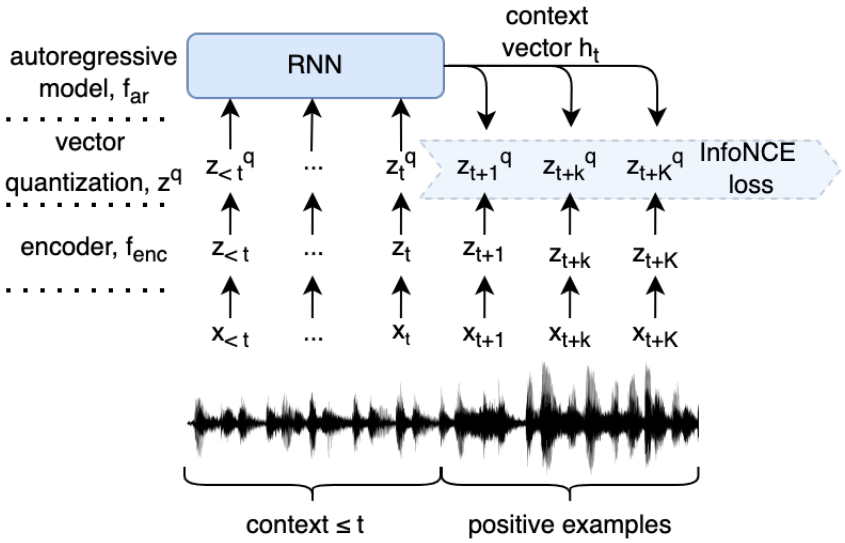


Figure 2.8 – Schematic of the VQCPC training framework applied to audio in analogy to that described by Hadjeres and Crestel [2020] for symbolic music.

its closest centroid

$$c(\boldsymbol{z}) := \text{argmin}_{\boldsymbol{c}\in\mathcal{C}}||\boldsymbol{z}-\boldsymbol{c}||_2. \tag{2.12}$$

This layer is not differentiable due to the *argmin* operator so the stop gradient operator *sg* is used to enable back-propagation [van den Oord et al., 2017]. Given an input vector $\boldsymbol{z}$, the VQ layer is then defined as

$$z^q(\boldsymbol{z}) := sg[c(\boldsymbol{z})-\boldsymbol{z}]+\boldsymbol{z}. \tag{2.13}$$

The centroid positions and the non-quantized values $\boldsymbol{z}$ are updated incrementally by minimizing

$$\mathcal{L}_{\text{VQ}}(\boldsymbol{z},\mathcal{C}) = \sum_{\boldsymbol{c}\in\mathcal{C}} \delta^{\boldsymbol{c}}_{z_q(\boldsymbol{z})}((||sg[\boldsymbol{z}]-\boldsymbol{c}||_2)^2+\beta(||\boldsymbol{z}-sg[\boldsymbol{c}]||_2)^2), \tag{2.14}$$

where $\delta^a_b = 1 \iff a = b$ and zero otherwise, and β is a parameter to control the trade-off between the two terms. In a nutshell, this loss encourages non-quantized values $\boldsymbol{z}$ to be close to their assigned centroid.

2.3.3 Vector Quantized Contrastive Predictive Coding

In Fig. 2.8 we depict the VQCPC framework combining the VQ and CPC blocks. The VQ [van den Oord et al., 2017] bottleneck is introduced on top of the encoder f_{enc} and before the context encoder f_{ar}. At test time, we remove f_{ar}, and encode new elements x_i as

$$\text{VQCPC}(x_i) := z^q(f_{\text{enc}}(x_i)) \in \mathcal{C}, \tag{2.15}$$

where the *codebook* $\mathcal{C}$ is a set with C centroids partitioning the embedding space R^{d_c}.

2.4 Audio Representations

Audio signals consist of large amounts of data in which relevant information for a specific task is often hidden and spread over large time spans [Dieleman et al., 2018]. Neural Networks can benefit from feeding in specific representations of the audio data where information is structured in a suitable way for the specific architecture, or where few coefficients compress the information of interest. Different representations may yield different trade-offs between training/sampling times, architecture size, and generation quality. In the following, we review some common audio representations that we will compare in the context of audio synthesis with GANs (see Chapter 5), highlighting their strengths and weaknesses for the specific task. Except stated otherwise, we compute the audio representations using Librosa [McFee et al., 2020].

2.4.1 Waveform

The **raw audio waveform** consists of a sequence of numerical samples $\boldsymbol{x} = [x_1, ..., x_t]$ that specify the amplitude values of the signal at time steps t. Using this representation as input is challenging for generative modeling, particularly in the case of music signals [Dieleman et al., 2018]. On the other hand, it enables neural networks to build the representation that better suits a specific task without any prior assumptions.

2.4.2 Short-Time Fourier Transform

The **Short-Time Fourier Transform (STFT)** decomposes a signal as a weighted sum of complex sinusoidal basis vectors $\boldsymbol{\phi_{k,t}}$ with linearly spaced center frequencies as

$$\boldsymbol{\phi_{k,t}} = \frac{1}{T} \exp\left(\frac{2\pi k j}{T} t\right), \tag{2.16}$$

where j is the imaginary unit, k is the bin number, t is time, and T is the window size in samples. The STFT unveils the time-frequency structure of an audio signal under the assumption that it is stationary within one frame (typically of length 512-2048 samples), which is often a good approximation for natural sounds, such as speech or music. The complex STFT coefficients are typically further decomposed into magnitude and phase components. The latter are typically noisy, which makes them difficult for neural networks to model. This problem is mitigated by using the Instantaneous Frequency (IF), which provides a measure of the rate of change of the phase information over time [Boashash, 1992]. The IF, however, only works well for quite tonal sounds, as they are not capable of modeling steep transients in the signal (the phases may not align well, as they are considered independent). The STFT transform is cheap to compute and perfectly invertible, which makes it popular for audio synthesis of tonal sounds [Engel et al., 2019, Marafioti et al., 2019]. The complex STFT has also been used for sound texture synthesis with CNNs by Caracalla and Roebel [2020].

2.4.3 Constant-Q Transform

The **Constant-Q Transform (CQT)** decomposes a signal as a weighted sum of tonal-spaced filters, where each filter is equivalent to a subdivision of an octave [Brown, 1991]. As opposed to the STFT where the central frequencies of the basis vectors are linearly spaced, in the CQT the filters are geometrically spaced as $f_k = (2^{\frac{1}{b}})^k f_{min}$, where f_k denotes the frequency of the $k-th$ spectral component, b is the number of filters per octave, and f_{min} is the central frequency of filter $k = 0$. The Q value is the ratio of center frequency to bandwidth and is meant to be constant

$$Q = \frac{f_k}{\Delta f_k} = \frac{f_k}{f_{k+1} - f_k} = (2^{\frac{1}{b}} - 1)^{-1}. \tag{2.17}$$

Similarly to the Fourier Transform, the CQT has a basis matrix given by

$$\boldsymbol{\phi_{k,t}} = \frac{1}{T_k} \exp\big(j\frac{2\pi Q}{T_k}t\big), \tag{2.18}$$

where the sequence length or window size T_k is now a function of the component k.

This musically motivated spacing of frequencies enables representing pitch transpositions as simple shifts along the frequency axis, which is well-aligned with the equivariance property of the convolution operation. The CQT transform has been used as a representation for Music Information Retrieval [Lidy, 2016] and some works have exploited it for audio synthesis [Esling et al., 2018b]. The main disadvantage of CQT over STFT is the loss of perceptual reconstruction quality due to the frequency scaling in lower frequencies [Barry and Kim, 2018].

2.4.4 Mel Spectrogram

The **Mel spectrogram** compresses the STFT in frequency axis by projecting it into a perceptually inspired frequency scale, called the Mel-scale [Stevens et al., 1937] as

$$M(f) = 1125 \ln\big(1 + \frac{f}{700}\big). \tag{2.19}$$

Mel discards the phase information, so we use the iterative method from Griffin and Lim [1983] to recover the phase for synthesis. We refer to this representation as *mel* throughout our experiments. The mapping can be used to create a filter bank for projecting the magnitude STFT onto a perceptually optimal smaller number of channels. Because the Mel spectrogram represents spectral content of the STFT in a perceptually uniform manner, it has been a popular choice for state-of-the-art neural networks trained on large corpora of musical audio [Barry and Kim, 2018].

2.4.5 Mel Frequency Cepstral Coefficients

The **Mel Frequency Cepstral Coefficients (MFCC)** [Davis and Mermelstein, 1980] provide a compact representation of the spectral envelope of an audio signal. Originally developed for speech recognition, they are now widely used in musical applications, as they capture perceptually meaningful musical timbre features

[Ravelli et al., 2010]. For synthesis, we invert MFCC to the Mel scale and use Griffin-Lim algorithm to recover the phase.

Chapter 3

Related Work

Many works have applied deep generative methods to address general audio synthesis. In Chapter 2, we provided a theoretical introduction to each of these methods, categorised into *exact*, *approximate*, and *implicit*, depending on the way they estimate the probability distribution of a given training dataset. From the *exact* family, we highlighted Neural Autoregressive Models (NAMs) and Normalizing Flows (NFs), from the *approximate* methods, we reviewed Variational Autoencoders (VAEs), and, from the *implicit* strategies, we described Generative Adversarial Networks (GANs). This chapter provides a broad overview of the state-of-the-art works applying such generative strategies to various audio synthesis tasks (see Section 3.1). Special attention is paid to those works focused on musical audio and the control they offer over the generated sound. We also review other audio modeling techniques based on Digital Signal Processing (DSP) and that, while not relying on deep learning, have been used to study intuitive and controllable audio synthesis (see Section 3.2). In Section 3.3, we conclude with some discussion thereof.

3.1 Neural Audio Synthesizers

Neural Audio Synthesizers are generative models that learn from audio data. In this section, we provide an extensive review of the literature on neural audio synthesis, organized based on the generative principles seen in Chapter 2 (NAMs, NFs, VAEs, and GANs). While there exist many direct applications of these methods to audio [van den Oord et al., 2016a, Aouameur et al., 2019, Engel et al., 2019], we will see that many works advocate for distributed solutions combining various of these techniques, e.g., VAEs and NFs [Kingma et al., 2016], AEs and NAMs [Engel et al., 2017]. Special attention is drawn towards the controllability of generative models of audio. Also, we highlight the type of audio sources modeled by each work, the conditional information (if applicable), and the form of the audio representation.

3.1.1 Controllable Neural Audio Synthesis

One important aspect that we stressed in the introduction of this thesis is that of *intuitive* control over the audio synthesis process. Two common strategies exist for achieving controllable generative models: supervised and unsupervised. Su-

pervised methods explicitly condition the model on auxiliary information during training. Conditional information is said to be *sparse* or *dense* depending on its amount of information or, in other words, how much of the variance it captures [Dieleman, 2020]. Also, each generative approach supports conditioning differently. Autoregressive models, which operate on a sample-by-sample basis, may be more inefficient when conditioned on global aspects of the data as information has to be repeated at each step in the sequence. GANs, on the contrary, can deal easily with global properties as they generate the whole piece of data in one pass.

An obvious example of a conditional generative model can be seen in text-to-speech synthesis, where the task of generating realistic speech is conditioned on some input text information [Shen et al., 2018]. Similarly, singing voice synthesizers are generally conditioned on pitch and lyric information [Nishimura et al., 2016, Blaauw and Bonada, 2017]. Neural audio synthesizers of instrument sounds condition on the instrument category and pitch [Engel et al., 2019, 2017, Roche et al., 2018]. Conditioning is not restricted to symbolic or *sparse* information (e.g., pitch, words, instrument). Other works use rather *dense* information and condition the models on preexisting audio content to drive the generative process. For example, in style transfer tasks, the goal is to take some piece of music in a specific style (e.g., rock, pop) and transform it into another style while preserving some fundamental content [Huang et al., 2019b, Mor et al., 2018, Cífka et al., 2021]. Other tasks conditioning on dense information are audio enhancement [Michelsanti and Tan, 2017, Biswas and Jia, 2020] or spectrogram inversion [Kumar et al., 2019]. While supervised methods rely on preexisting information to condition the model, unsupervised methods employ feature learning mechanisms to discover important factors of variations of the data autonomously. At test time, such learned features can potentially be used to guide the generation process. Some of the most successful applications of unsupervised feature learning for controllable generation can be found in face image synthesis tasks, where GANs can autonomously learn high-level attributes (e.g., pose, identity) separately from stochastic variation (e.g., hair) [Karras et al., 2018].

3.1.2 Neural Autoregressive Models

Neural Autoregressive Models (NAMs) are probably the most popular approach for building generative neural networks of audio. In the following sections we review some of the most important works applying NAMs to audio. First we focus on WaveNet and other popular works down the line which are based on causal convolutions. Next, we revise some other approaches that use different operations (e.g., recurrent, attention). Last, we mention some hybrid approaches that introduce autoregressive models as part of larger distributed systems. This enables, for example, to combine the robustness of autoregressive modeling with the latent space control that autoencoders offer.

WaveNet-like Architectures

The recently-developed WaveNet architecture [van den Oord et al., 2016a] is one of the most important architectures used for realistic speech synthesis and the most influential work in autoregressive models for audio generation in general. Inspired by other works in image [van den Oord et al., 2016c], it operates directly

Arch.	Name		Audio representation	Data	Conditioning
NAM	waveNet	[van den Oord et al., 2016a]	waveform	speech, piano	speaker ID, text
	Universal music Translation	[Mor et al., 2018]	waveform	classical music	-
	Hierarchical waveNet	[Dieleman et al., 2018]	waveform	piano music	-
	SampleRNN	[Mehri et al., 2017]	waveform	speech, piano music	-
	MelNet	[Vasquez and Lewis, 2019]	mag. spec.	speech, piano music	speaker ID text
	wavenetAE	[Engel et al., 2017]	waveform	tonal sounds	pitch
	sparse Transformer	[Child et al., 2019]	waveform	piano music	-
NFs	Parallel waveNet	[van den Oord et al., 2018a]	waveform	speech	text pitch
	ClariNet	[Ping et al., 2018]	waveform	speech	text
	FlowaveNet	[Kim et al., 2018]	waveform	speech	text Mel spec.
	waveGlow	[Prenger et al., 2018]	waveform	speech	text Mel spec.
	waveFlow	[Ping et al., 2020]	waveform	speech	text Mel spec.
	Blow	[Serrà et al., 2019]	waveform	speech	speaker ID
VAEs	Planet Drums	[Aouameur et al., 2019]	Mel-scaled mag. spec.	drums	instrument ID
	Jukebox	[Dhariwal et al., 2020]	waveform	music	artist & genre ID lyrics
	NOTONO	[Bazin et al., 2020]	mag. & IF	tonal instruments	pitch
	FlowSynth	[Esling et al., 2019]	mag.	synth. sounds	semantic tags
	Neural Granular Sound Synth.	[Bitton et al., 2020]	waveform	orchestral drums animals	pitch instrument ID
GANs	WaveGAN	[Donahue et al., 2019]	waveform	speech drums piano birds	-
	GANSynth	[Engel et al., 2019]	mag. & IF	tonal instruments	pitch ID
	MelGAN	[Kumar et al., 2019]	mag. spec.	speech music	Mel-scaled spec. text
	GAN-TTS	[Binkowski et al., 2020]	waveform	speech	pitch, text, speaker ID

Table 3.1 – Summary of the most important neural audio synthesis approaches

on the raw audio by modelling the probability of a waveform $\boldsymbol{x} = [x_1, ..., x_T$, factorised as a product of conditional probabilities (see Sec. 2.1.1). The architecture is built as a stack of Dilated Causal Convolutional layers. The filters in each convolutional layer are applied over an area larger than its length by skipping input values with a certain step or dilation. At each layer in the network, the dilation factor is doubled, allowing the network to grow its receptive field (i.e., the region of the sensory space that the network observes) exponentially with depth while preserving the number of computations. This dilation enables the model to operate on a coarser scale, capturing longer-term audio dependencies while preserving the information's resolution throughout the network. The output layer consists of a Softmax activation unit that models a categorical distribution over 256 possible amplitude values. Given an additional conditioning input, the authors can guide the generation of audio with certain characteristics and at different scales. For example, when applying WaveNet in speech generation, they can impose global characteristics on the speaker, such as its identity, or local characteristics, such as the phoneme to be synthesized, by conditioning the network on text information. During training, every causal convolutional layer can process its input in parallel, making these architectures faster than RNNs, which can only be updated sequentially. At generation time, however, the waveform has to be synthesized sequentially as x_t must be sampled first to obtain $x_{i>t}$. Due to this fact, real-time synthesis is challenging, in particular for music applications [van den Oord et al., 2018a].

Many efforts have been made to improve WaveNet's time and computation efficiency. Fast WaveNet [Paine et al., 2016] reduces the complexity of the algorithm from $\mathrm{O}(2^L)$ to $O(L)$ time (being L the network's number of layers) by storing previous convolution calculations in order to remove redundant operations. This approach requires the use of smaller networks, impacting the quality of the synthesized audio severely. In Sec. 3.1.4 we review a more recent approach, based on NFs, that introduces Probability Density Distillation [van den Oord et al., 2018a], a method for transferring knowledge from pre-trained WaveNet to a smaller NF with no significant degradation in quality. The resulting system is capable of generating high-quality speech and in real-time.

The use of WaveNet in speech, singing voice, and music synthesis has been predominant. As we will see later in this section, most of these works apply WaveNet as a waveform synthesizing building block, part of a larger distributed parametric system [Gibiansky et al., 2017, Ping et al., 2017, Shen et al., 2018, Roebel and Bous, 2021] or following an encoder-decoder architecture to provide means of control through the latent space [Engel et al., 2017].

Non-Convolutional Approaches

Neural autoregressive models may employ operations other than dilated convolutions such as recurrent connections or attention mechanisms. SampleRNN [Mehri et al., 2017] uses multiple RNNs stacked on top of each other, where each block in the stack operates at a different rate. Higher-level RNNs update less frequently, which means they can more easily capture long-range data dependencies and learn high-level features. Conversely, lower layers in the stack running at a faster rate capture local, fast-varying dependencies of the data (e.g., pitch, timbre, envelope).

We mentioned in Section 2.1.1 that autoregressive models, while naturally fitting the sequential scheme of the audio waveform, can be used on other audio representations. Using spectrograms, for example, one can easily increase the receptive field of a model (i.e., spectrograms condense the time information of a whole analysis frame in each frequency bin), simplifying the task of capturing global structure in comparison to other autoregressive approaches that work with audio in the time domain. MelNet [Vasquez and Lewis, 2019] is an RNN-based autoregressive model that operates on high-resolution time-frequency magnitude spectrograms, capturing long-range dependencies of the data. It combines a fine-grained autoregressive model and a multi-scale generation procedure to capture structure in a coarse-to-fine-grain manner jointly. The autoregressive model factorizes the distribution over both the time and frequency dimensions. Thanks to the time-condensed representation of magnitude spectrograms, coupled with the power of autoregressive models, MelNet achieves highly expressive and end-to-end unconditional audio generation for speech and music data.

Plain recurrent blocks tend to be slow during training and have difficulty learning dependencies between distant elements from the sequence. Introducing attention mechanisms allows an autoregressive model to access any part of the previously generated output at every step of generation [Vaswani et al., 2017]. Works on audio have used attention as part of an encoder-decoder architecture to pass relevant information from a latent space to a decoder generating, e.g., vocoder parameters [Sotelo et al., 2017], magnitude spectrograms [Wang et al., 2017] or Mel-scaled spectrogram representations [Shen et al., 2018]. Continuations of these works were able to generate prosodic speech by conditioning the attention layers on emotion labels [Lee et al., 2017b] as well as to synthesize speech for multiple speakers [Ping et al., 2017]. Other approaches such as the Transformer [Vaswani et al., 2017, Shaw et al., 2018], abandoned the traditional encoder-decoder configuration and adopted architectures based solely on attention mechanisms. These architectures have been successfully applied to symbolic music [Huang et al., 2019a] and, with the introduction of sparsity, to audio [Child et al., 2019], making it possible to generate minute-long music with rich structure at multiple scales.

Hybrid approaches

A downside of purely autoregressive models is that they do not explicitly produce latent representations of the data, limiting the extent to which they can be controlled at generation time. However, it is possible to combine an autoregressive sequence generation model with an encoder-decoder architecture [Engel et al., 2017, Mor et al., 2018, Chorowski et al., 2019]. In these works, an encoder reads a sequence of raw audio samples or feature vectors and extracts a sequence of latent representations. The decoder reconstructs the utterance by conditioning a WaveNet network on these latent representations and on additional features (e.g., pitch [Engel et al., 2017], speaker embedding [Chorowski et al., 2019]) to make the models invariant to specific feature-dependent information. The WaveNet Autoencoder [Engel et al., 2017] yields a pitch-independent timbre latent space where instruments can be morphed together through interpolation, and new types of sounds can be created that are realistic and expressive. Another work focused on musical audio style transfer uses WaveNet-like autoencoders to transform the

timbre of some input audio to target a specific style [Mor et al., 2018]. The architecture follows a single-encoder multi-decoder framework with a shared latent space, enforcing the network to learn a style-invariant latent representation, and each decoder is therefore responsible for conferring sound style-specific characteristics.

Most of the work in speech [Arik et al., 2017, Gibiansky et al., 2017, Ping et al., 2017, Shen et al., 2018] incorporated an optimized version of WaveNet as a vocoder model for reconstructing speech audio from linguistic features and f_0 [Arik et al., 2017], linear-scaled log-magnitude spectrograms [Gibiansky et al., 2017] or Mel-scaled spectrograms [Ping et al., 2017], being this last version the one that yields the best performance and a more compact representation of the conditioning audio. Tacotron 2 [Shen et al., 2018] follows the same approach and introduces a WaveNet vocoder as an improvement of the Griffin-Lim reconstruction module used in Tacotron [Wang et al., 2017]. Tacotron 2 yields some of the most human natural-sounding reconstructions. Similar techniques use WaveNet as part of a distributed system for singing voice synthesis [Blaauw and Bonada, 2017]. These works train WaveNet on features produced by a parametric vocoder that separates the influence of pitch and timbre. This separation allows to modify pitch to match any target melody conveniently, facilitates training on reduced dataset sizes, and significantly improves training and generation times.

3.1.3 Variational Autoencoders

Variational Autoencoders (VAE) [Kingma and Welling, 2014] are one of the most popular strategies for generative modeling. One of the main attractive properties behind VAEs is their capability to map data into a structured latent space that captures fundamental features. The possibility of controlling the generative process through such a latent space makes them an interesting asset in music modeling. Various successful works employ VAEs in symbolic representations [Roberts et al., 2017, Brunner et al., 2018a]. In audio, most initial works were tailored towards synthesis and transformation of speech [Blaauw and Bonada, 2016, Hsu et al., 2017]. Even though the latent space of VAEs tends to self-organize according to fundamental dependencies in the data, these can still be difficult to interpret. Some works in music data focused on regularizing the latent space of VAEs to accommodate perceptual distances collected from timbre studies [Esling et al., 2018a, Roche, 2020], and synthesize from such latent space audio that matches a semantically meaningful target descriptor [Esling et al., 2018b]. The original VAE formulation, where the inference network is used to parametrize a normal distribution (see Sec. 2.1.3), yields blurred generations [Huang et al., 2018]. Some works use Maximum Mean Discrepancy (MMD) distance instead of $D_{KL}(q\phi(z|x)||p_\theta(z))$ 2.4 to alleviate this problem. This approach has been successfully applied to synthesize percussive sounds, enabling to interpolate between a wide variety of instruments [Aouameur et al., 2019]. As we will see further on in this section, some other work implicitly imposes the prior distribution by using an adversarial loss in the latent space [Bitton et al., 2019]. Other works used two VAEs to implement a granular synthesizer [Bitton et al., 2020]: one that encodes grain series into compressed codes and a second VAE learning combinations of codes to define paths in the latent space of the first VAE. Some other interesting

applications of VAEs can encode pre-existing multi-track music material into an intuitive two-dimensional latent space and, from this, generate bass lines fitting the provided music content [Grachten et al., 2020]. A novel application of VAEs in combination with Normalizing Flows (NFs) can map the learned latent space of the VAE to parameters of a synthesizer [Esling et al., 2019]. This formulation enables a single model to perform high-fidelity audio synthesis and automatic parameter inference, macro-control learning, and audio-based preset exploration.

Another problem when generating high-quality audio with VAEs is the posterior collapse, by which robust decoding architectures such as WaveNet, may end up ignoring the latent codes. Some techniques discretize the latent space through parametrization of the posterior distribution using Vector Quantization (VQ), enabling the prior to be learned instead of imposed [van den Oord et al., 2017]. This architecture has shown remarkable results in tasks such as speech generation and speaker translation [Chorowski et al., 2019]. Following this line, Jukebox [Dhariwal et al., 2020] is a multi-scale VQ-VAE combined with Transformers that generates minute-long music with a singing voice in the raw audio domain. It allows conditioning on the artist, genre, and lyric information to steer the musical and vocal style of the generated content. This work sets a milestone in modeling long-term structure from large-scale music audio datasets and demonstrates the power of deep learning to model creative tasks. Other applications of VQ-VAEs include in-painting-based synthesis of tonal instruments [Bazin et al., 2020] or one-shot timbre style transfer [Cífka et al., 2021].

3.1.4 Normalizing Flows

Normalizing Flows (NFs) have recently become popular in the speech synthesis community. In Section 2.1.2 we studied how Normalizing Flows (NF) can be used to learn rich and flexible posteriors in DL-based variational inference by using neural networks implementing invertible transformations. One of the main shortcomings of NFs is their requirement to have the same input and latent dimensions, challenging the modeling of high-dimensional data as is the case in audio signals. A specific type of NF known as Inverse Autoregressive Flow (IAF) [Kingma et al., 2016] scales well to high-dimensional data by implementing the invertible transformation as an autoregressive neural network. The increased efficiency of IAFs has been used in audio to accelerate WaveNet-based speech synthesis to 20x faster than real-time [van den Oord et al., 2018a]. This work introduces a new method coined Probability Density Distillation, which allows training an IAF from a pre-trained teacher WaveNet with no significant difference in quality and enabling parallel sampling. However, this two-stage training pipeline is cumbersome and requires highly regularized training to avoid mode failure in the student. Subsequent works combine insights from Glow [Kingma and Dhariwal, 2018], and WaveNet [van den Oord et al., 2016a] to design flow-based models that provide fast, efficient, and high-quality speech generation without the need of two-stage training schemes nor additional auxiliary loss terms [Kim et al., 2018, Prenger et al., 2018, Ping et al., 2020]. Some works down this line employed similar flow-based architectures for non-parallel voice conversion [Serrà et al., 2019]. Applications of NFs to musical audio synthesis are scarce. As we have seen, some works have used flows in combination with VAEs to learn in-

vertible mappings between the VAE's latent space and a synthesizer's parameter space [Esling et al., 2019].

3.1.5 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] have been shown successful in various computer vision tasks such as image inpainting [Denton et al., 2016], domain translation and style transfer [Zhu et al., 2017, Choi et al., 2018, Liu and Tuzel] or high-fidelity image generation [Gulrajani et al., 2017, Chen et al., 2016b, Karras et al., 2018]. Taking inspiration from these works, applications of GANs to audio synthesis have mainly focused on speech tasks [Saito et al., 2018, Kaneko and Kameoka, 2017, Huang et al., 2019b, Binkowski et al., 2020, Kong et al., 2020a, Kumar et al., 2019, Yamamoto et al., 2020]. Initial works demonstrated that adversarial training could convert one speaker into another while preserving the linguistic content [Kaneko and Kameoka, 2017] or synthesizing realistic speech from text [Saito et al., 2018]. Novel cross-modal applications use conditional GANs to generate sound from image information and vice-versa [Chen et al., 2017, Iashin and Rahtu, 2021]. GANs have been used for symbolic music generation using a Recurrent Neural Network generator [Lee et al., 2017a] or in music genre transfer using cycle-consistent architectures [Brunner et al., 2018b]. The first application to musical audio synthesis was WaveGAN [Donahue et al., 2019]. Although it did not match autoregressive baselines such as WaveNet [van den Oord et al., 2016a] in terms of audio quality, it could generate piano and drum sounds in a short amount of time and in an entirely unconditional way. Recent work along the line of WaveGAN has achieved some promising results in footstep sound synthesis [Comunità et al., 2021]. General improvements in the stabilization and training of GANs [Karras et al., 2017, Gulrajani et al., 2017, Salimans et al., 2016] enabled GANSynth [Engel et al., 2019] to outperform WaveNet baselines on the task of audio synthesis of musical notes using sparse pitch conditioning labels. GANSynth follows the principle of Progressive Growing of GANs (PGAN) [Karras et al., 2017], where a generative network, composed of convolutional and up-sampling blocks, is built on the fly while training (see Section 2.1.4). Follow-up works building on GANSynth applied similar architectures to conditional drum sound synthesis using different metadata [Nistal et al., 2020, Drysdale et al., 2020]. DrumGAN [Nistal et al., 2020] synthesizes a variety of drum sounds based on high-level input features describing timbre (e.g., boominess, roughness, sharpness). Given their mode-seeking behaviour (see Sec. 2.1.4), GANs have been popular in densely conditioned tasks such as Mel-spectrogram inversion for speech [Kumar et al., 2019] or singing voice synthesis [Chen et al., 2021], audio domain adaptation [Hosseini-Asl et al., 2018, Michelsanti and Tan, 2017] or audio enhancement [Biswas and Jia, 2020]. Some works introduce adversarial objectives into the VAE training scheme to synthesize Mel-spectrograms of orchestral instruments given a note class and some latent vector capturing style parameters [Bitton et al., 2019]. While GANs often require large amounts of data to learn some specific task, recently, patch-based GANs were shown capable of learning from one single image example, capturing its internal distribution, and enabling to generate variations from it [Shaham et al., 2019]. This approach has been successfully translated to the audio domain for sound effect, speech, or mu-

sic generation [Barahona-Ríos and Collins, 2021, Greshler et al., 2021]. A recent work proposes a combination of the autoregressive and adversarial schemes by sampling large chunks of the waveform during each autoregressive forward pass, bringing together the fast generation capabilities of GANs with the benefits of the autoregressive inductive bias [Morrison et al., 2021].

3.2 Audio Synthesis Prior the Deep Learning Era

The interest of humans in crafting machines that can generate sounds and music dates back to at least the 19th century when Ada Lovelace anticipated the era of computer music [Fuegi and Francis, 2015] and the first generation of Electronic Musical Instruments (EMI) appeared [Crab, 2016]. A wide variety of sound models have been proposed since then. These can be categorized into abstract, spectral, physical, or based on processed recordings, depending on how they model sound [Smith, 1991]. Essentially, most of these methods start from some fundamental waveforms, which are combined and transformed in various ways to produce different sounds. Techniques differ in the shape of such fundamental waveforms and the way these are processed to form richer sounds. Also, each method exhibits characteristics that could make it preferable over others depending on the specific musical purposes. In this section, we briefly overview some of these modeling strategies and research aimed at devising semantically intuitive interfaces for their control. For an in-depth review of these works we refer the reader to Roads et al. [1997], Smith [a], Miranda [2002].

3.2.1 Abstract Models

Abstract methods such as Frequency Modulation (FM) [Chowning, 1973], implemented in the famous Yamaha DX7, use algorithmic procedures or conceptual mathematical formulations to model sound [Roads et al., 1997]. As a result, the parameters offered by these types of synthesis techniques do not have a direct physical or perceptual meaning and fail to precisely model existing natural sounds [Miranda, 2002]. However, they have been highly appreciated due to their low computational and memory requirements as well as their rich timbre capabilities, with just a few parameters, that allow synthesizing sounds that would otherwise be impossible to generate through physical means [Kleimola, 2013]. Today, while abstract methods have lost some of their original prominence and are considered obsolete from a research perspective [Serra, 2007], we can find them as a building block of many commercial applications based on other modeling principles (e.g., subtractive[1], wavetable[2]).

3.2.2 Spectral Models

Spectral models synthesize sound by characterizing its spectral content following Fourier theory. The earliest and simplest form of spectral modeling is additive synthesis, which forms sound as a sum of discrete sinusoidal components modulated by time-varying amplitude and frequency envelopes [Smith, b]. While

[1] https://www.waves.com/plugins/flow-motion-fm-synth

[2] https://www.reasonstudios.com/shop/rack-extension/wtfm-wavetable-fm-synthesizer/

their parameters are closer to human perception than other strategies [Serra et al., 2007], these require many components to properly model rich sounds, which makes them computationally expensive [Miranda, 2002]. Alternative techniques introduce a time-varying filtered noise to model stochastic components in the sound and also allow for analysis of existing audio signals [Serra and Smith, 1990].

Another spectral modeling strategy, and one of the most popular techniques implemented in commercial synthesizers (e.g., Minimoog Model D, Roland TR-808), is subtractive synthesis. Loosely categorized as a source-filter modeling technique, subtractive synthesis can be seen as the inverse process to additive synthesis, where rich broadband signals such as square, saw-tooth, pulses, or noise are filtered to remove undesired frequency components [Roads et al., 1997]. While their controls are much less numerous than for additive synthesis and are computationally cheap, they are less flexible and fail to capture faithfully many acoustic instruments [Miranda, 2002].

3.2.3 Physical Models

Physical models can emulate acoustic instruments by mathematically reproducing in a computer their mechanical behavior and solving their associated differential equations to produce sound [Miranda, 2002]. These techniques can synthesize realistic sounding instruments and have the benefit of providing intuitive controls responding to physical properties of the acoustic source (e.g., the stiffness, tension) or the excitation signal (e.g., strength, friction) [Roads et al., 1997]. However, they are computationally expensive and, analogously to their acoustic counterparts, they can only generate a limited variety of timbre. Today, efforts are focused on providing more flexible approaches based on a combination of elementary model blocks, physical modeling based on data analysis, or its use in combination with spectral methods [Serra et al., 2007, Smith, a].

3.2.4 Processed Recording

Early examples of sound creation from processed recordings were based on transforming and looping short snippets of sounds recorded in tapes to create novel sound compositions. This technique was pioneered in music by many technologists, and music futurists such as Edgard Varèse and Pierre Schaeffer in Musique Concrète [Miranda, 2002]. With the increasing storage and computing capabilities of computers, these sample-based techniques evolved into more sophisticated ones such as granular synthesis, producing sounds based on short, time-varying portions of sampled sounds, so-called grains [Roads et al., 1997]. From a more general perspective, wavetable techniques allowed to repeatedly playback arbitrary wave shapes stored in a lookup table [Smith, 1991]. More recently, and with the ever-growing availability of audio datasets, concatenative synthesis introduced the notion of analysis to assemble the desired sound according to some predefined sound descriptors or by analysis of an existing sound [Schwarz, 2007]. A prominent example of these is Vocaloid,[3] a real-time singing voice synthesizer

[3]https://www.vocaloid.com/en/

that can be controlled with some input text and timbre features. Today, many advanced synthesizers are based on samplers such as Native Instruments' Kontakt[4] and Steinberg's HALion,[5] achieving some of the most detailed and accurate emulations of acoustic instruments, where no other synthesis technique is capable of the same levels of realism. Nevertheless, these techniques generally do not allow for a rich manipulation or require large amounts of data to generate expressive sounds [Smith, 2004].

3.2.5 Knowledge-driven Controllable Audio Synthesis

Audio synthesizers have given birth to a new paradigm for producing sounds where no a-priory limitation exists on the kind of sounds that can be produced or how we can interact with them. In contrast, acoustic instruments are limited by their specific physical characteristics constraining the sounds they can produce and the means of interaction. However, acoustic instruments offer a very intuitive interface where the interaction mechanics are directly related to high-level properties of the sound, i.e., the pressure of the bow in a violin is directly related to the intensity of the produced sound. A question in synthesizers, therefore, is how to devise means of control that are suitable for the synthesis algorithm in such a way for actions and expectations to be consistent [Roads et al., 1997].

As computing capabilities became more powerful during the 90s and early 2000s, new research directions appeared related to intuitive control of synthesizers where perceptual and cognitive aspects are taken into account in order to steer the sound synthesis process [Ystad et al., 2019]. In other words, these works studied how to map a specific control signal (e.g., gestures [Camurri et al., 2000], perceptual attributes [Aramaki et al., 2011a]) into the synthesizer's parameters. To this end, perceptual and cognitive studies were carried out to understand the principles of how a sound is perceived and its relationship to specific acoustic features present in the signal, so-called invariants, that can be identified from analysis. Identifying such signal invariants makes it possible to propose perceptual control over the sound synthesis processes that enable direct, evocative control of such perceptual properties. Along this line, some works have attempted to drive the physical synthesis of environmental sounds such as rain, waves, wind, and fire based on semantic labels, gestures, or drawings [Aramaki et al., 2011b]. Other works can derive intuitive perceptual controls on synthesizers of impact sounds by careful study of their acoustic features in consonance with human-annotated categories of materials [Aramaki et al., 2006, 2011a]. By considering such relationships, a synthesizer can then be designed to control the acoustic features found to correlate with the annotated categories. This process is shown to offer manipulation of intuitive parameters responding to the material label (i.e., Wood, Metal, or Glass). Physical and spectral models have been combined to synthesize flute sounds based on simulation of the wave propagation in the medium and by using deterministic plus stochastic decomposition to control independent components [Ystad, 1998]. The authors can obtain a gesture-driven interface to control the proposed model by equipping a flute with sensors. Acoustic invariants related to the evocation of continuous interacting solids such as rubbing, scratching, and

[4]https://www.native-instruments.com/en/products/komplete/samplers/kontakt-6/
[5]https://www.steinberg.net/vst-instruments/halion/

rolling were also identified and used for sound synthesis purposes [Conan et al., 2014]. A synthesizer was developed where the actions (e.g., from rolling to slipping) and the properties of the acoustic source (shape, size, and material) could be controlled continuously over time [Pruvost et al., 2015]. Extensions of these works propose a cross-synthesis approach to modify the intrinsic properties of a given sound texture to evoke a particular interaction (rolling or rubbing) in a way to create sonic metaphors [Conan, 2014]. A broad perspective of this research is further described by Ystad et al. [2019].

3.3 Discussion

This chapter reviewed works for synthesizing audio using deep learning and, less extensively, techniques based on traditional signal processing methods, which are driven by expert knowledge. Here we extend the discussion in Chapter 2, on the benefits of each deep generative modeling technique, by taking into consideration the specific advancements in neural audio synthesis and by contrasting them with those methods employing expert knowledge. Following, we highlight some of the most relevant aspects of the works reviewed in this chapter.

- **Expert vs. DL-driven audio synthesis**. Many expert-driven synthesis methods have been proposed, each offering specific manipulation and sound capabilities [Smith, 1991]. While these can generate a wide variety of timbres, many complex natural sounds are still not faithfully modeled by these techniques, imposing the need for heavy physical models or data-hungry corpus-based synthesis techniques [Roads et al., 1997]. Neural audio synthesizers, on the contrary, and Generative Adversarial Networks (GANs) specifically, have been shown to model a great variety of sound sources ranging from sound effects to music and speech and using general formulations [Barahona-Ríos and Collins, 2021, Engel et al., 2019, Morrison et al., 2021]. Also, as opposed to expert systems [Ystad, 1998], GANs can be controlled based on abstract descriptors without requiring a principled understanding of the perceptual or timbral properties of the sound and their correspondence to feature invariants [Engel et al., 2019]. However, a consequence of this is that DL models tend to behave as black boxes whose parameters are difficult to interpret, whereas expert systems are built upon well-established rules and understanding.

- **Conditioning & Control.** Many works have successfully implemented conditional models of audio to allow some degree of control over the generative process [Engel et al., 2019, 2017, Aouameur et al., 2019]. Models can be conditioned on sparse, categorical data, for example, to choose a speaker identity in speech synthesis [van den Oord et al., 2016a, Vasquez and Lewis, 2019] or an instrument in sound synthesis [Aouameur et al., 2019]. Other works have conditioned the model on rather dense information, such as spectral information, to constrain some target domain features in style transfer [Mor et al., 2018] or in Mel-spectrogram inversion [Kumar et al., 2019]. Autoregressive models of audio, such as WaveNet [van den Oord et al., 2016a], do not directly offer a latent space that can be manipulated, and their hidden layers are not regularized to follow any prior

distribution, which makes them difficult to control without external conditioning. Therefore, many works have made use of WaveNet-like blocks in an encoder-decoder fashion to allow encoding and manipulation of sounds in a latent space [Engel et al., 2017]. Variational Autoencoders (VAEs) and Normalizing Flows (NFs) naturally provide encoders that capture fundamental aspects of the data and that allow encoder sounds, although these can often be hard to effectively condition due to their specific inductive bias [Esling et al., 2019]. GANs have no encoder at all, yet they can be easily conditioned by just concatenating arbitrary external information to their latent noise vectors [Engel et al., 2019].

- **Inference and synthesis efficiency.** One of the main shortcomings of Neural Autoregressive Models (NAMs) is their slow generation time due to their inherent sequential generation scheme [van den Oord et al., 2016a]. While methods have been proposed to speed up audio generation in autoregressive models, these are generally based on cumbersome two-stage training schemes or careful architecture designs using flows [van den Oord et al., 2018a, Kim et al., 2018]. GANs, however, can generate full audio samples in a single forward pass much faster than NAMs. While NFs and VAEs can also be fast, the former tend to require rather deep and inefficient networks due to the invertibility constraint, or, in the latter, they tend to produce blurred, lower-quality samples than other generative strategies.

- **Sample quality and diversity.** NAMs and GANs have the advantage of generating high-quality audio fidelity with relatively simple networks. While VAEs have traditionally produced worse quality than other generative models, recent works introduce vector-quantization and autoregressive blocks, achieving impressive audio quality and diversity at the expense of generation time [Dhariwal et al., 2020]. Although GANs lack the same degree of diversity as other models due to their mode-seeking nature, they can generate audio much faster and obtain extremely good sample quality.

Chapter 4

Methodology

In this chapter we describe the global methodology followed throughout our experiments, unless specified otherwise in each chapter. In Section 4.1 we describe the general GAN architecture and its training procedure. Section 4.2 presents the main datasets: the NSynth dataset, CSL-Drums, and MP3-to-WAV. Finally, in Section 4.3 we mention some of the evaluation metrics used to assess the performance of our models.

4.1 Architecture

Our reference architecture is a Progressive Growing GAN (PGAN), described in Section 2.1.4 and which is inspired by previous work on image generation [Karras et al., 2017]. As we have seen in Chapter 3, this architecture was firstly employed to generate audio in GANSynth [Engel et al., 2019], comfortably surpassing WaveNet baselines in the tasks of tonal sound synthesis, according to human evaluation tests and quantitative metrics.

The architecture is depicted in Figure 4.1. The generator G samples a random vector $\boldsymbol{z} \in R^{n_z}$ from a standard normal distribution $\boldsymbol{z} \sim \mathcal{N}_{n_z}(\mu = 0, \sigma^2 = \mathbf{I})$ and feeds it together with some conditional information $\boldsymbol{c} \in R^{n_c}$ through an *input* block and a stack of N *scale* blocks.[1] The *input* block turns the $1D$ input vector *cat(**z**, **c**)*, with size $n_z + n_c$, into a 4D convolutional input by first zero-padding in the time and frequency-dimension (i.e., placing the input vector in the middle of the convolutional input tensor with $n_z + n_c$ convolutional maps) and then passing it through two convolutional layers with Leaky ReLU activation. The resulting tensor has shape (b, n_{ch0}, w_0, t_0), where b indicates the batch dimension, n_{ch0} are the number of convolutional channels in the *input* block, and (w_0, t_0) are the number of bins ($w_0 = 1$ if the audio representation is the raw audio waveform) and the number of frames/samples, respectively, for the first scale.[2] Following the *input* block, each *scale* block is composed of a nearest-neighbour up-sampling step at the input followed by two convolutional layers with filters of size $(3, 3)$ and Leaky ReLU as activation function. As depicted in Fig. 4.1,

[1] Generally we employ $N = 6$, although in our initial work, presented in Chapter 5, we employ $N = 5$ for simplification.

[2] May the reader be reminded from Chapter 2 that, in PGANs, the architecture is built on the fly while training and therefore (w_0, t_0) refers to the shape of the corresponding audio representation (e.g., spectrograms, waveform) generated by G on the earliest stages of training.

the discriminator D is composed of convolutional and down-sampling blocks, mirroring the configuration of the generator. However, D, has an *output* block which is composed of one convolutional layer followed by two fully-connected layers, all with Leaky ReLU activation. As explained early on in Section 2.1.4, D estimates the Wasserstein distance between the real and generated distributions [Gulrajani et al., 2017] using the gradient penalty method, with $\lambda = 10.0$ in (2.7), to enforce the Lipschitz constraint. As depicted in Fig. 4.2, in order to encourage G to use the conditional information $\boldsymbol{c}$, D predicts $\hat{\boldsymbol{c}}$ and an auxiliary loss term is added to the Wasserstein objective following previous approaches in conditional GANs [Odena et al., 2017]. The specific loss will depend on the task under consideration and the nature of the conditional data, e.g., continuous features, multi/single-class attribute labels, probabilities.

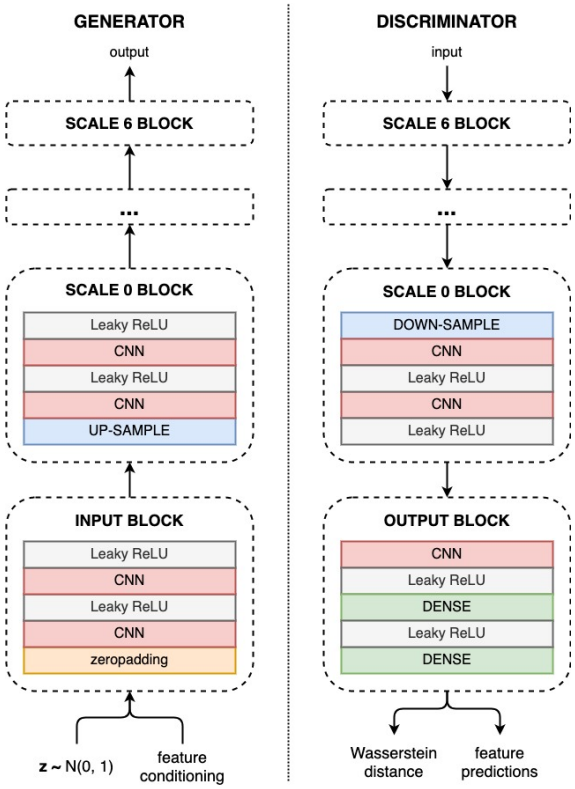


Figure 4.1 – On the left: the architecture of the generator G; on the right: the architecture of D mirroring G's configuration.

Following the process explained in Section 2.1.4, pixel normalization is applied after each convolutional layer, i.e., normalizing the norm over the output maps at each spatial location or, in the case of audio, time-frequency position. We initialize weights to zero and apply He's constant [He et al., 2015] for normalizing each layer at run-time in order to ensure an equalized learning rate (see Section 2.1.4). Such normalization ensures a balanced training between G and D by keeping the weights in the network at a similar scale. Also, we use a *mini-batch standard deviation* before the last layer of D in the *output* block [Salimans et al., 2016] (see Section 2.1.4) in order to encourage G to generate more variety and reduce mode collapse.

Training follows the procedure of Progressive Growing GANs [Karras et al.,

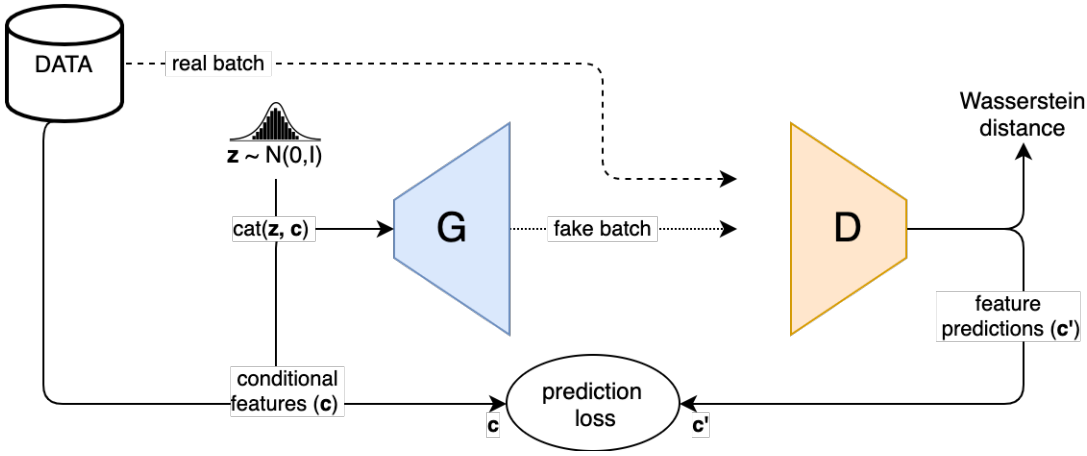


Figure 4.2 – Conditional GAN training scheme.

2017] explained in Section 2.1.4. We have seen that in a PGAN, the architecture is built dynamically during training. The training process is divided into stages wherein each stage a new corresponding *scale* block is introduced to both G and D. While training, a blending parameter α progressively fades in the gradient derived from the new blocks, minimizing possible perturbation effects. We train each *scale* block for 200k training iterations except for the first and last blocks which are trained for 128k and 300k iterations respectively. As for the batch size, we employ a different one for each *scale* block. For early stages we use higher batch sizes (e.g., 30 and 20) and for the last stages we generally use 12 samples. We employ Adam as the optimization method and a learning rate of 0.001 for both networks.

4.2 Datasets

Three main datasets are used in our experiments. First, the NSynth dataset [Engel et al., 2017] is used in Chapters 5, 7 and 8 in the task of audio synthesis of tonal sounds. CSL-Drums is used in Chapter 6 for synthesis of percussion sounds. Finally, in Chapter 9 we employ the MP3-to-WAV dataset for the task of restoring heavily compressed musical audio.

- **NSynth** [Engel et al., 2017]. This dataset[3] contains over 300k single-note audios played by more than 1k different instruments from 10 different families (e.g. bass, flute, guitar). The samples are aligned, meaning that each sample's onset occurs at time 0. The dataset contains various labels (e.g., pitch, velocity, instrument type), but, unless stated otherwise, we only make use (i.e., condition the model on) pitch information. As we will see later in this chapter, we consider the instrument class labels in order to train an Inception network for evaluation purposes. Each sample is four seconds long, with a 16kHz sample rate. For computational simplicity, we trim down the audio samples from 4 to 1 seconds and only consider samples with a MIDI pitch range from 44 to 70 (103.83 - 466.16 Hz). For the initial experiments described in Chapter 5 we only consider acoustic instruments

[3]https://magenta.tensorflow.org/datasets/nsynth

from the brass, flutes, guitars, keyboards, and mallets families. For the evaluation, we perform an 90/10% split of the data.

- **CSL-Drums**. In Chapter 6 we describe experiments on synthesis of percussive sounds. To this end we make use of an internal, non-publicly available dataset of approximately 300k one-shot audio samples aligned and distributed across a balanced set of kick, snare, and cymbal sounds. The samples originally have a sample rate of 44.1kHz and variable duration. For simplification, each sample is correspondingly shortened or zero-padded to a duration of one second. Unless stated otherwise, we carry out experiments using audio with a 16 kHz sample-rate. We perform a 90% / 10% split of the dataset for validation purposes.

- **MP3-to-WAV**. This dataset is composed of audio data pairs, where one part is an MP3 audio signal and the other is an uncompressed, high-quality (44.1 kHz) version. We use a dataset of approximately 64 hours of Nr 1 hits of the US charts between 1950 and 2020. The high-quality data is then compressed to 16kbit/s, 32kbit/s and 64kbit/s mono MP3 using the LAME MP3 codec, version 3.100.[4] The total number of songs is first divided into train, eval, and test sub-sets with a ratio of 80%, 10%, 10%, respectively. We then split each of the songs into 4-second-long segments with 50% overlap for training and validation.

4.3 Evaluation

Evaluating generative models is not straight-forward. Particularly in the case of GANs which, as we saw in Chapter 2, are an *implicit* density estimation method and therefore they do not provide direct means to evaluate the likelihood of each element in the training set. Additionally challenging is the task of synthesizing audio per se, where the goal of generating realistic audio is hard to formalize from a perceptual point of view. A common practice is to compare models by listening to samples or to measure their performance in some surrogate classification task [Engel et al., 2019]. Similarly, we evaluate our models against a diverse set of metrics, each capturing a distinct aspect of the model's performance.

4.3.1 Inception Score

The *Inception Score (IS)* [Salimans et al., 2016] is defined as the mean KL divergence between the conditional class probabilities $p(y|x)$, and the marginal distribution $p(y)$ using the predictions of a pre-trained Inception classifier (see Fig. 4.3), as

$$\exp\big(\underset{x}{E}[KL(p(y|x)||p(y))]\big). \tag{4.1}$$

IS penalizes models whose examples cannot be classified into a single class with high confidence, as well as models whose examples belong to only a few of all the possible classes. Spectrograms that contain meaningful objects should have a conditional label distribution $p(y|x)$ with low entropy. At the same time, we

[4] `https://lame.sourceforge.io/` (accessed on 31 May 2021)

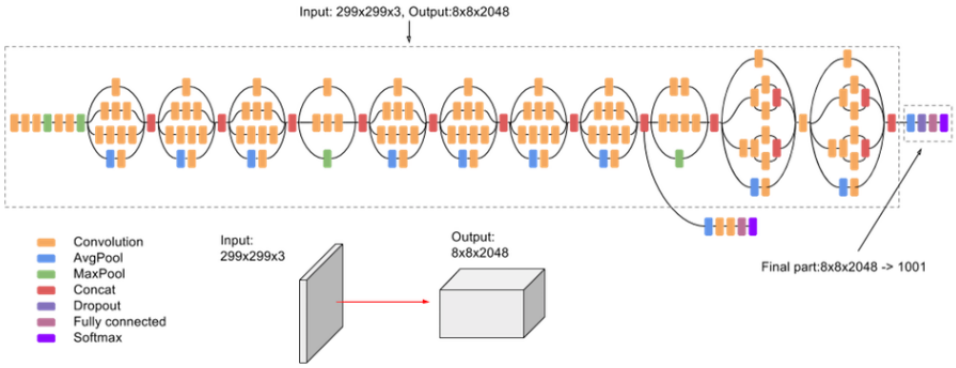


Figure 4.3 – Architecture of the Inception Model for image classification as described by Szegedy et al. [2016]. We adapt this architecture to audio and train our own inception model on instrument, and/or pitch classification.

expect the model to generate varied sounds, so the marginal $\int p(y|x = G(z))\, dz$ should have high entropy. This metric is found to be useful for the evaluation of image models, correlating well with human judgment, although it is not sensible to over-fitting [Barratt and Sharma, 2018].

Following previous work [Engel et al., 2019], we adapt this metric to audio and train our own Inception network[5] to classify the attributes accompanying the corresponding dataset, e.g, the instrument and pitch classes in the case of those experiments involving the NSynth dataset, or, the instrument class and perceptual features (see Chapter 6) in the case of the CSL-Drums dataset. The Inception model is trained on 1-second long Mel-scaled magnitude STFT spectrograms with 128 bins. We use a train/validation split of 90% / 10%.

4.3.2 Kernel Inception Distance

The *Kernel Inception Distance (KID)* [Binkowski et al., 2018] measures the dissimilarity between samples drawn independently from a real p_r and generated p_g distributions. It is defined as the squared Maximum Mean Discrepancy (MMD) between representations of the last layer of the same Inception model mentioned in the previous section. A lower MMD means that the generated p_g and real p_r distributions are close to each other. We employ the unbiased estimator of the squared MMD [Gretton et al., 2012] between m samples $x \sim p_r$ and n samples $y \sim p_g$, for some fixed characteristic kernel function k, defined as

$$\begin{aligned} MMD^2(X,Y) = & \frac{1}{m(m-1)} \sum_{i \neq j}^{m} k(x_i, x_j) \\ + & \frac{1}{n(n-1)} \sum_{i \neq j}^{n} k(y_i, y_j) \\ - & \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i, y_j). \end{aligned} \tag{4.2}$$

[5] www.github.com/pytorch/vision/blob/master/torchvision/models/inception.py

Here, we use an inverse multi-quadratic kernel (IMQ) $k(x, y) = 1/(1 + ||x - y||^2/2\gamma^2)$ with $\gamma^2 = 8$ [Rustamov, 2019], which has a heavy tail and, hence, it is sensitive to outliers.

4.3.3 Fréchet Audio Distance

The *Fréchet Audio Distance (FAD)* [Kilgour et al., 2018] compares the statistics of real and generated data computed from an embedding layer of a pre-trained VGG-like model.[6] Viewing the embedding layer as a continuous multivariate Gaussian, the mean and co-variance are estimated for real and fake data, and the FAD between these is calculated as

$$FAD = ||\mu_r - \mu_g||^2 + tr(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}), \tag{4.3}$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and co-variances of the embedding of real and generated data respectively. Lower FAD means smaller distances between synthetic and real data distributions. FAD performs well in terms of robustness against noise, computational efficiency, consistency with human judgments and sensitivity to intra-class mode dropping.

[6] `https://github.com/google-research/google-research/tree/master/frechet_audio_distance`

Chapter 5

Comparing Representations for Audio Synthesis Using GANs

In recent years, deep learning for audio has shifted from using hand-crafted features requiring prior knowledge, to features learned from raw audio data or mid-level representations such as the Short-Time Fourier Transform (STFT) [Dieleman and Schrauwen, 2014]. Indeed, this has allowed us to build models requiring less prior knowledge, yet at the expense of data, computational power, and training time [Zhu et al., 2016]. For example, deep autoregressive techniques working directly on raw audio [van den Oord et al., 2016a], as well as on Mel-scaled spectrograms [Vasquez and Lewis, 2019], currently yield state-of-the-art results in terms of quality. However, these models can take up to several weeks to train in a conventional GPU, and also, their generation procedure is too slow for typical production environments. On the other hand, GANs [Goodfellow et al., 2014], have achieved comparable audio synthesis quality and faster generation time [Engel et al., 2019], although they still require long training times and large-scale datasets when modeling low or mid-level feature representations [Marafioti et al., 2019, Donahue et al., 2019].

It is still subject to debate what the best audio representations are in machine learning in general, and the best choice may also depend on the respective application and the models employed. In audio synthesis with GANs, different representations may result in different training and generation times, and may also influence the quality of the resulting output. For example, operating on representations that compress the information with respect to perceptual principles, or are structured to better support a specific model architecture, may yield faster training and generation times, but may result in worse audio quality. In this chapter we compare different audio signal representations, including the raw audio waveform and a variety of time-frequency representations, for the task of adversarial audio synthesis with GANs. To this end, we evaluate our models using the evaluation metrics described in Sec. 4.3 and report on the respective training, generation, and inversion times. Furthermore, we investigate whether global attribute conditioning may improve the quality and coherence of the generated audio. For that, we perform extensive experimental evaluation when conditioning our models on the pitch information, as well as in a fully unconditional setting. We use the Progressive Growing Wasserstein GAN described in Sec. 4.1.

The content of this chapter is extracted from our paper:

Nistal, J., **Lattner, S.**, and **Richard, G.** "Comparing Representations for Audio Synthesis Using Generative Adversarial Networks." In Proceedings of the 28th European Signal Processing Conference (*EUSIPCO*), 2020.

The rest of the chapter is organized as follows: In Section 5.1, we describe the experiment setup: the dataset, architecture design, training procedure, and the evaluation metrics. Results are discussed in Section 5.2, and we conclude in Section 5.3.

5.1 Experiment Setup

Architecture. The architecture follows the design described in Sec. 4.1. The generator G implements a latent space with dimension $n_z = 128$ which is concatenated with a one-hot encoding of the conditional pitch class c_p with $n_c = 27$, resulting in a 1D input vector $cat(z, c_p)$ with size $n_z + n_c = 155$. We employ $N = 5$ scale block wherein each block, the CNNs have {128, 64, 64, 64, 32} feature maps, from low to high resolution, respectively.

Dataset. For this work, we employ the NSynth dataset [Engel et al., 2017] described early on in Sec. 4.2. As mentioned there, the subset of NSynth that we use only contains acoustic instruments from the brass, flutes, guitars, keyboards, and mallets families. This yields a subset of approximately 22k sounds with balanced instrument class distribution.

Audio representation. In this work we compare the audio representations described in Section 2.4: the raw audio waveform (referred to as *waveform*), the complex-valued STFT (*complex*), the magnitude and instantaneous frequency of the STFT (*mag-if*), the CQT transform (*cqt*) and it's invertible implementation using the Non-Stationary Gabor Transform[1] [Velasco et al., 2011] (*cq-nsgt*), the Mel-scaled magnitude of the STFT (*mel*) and, finally, the MFCCs (*mfcc*). All time-frequency representations, except *cqt* and *cq-nsgt*, are computed using an FFT size of 1024 and 75% overlapping. In the case of *mel* and *mfcc*, we employ a filter-bank of 128 Mel bins. For *mfcc*, we do not compress the Mel frequency information so as to preserve pitch information. *cqt* is computed using 12 bins per octave with a total of 84 bins. *cq-nsgt* is computed using 193 bins and assuming a complex signal. This leads to a non-symmetric spectrogram in which correlated frequency information is mirrored around the DC component. In order to make the information more local, we fold the magnitude and phase components and discard the DC, yielding a representation with 4 channels (corresponding to the upper and lower spectrogram replicas of the magnitude and phase components). The resulting tensor sizes for each representation are summarized in Table 5.1.

Evaluation. We evaluate our models in terms of informal listening tests and quantitative metrics computed on the generated content. For each audio representation, models are compared in conditional and unconditional settings and are also assessed in terms of complexity (e.g., generation time). As quantitative metrics, we employ those described in Section 4.3: the Inception Score (IS), the Kernel Inception Distance (KID), and the Fréchet Audio Distance (FAD). For

[1] `https://github.com/grrrr/nsgt`

Audio rep.	channels	freq. bins	time frames/samples
waveform	1	-	16000
complex	2	512	64
mag-if	2	512	64
cq-nsgt	4	97	948
cqt	2	84	256
mel	1	128	64
mfcc	1	128	64

Table 5.1 – Audio representation configuration

Models	PIS	IIS	PKID	IKID	FAD
real data	12.5	4.0	0.000	0.000	0.01
waveform	3.7	1.8	0.083	0.291	6.46
complex	**9.5**	2.8	**0.007**	0.124	3.17
mag-if	7.3	2.7	0.015	0.149	2.71
cq-nsgt	8.1	**3.4**	0.012	**0.041**	**2.11**
cqt	7.8	2.6	0.013	0.112	2.55
mel	2.3	1.1	0.147	0.300	5.20
mfcc	8.9	3.0	0.008	0.080	2.92

Table 5.2 – Unconditional models (i.e., trained without pitch conditioning). Higher is better for PIS and IIS, lower is better for PKID, IKID and FAD.

the inception-based metrics, we train an Inception model on pitch and instrument classification and report the IS on each task. We refer to these as Pitch IS (PIS) and Instrument IS (IIS). In the case of FAD, a publicly available pre-trained model is used.[2]

5.2 Results

In the following sections, we present the results of the quantitative and complexity studies for each model. We also provide some qualitative analysis by means of informal listening tests.

5.2.1 Evaluation Metrics

The quantitative results for samples generated by the unconditional and conditional models are shown in Tables 5.2 and 5.3, respectively. We observe a trend that the figures get worse from *complex* and *mag-if* to *mel* and *waveform*. In

[2] `https://github.com/google-research/google-research/tree/master/frechet_audio_distance`

Models	PIS	IIS	PKID	IKID	FAD
real data	12.5	4.0	0.000	0.000	0.01
waveform	3.4	2.1	0.222	0.108	1.87
complex	12.0	2.7	0.005	0.159	**0.11**
mag-if	**12.6**	**3.9**	**0.002**	**0.020**	0.12
cq-nsgt	7.6	3.3	0.014	0.049	0.12
cqt	12.3	**3.9**	0.008	0.107	2.03
mel	12.3	3.8	0.165	0.371	4.79
mfcc	9.7	3.7	0.006	0.074	2.62

Table 5.3 – Conditional models. Higher is better for PIS and IIS, lower is better for PKID, IKID and FAD.

Models	PIS	IIS	PKID	IKID	FAD
cqt	10.5	3.1	0.001	0.001	0.66
mel	12.5	3.7	0.001	0.001	0.31
mfcc	12.8	3.4	0.001	0.001	1.29

Table 5.4 – Metrics of post-processed real data for lossy transformations. Higher is better for PIS and IIS, lower is better for PKID, IKID and FAD.

some metrics, the highest quality models (*complex*, *mag-if*, and *cqt*) obtain results close to the real data. Furthermore, the results are generally better in the conditional setting. This is probably because the pitch-conditioning signal guides the generator in covering the variance over pitches, making it easier for the generator / discriminator pair to learn the remaining variances. Informal listening tests suggest that PKID, IKID and FAD are better aligned with perceived sound quality than PIS and IIS. In PKID, IKID and FAD (in both, the conditional and unconditional setting), the models of all representations seem to perform similarly, except *mel* and *waveform*, which both yield considerably worse results.

PIS and IIS seem to correspond better with perceived quality in the unconditional setting (with *waveform* and *mel* having low PIS and IIS) than in the conditional setting. In the latter, PIS and IIS fail to reflect the incapability of the model trained on *mel* to produce clear pitches, and to faithfully reproduce the timbral characteristics of the training data. Despite this, we note that both PIS and IIS are high for that model. Conversely, for data generated in the *waveform* domain, the PIS and IIS are low, even though pitch and instrument types can be clearly perceived in informal listening tests. This suggests that the inception models are not robust to the particular artefacts of these representations and therefore not very reliable in measuring the overall generation quality.

For lossy representations (i.e., *cqt*, *mel* and *mfcc*), the quantitative evaluation may suffer from a bias introduced by the lossy compression itself. Therefore, we compute the lower bounds of each representation by encoding/decoding the dataset used for our experiments in the respective transformations, and treating that as "generated data" in the evaluation. Table 5.4 shows the results of this

Models	training (days)	sampling (s)	inversion (s)
waveform	6.1	1.31	0.00
complex	3.5	0.20	0.01
mag-if	4.5	0.24	0.02
cq-nsgt	5.3	0.46	0.03
cqt	2.1	0.09	0.03
mel	1.5	0.04	3.69
mfcc	2.0	0.07	10.80

Table 5.5 – Training, sampling and inversion times for each model

experiment. While *cqt* seems to have slightly worse lower bounds in general, the FAD of *mfcc* is worse than that of *mel*, even though there are no audible differences in the audio. Apparently the cosine-transform used to compute *mfcc* from *mel* introduces non-audible artifacts, which have considerable effect on the latent representations of the Inception model.

Table 5.5 shows the training, sampling, and inversion times associated with each model and representation. Note that training times are just rough measures, as they might be affected by variations in performance and resource availability in the training infrastructure. We can observe that, in general, representations with higher compression yield faster training and sampling times, but at the expense of slower inversion. *cqt* produces the best training, sampling, and inversion times trade-off, followed by the *complex* and *mag-if* representations.

5.2.2 Informal listening

We encourage the reader to listen to the audio examples provided in the accompaniment website.[3] *mag-if* and *complex* seem to have the best-perceived quality, and are comparable to state-of-the-art works on adversarial audio synthesis (e.g., [Engel et al., 2019, Donahue et al., 2019]). We note that every representation has specific artifacts. While *waveform* seems to suffer from general broad-band noise, in *nsgt* problems in reproducing plausible phase information sometimes lead to percussive artifacts (and frequency sweeps) at the beginning and end of a sample. The samples in other representations suffer from ringing (e.g., *complex*) or from pitch distortion (e.g., *cqt*).

Interpolation between random points in the latent space seems to produce particularly smooth transitions in *complex*, followed by *mag-if*, *cqt*, and *cq-nsgt*. The model trained on *mel* fails to faithfully reproduce the timbral characteristics of the training data, and also does not generate the required pitches in the pitch-conditional setting (it always produces the same pitch for a given z). As the training setup is the same for every representation, the reason for that is not clear.

[3] `https://sites.google.com/view/audio-synthesis-with-gans`

5.3 Conclusion

The work described in this chapter compares a variety of audio representations for the task of adversarial audio synthesis of tonal sounds. We performed quantitative and qualitative evaluation, and reported on training, generation, and inversion times. We found that *complex* and *mag-if* yield the best quantitative metrics, which is also aligned with informal listening of the generated samples. Previous work by Caracalla and Roebel [2020] demonstrated the suitability of the *complex* spectrogram for sound texture synthesis with CNNs. It is interesting to see that this extends to audio generation with GANs. We also found that evaluation metrics are generally aligned with perceived quality, but in some cases they can be sensitive to non-audible representation-specific artifacts (e.g., FAD), or yield figures which seem over-optimistic when listening to the examples (e.g., PIS and IIS). In the following chapters, we extend this work to explore other types of sound sources such as percussive sounds or music, and experiment with rich conditional information such as perceptual features (see Chapter 6) or semantically intuitive attributes (see Chapter 7).

Chapter 6

DrumGAN: Synthesis of Drum Sounds with Timbral Feature Conditioning Using GANs

Drum machines are electronic musical instruments that create percussion sounds and allow to arrange them in patterns over time. The sounds produced by some of these machines are often created synthetically using analog or digital signal processing. For example, a simple snare drum can be synthesized by generating noise and shaping its amplitude envelope [Gordon, 2002b] or, a bass drum, by combining low-frequency harmonic sine waves with dense mid-frequency components [Gordon, 2002a]. Generally, drums have been modeled following spectral models (see Chapter 3) using subtractive synthesis, or sample-based techniques (e.g., Roland TR-series). The characteristic sound of this synthesis process contributed to the cult status of electronic drum machines in the '80s.

As we have seen throughout Chapters 1 to 3, deep generative neural networks are a viable alternative to traditional signal processing methods for audio synthesis. This new paradigm allows us to steer the synthesis process by manipulating learned higher-level latent variables or by conditioning the model on preexisting descriptive information. By doing so, more intuitive controls can be devised for audio synthesis compared to those systems based on conventional, expert-driven mechanisms. In addition, as deep learning models can be trained on arbitrary data, comprehensive control over the generation process can be enabled without limiting the sound characteristic to that of a particular synthesis technique.

We have seen in Chapter 3 that GANs allow to control drum synthesis through their latent input noise [Donahue et al., 2019] and Variational Autoencoders (VAE) can be used to create variations of existing sounds by manipulating their position in a learned timbral space [Aouameur et al., 2019]. However, an essential issue when learning latent spaces in an unsupervised manner is the missing interpretability of the learned latent dimensions. This can be a disadvantage in music applications, where comprehensible interaction lies at the core of the creative process. Therefore, it is desirable to develop a system which offers expressive and musically meaningful control over its generated output. A way to achieve this, provided that suitable annotations are available, is to feed higher-level conditioning information to the model. The user can then manipulate this conditioning information in the generation process. Along this line, in previous chapters we

studied works on neural audio synthesis that incorporate pitch-conditioning [Engel et al., 2017, 2019], or categorical semantic tags [Esling et al., 2019], capturing rather abstract sound characteristics. In the case of drum pattern generation, there are approaches that can create full drum tracks conditioned on existing musical material [Lattner and Grachten, 2019].

In a recent study [Ramires et al., 2020], a U-Net is applied to neural drum sound synthesis, conditioned on continuous perceptual features describing timbre (e.g., boominess, brightness, depth). These features are computed using the Audio Commons timbre models.[1] Compared to prior work, this *continuous* feature conditioning (instead of using categorical labels) for audio synthesis provides more fine-grained control to a musician. However, this U-Net approach learns a deterministic mapping of the conditioning input information to the synthesized audio. This limits the model's capacity to capture the variance in the data, resulting in a sound quality that does not seem acceptable in a professional music production scenario.

The work described in this chapter builds upon the same idea of conditional generation using continuous perceptual features, but instead of a U-Net, we employ the Progressive Growing Wasserstein GAN (PGAN) [Karras et al., 2017] described in Chapter 4. Our contribution is two-fold. First, we employ a PGAN on the task of conditional drum sound synthesis. Second, we use an auxiliary regression loss term in the discriminator as a means to control audio generation based on the conditional features. We are not aware of previous work attempting *continuous* sparse conditioning of GANs for musical audio generation. We conduct our experiments on a dataset of a large variety of kick, snare, and cymbal sounds comprising approximately 300k samples (see Sec. 4.2). Also, we investigate whether the feature conditioning improves the quality and coherence of the generated audio. For that, we perform an extensive experimental evaluation of our model, both in conditional and unconditional settings. Following the methodology described in Sec.4.3, we evaluate our models by comparing the Inception Score (IS), the Fréchet Audio Distance (FAD), and the Kernel Inception Distance (KID). Additionally, we evaluate the perceptual feature conditioning by testing if changing the value of a specific input feature yields the expected change of the corresponding feature in the generated output. Audio samples of DrumGAN can be found on the accompaniment website.[2]

The content of this chapter is extracted from our paper:

> **Nistal, J.**, **Lattner, S.**, and **Richard, G.**. "DrumGAN: Synthesis of Drum Sounds with Perceptual Feature Conditioning using GANs." In Proceedings of the 28th International Society for Music Information Retrieval (*ISMIR*), 2020.

The rest of the chapter is organized as follows: Section 6.1 presents the Audio Set ontology and the pre-trained teacher model; in Section 6.2 we describe the experiment setup; results are presented in Section 6.3; in Section 6.4 we describe the implementation of DrumGAN as VST plugin and, in Section 6.5, we present the AI Drum-Kit; we conclude in Section 6.6.

[1] `https://github.com/AudioCommons/ac-audio-extractor`
[2] https://sites.google.com/view/drumgan

6.1 Audio-Commons Timbre Models

In this work we explore perceptually-driven adversarial audio synthesis of percussion sounds. To that end, we condition a GAN on perceptually inspired features obtained from the Audio Commons project,[3] which offers a publicly available collection of perceptual models of features that describe high-level timbral properties of the sound. These features are designed from the study of popular timbre ratings given to a collection of sounds obtained from Freesound.[4] The models are built by combining existing low-level features found in the literature (e.g., spectral centroid, dynamic-range, spectral energy ratios, etc), which correlate with the target properties enumerated below. All features are defined in the range [0-100] although we normalize them to [0-1]. We employ these features as conditioning to the generative model. For more information, we direct the reader to the project deliverable.[3]

- **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.
- **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.
- **depth**: refers to the sensation of perceiving a sound coming from an acoustic source beneath the surface. A linear regression model estimates depth from the spectral centroid of the lower frequencies, the proportion of low frequency energy and the low-frequency limit of the audio excerpt.
- **roughness**: refers to the irregular and uneven sonic texture of a sound. 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.
- **boominess**: refers to a sound with deep and loud resonant components.[5]
- **warmth**: refers to sounds that induce a sensation analogous to that caused by the physical temperature. [5]
- **sharpness**: refers to a sound that might cut if it were to take on physical form. [5]

6.2 Experiment Setup

In this section details are given about the conducted experiments, including the data used, the model architecture and training details, as well as the metrics employed for evaluation.

[3] `https://www.audiocommons.org/2018/07/15/audio-commons-audio-extractor.html`

[4] `https://freesound.org/`

[5] Description of the calculation method for this feature is not available to the authors at current time.

Dataset. For the experiments described here we use the CSL-Drums dataset, described in Section 4.2. As for the conditional features, for each audio sample in the dataset, we extract the corresponding perceptual features with the Audio Commons timbre model described in Section 6.1.

Data Representation. The model is trained on the real and imaginary components of the Short-Time Fourier Transform (STFT), which we have shown to work well in audio synthesis of tonal sounds [Nistal et al., 2021c, Gupta et al., 2021], and which we observed to perform better in percussive sounds. We compute the STFT using a window size of 2048 samples and 75% overlapping. The generated spectrograms are then simply inverted back to the signal domain using the inverse STFT.

Architecture. The proposed architecture follows the configuration described in Sec. 4.1. The input to G is a concatenation of the $n_c = 7$ *audio commons* features $\boldsymbol{c_{AC}}$, described in Section 6.1, and a random vector sampled from an independent Gaussian distribution $\boldsymbol{z} \sim \mathcal{N}_{n_z=128}(\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\sigma^2} = \mathbf{I})$ with $n_z = 128$ latent dimensions. The resulting vector with size $n_z + n_{AC} = 135$ is fed to G to generate the output signal $\boldsymbol{x} = G(\boldsymbol{z}, \boldsymbol{c_{AC}})$ as illustrated in Fig. 6.1. We use $N = 6$ scale blocks in this architecture, where the number of feature maps in each block decreases from low to high resolution scales as {256, 128, 128, 128, 64, 32}. Also, differently from our first experiment in Chapter 5, we perform up/down-sampling (respectively for G and D) of the temporal dimension just up to the 3rd scale block (i.e., just in the 0th, 1st, and 2nd scales).[6] Given a batch of either real or generated STFT audio (i.e. using the *real* and *imaginary* components of the STFT as separate channels in the input tensor), D estimates the Wasserstein distance (2.7) between the real and generated distributions [Gulrajani et al., 2017], and predicts the perceptual features accompanying the input audio in the case of a real batch, or those used for conditioning in the case of generated audio. In order to promote the usage of the conditioning information by G, we add an auxiliary Mean Squared Error (MSE) loss term to the objective function, following a similar approach as in [Odena et al., 2017], as explained in Section 4.1. This process is illustrated in Fig. 6.1.

Baseline. As mentioned in the introduction, we compare DrumGAN against a previous work tackling the exact same task (i.e., neural synthesis of drums sounds, conditioned on the same perceptual features described in Section 6.1), but using a U-Net architecture operating in the time domain [Ramires et al., 2020]. The U-Net model is trained to deterministically map the conditioning features (and an envelope of the same size as the output) to the output. The dataset used thereby consists of 11k drum samples obtained from Freesound,[7] which includes kicks, snares, cymbals, and other percussion sounds (referred to as *Freesound drum subset* in the following).

Evaluation. In addition to the evaluation metrics described in Sec. 4.3 (IS, KID, and FAD), we carry out informal listening tests and assess the model's responsiveness to the conditional input features by performing a *feature coherence* test and compare against the above-described baseline. We follow the methodol-

[6] Given that we are interested on generating only 1-second-long audio, we observed that the model performed better when only performing progressive growing of the temporal dimension in the early stages of training, while maintaining full-temporal resolution in the last scales.

[7] `www.freesound.org`

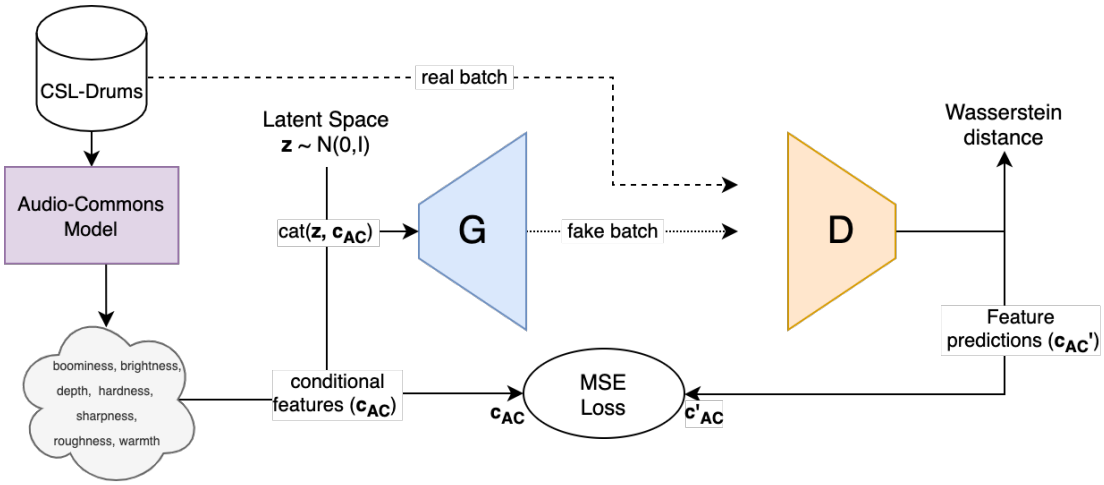


Figure 6.1 – Conditional GAN training scheme.

ogy proposed by [Ramires et al., 2020] for evaluating the feature control coherence. The goal is to assess whether increasing or decreasing a specific feature value of the conditioning input yields the corresponding change of that feature in the synthesized audio. To this end, a specific feature i is set to 0.2 (low), 0.5 (mid), and 0.8 (high), keeping the other features and the input noise fixed. The resulting outputs x^i_{low}, x^i_{mid}, x^i_{high} are then evaluated with the Audio Commons Timbre Models (yielding features fx^i). Then, it is assessed if the feature of interest changed as expected (i.e., $fx^i_{\text{low}} < fx^i_{\text{mid}} < fx^i_{\text{high}}$). More precisely, three conditions are evaluated: E1: $fx^i_{\text{low}} < fx^i_{\text{high}}$, E2: $fx^i_{\text{mid}} < fx^i_{\text{high}}$, and E3: $fx^i_{\text{low}} < fx^i_{\text{mid}}$. We perform these three tests 1000 times for each feature, always with different random input noise and different configurations of the other features (sampled from the evaluation set). The resulting accuracies are reported.

6.3 Results

In this section, we discuss on the quantitative analysis, including the comparison with the baseline U-Net architecture. Also, we briefly describe our subjective impression when listening to generated content.

6.3.1 Evaluation Metrics

Scores and Distances

Table 6.1 shows the DrumGAN results for the Inception Score (IS), the Kernel Inception Distance (KID), and the Fréchet Audio Distance (FAD), as described in Section 4.3. These metrics are calculated on the synthesized drum sounds of the model, based on different conditioning settings. Besides the unconditional setting of DrumGAN (*unconditional*), we use feature configurations from the train set (*train feats*), the valid set (*valid feats*), and features randomly sampled from a uniform distribution (*rand feats*). The IS of DrumGAN samples is close to that of the real data in most settings. This means that the model outputs are clearly assignable to either of the respective percussion-type classes (i.e., low entropy for kick, snare, and cymbal posteriors), and that it doesn't omit any of them (i.e.,

high entropy for the marginal over all classes). The IS is slightly reduced for random conditioning features, indicating that using uncommon conditioning configurations makes the outputs more ambiguous with respect to specific percussion types. While FAD is a measure for the perceived quality of the individual sounds (measuring co-variances within data instances), the KID reflects if the generated data overall follows the distribution of the real data. Therefore, it is interesting to see that *rand feats* cause outputs which overall do not follow the distribution of the real data (i.e., high KID), but the individual outputs are still plausible percussion samples (i.e., low FAD). This quantitative result is in-line with the perceived quality of the generated samples (see Section 6.3.2). In the unconditional setting, both KID and FAD are worse, indicating that feature conditioning helps the model to both generate data following the true distribution, overall, as well as in individual samples.

Table 6.2 shows the evaluation results for the U-Net architecture (see Section 6.2). As the train / valid split for the Freesound drum subset (on which the U-Net was trained) is not available to the authors, the U-Net model is tested using the features of the full Freesound drum subset (*real feats*), as well as random features. Also, we do not report the IS for the U-Net architecture, as it was trained on data without percussion-type labels, making it impossible to train the inception model on such targets. As a baseline, all metrics are also evaluated on the real data on which the respective models were trained. While evaluation on the real data is straight-forward for the IS (i.e., just using the original data instead of the generated data to obtain the statistics), both KID and FAD are measures usually *comparing* the statistics between features of real and generated data. Therefore, for the *real data* baseline, we split the real data into two equal parts and compare those with each other in order to obtain KID and FAD. The performance of the U-Net approach on both, KID and FAD is considerably worse than that of DrumGAN. While the KID for *real feats* is still comparable to that of DrumGAN (indicating a distribution similar to that of the real data), the high FAD indicates that the generated samples are not perceptually similar to the real samples. When using random feature combinations this trend is accentuated moderately in the case of FAD, and particularly in the case of the KID, reaching a maximum of almost 14. This is, however, understandable, as the output of the U-Net depends only on the input features in a deterministic way. Therefore, it is expected that the distribution over output samples greatly changes when perturbating the distribution of the inputs.

Feature Coherence

Table 6.3 shows the accuracy of the three feature coherence tests explained in Section 6.2. Note that, as both models were trained on different data, the figures of the two models are not directly comparable. However, also reporting the figures of the U-Net approach should provide some context on the performance of our proposed model. In addition, as both works use the same feature extractors and claim that the conditional features are used to shape the same characteristics of the output, we consider the figures from the U-Net approach a useful reference. We can see that for about half the features, the U-Net approach reaches close to 100% accuracy. Referring to the descriptions on how the features are computed it seems that the U-Net approach reaches particularly high accuracies for features

	↑ IS	↓ KID	↓ FAD
real data	2.26	0.05	0.00
train feats	**2.19**	0.39	0.77
val feats	2.18	**0.35**	0.76
rand feats	2.09	1.36	**0.70**
unconditional	**2.19**	1.07	1.00

Table 6.1 – Results of Inception Score (IS, higher is better), Kernel Inception Distance (KID, lower is better) and Fréchet Audio Distance (FAD, lower is better), scored by DrumGAN under different conditioning settings, against real data and the unconditional baseline. The metrics are computed over 50k samples, except for *val feats*, where 30k samples are used (i.e., the validation set size).

	↓KID	↓FAD
real data	0.04	0.00
real feats	1.45	3.09
rand feats	13.94	3.17

Table 6.2 – Results of Kernel Inception Distance (KID) and Fréchet Audio Distance (FAD), scored by the U-Net baseline [Ramires et al., 2020] when conditioning the model on feature configurations from the real data and on randomly sampled features. The metrics are computed over 11k samples (i.e., the Freesound drum subset size).

	U-Net			DrumGAN		
Feature	E1	E2	E3	E1	E2	E3
brightness	**0.99**	**0.99**	**1.00**	0.74	0.71	0.70
hardness	0.64	**0.65**	0.59	0.64	0.64	**0.62**
depth	**0.94**	0.65	**0.94**	0.79	**0.72**	0.74
roughness	0.63	0.59	0.57	**0.72**	**0.68**	**0.67**
boominess	**0.98**	**0.82**	**0.98**	0.80	0.74	0.77
warmth	**0.92**	**0.79**	**0.91**	0.76	0.71	0.71
sharpness	0.63	0.77	0.45	**0.84**	**0.82**	**0.82**
average	**0.83**	**0.76**	**0.78**	0.76	0.72	0.72

Table 6.3 – Mean accuracy for the feature coherence tests on samples generated with the baseline U-Net [Ramires et al., 2020] and DrumGAN.

which are computed by looking at the global frequency distribution of the audio sample, taking into account spectral centroid and relations between high and low frequencies (e.g., brightness and depth). U-Net performs considerably worse for features which take into account the temporal evolution of the sound (e.g., hardness) or more complex relationships between frequencies (e.g., roughness). While DrumGAN performs worse on average on these tests, the results seem to be more consistent, with less very high, but also less rather low accuracy values (note that the random-guessing baseline is 0.5 for all the tests). The reason for not performing better on average may lie in the fact that DrumGAN is trained in an adversarial fashion, where the dataset distribution is enforced, in addition to obeying the conditioned characteristics. In contrast, in the U-Net approach the model is trained deterministically to map the conditioning features to the output, which makes it easier to satisfy the simpler characteristics, like generating a lot of low- or high-frequency content. However, this deterministic mapping results in a lower audio quality and a worse approximation to the true data distribution, as it can be seen in the KID and FAD figures, described above.

6.3.2 Informal Listening

The results of the qualitative experiments discussed in this section can be found on the accompaniment website.[8] In general, conditional DrumGAN seems to have better quality than its unconditional counterpart and substantially better than the U-Net baseline (see Section 6.2). In the absence of more reliable baselines, we argue that the perceived quality of DrumGAN is comparable to that of previous state-of-the-art work on adversarial audio synthesis of drums [Donahue et al., 2019].

We also perform radial and spherical interpolation experiments (with respect to the Gaussian prior) between random points selected in the latent space of DrumGAN. Both interpolations yield smooth and perceptually linear transitions in the audio domain. We notice that radial interpolation tend to change the percussion type (i.e., kick, snare, cymbal) of the output, while spherical interpolation affects other properties (like within-class timbral characteristics and envelope) of the synthesized audio. This gives a hint on how the latent manifold is structured.

6.4 DrumGAN Plug-in

The work described in this chapter is materialized into an audio synthesis plug-in software integrating DrumGAN. The model used for this plug-in is a slightly modified version of the one presented in the previous sections. First, we scale DrumGAN to operate on high-resolution audio (i.e., 44.1 kHz sample-rate) and increase the latent space dimension from $n_z = 128$ to $n_z = 256$ to allow for a richer variety of sounds. We also remove the perceptual feature controls which, while being responsive as demonstrated in our experiments, we find them difficult to interpret in practice in order to purposely guide the synthesis towards a desired drum sound. Instead, we condition the model on *soft* instrument labels, i.e., continuous instrument class probabilities instead of one-hot class vectors. This

[8] `https://sites.google.com/view/drumgan`

way one can continuously and independently control at run-time the specific amount of each instrument class to be synthesized (i.e., some sort of "kickness", "snareness" or "cymbalness" control), enabling instrument interpolation. Finally, we also trained an encoder that enables to map any preexisting sound into the latent space of DrumGAN for its re-synthesis, enabling to generate variations of it. The interface of this plug-in is illustrated in Fig. 6.2.

Figure 6.2 – DrumGAN's Graphical User Interface (GUI) developed by Cyran Aouameur.

The resulting software was showcased at the Sony Technology Exchange Fair 2020 (STEF), an internal event for transferring technology across all Sony divisions. At STEF, DrumGAN was chosen among 10 projects from more than 300 to be demoed in front of some of Sony's executive officers, including the Vice-President. Also, the visibility in STEF helped foster collaboration around audio GAN research between Sony CSL and other departments such as Sony Interactive Entertainment (SIE), Sony R&D India, or Sony Music Japan (SMJ).[9] Furthermore, as part of an ongoing collaboration between Sony CSL and Steinberg,[10] a post-doc project is being planned aimed at further extending DrumGAN for its deployment and commercialization.

[9]As a result of the collaboration with SMJ, we created ChainsawGAN, an adaptation of DrumGAN to chainsaw sound synthesis to be used in the production of a soundtrack for the anime series *The Chainsaw Man*.

[10]https://steinberg.net

6.5 The A.I. Drum-Kit

«The A.I. Drum Kit»[11] is a collection of drums generated using DrumGAN and other DL-driven tools built at Sony CSL. It consists of 18 808-like samples, 20 kicks, 29 snares, 15 claps, 8 rimshots, 15 hi-hats, 8 open hats, and 12 percs. The collection was carefully curated by Sony ATV artist Twenty9 who is a platinum Hip-Hop producer collaborating with Sony CSL's music team. This collection melts Twenty9's know-how experience and lofi trap-like musical style with the characteristic sounds of DL-generated drums. The collection was publicly released together with a teaser (see Fig. 6.3) and can be downloaded for free.[12]

Figure 6.3 – A frame-shot from the teaser

6.6 Conclusion

In this work, we presented DrumGAN, an adversarial audio synthesizer of drum sounds. DrumGAN's generation process can be steered using perceptually motivated controls. To this end, we collected the CSL-Drums dataset, described in Section 4.2, and consisting of approximately 300k audio samples containing kicks, snares, and cymbals. We extracted a set of timbral features describing high-level semantics of the sound, and used these as conditional input to our model. We encouraged the generator to use the conditioning information by performing an auxiliary feature regression task in the discriminator and adding the corresponding MSE loss term to the objective function. In order to assess whether the feature conditioning improves the generative process, we trained a model in a completely unsupervised manner for comparison. We evaluated the models by comparing various metrics, each reflecting different characteristics of the generation process. Additionally, we compared the coherence of the feature control against previous work. Results showed that DrumGAN generates high-quality

[11] `https://csl.sony.fr/the-a-i-drum-kit-by-twenty9-and-sony-csl/`
[12] `https://twenty9.beatstars.com/`

drum samples and provides meaningful control over the audio generation. The conditioning information was proven to help the network to better approximate the real distribution of the data. Further, DrumGAN was extended and scaled to operate on high-resolution audio standards (e.g., 44.1kHz sample rate), and it was implemented in a commercially viable plug-in compatible with any Digital Audio Workstation (DAW).

Chapter 7

DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis with GANs

In Chapters 2 and 3, we reviewed some of the most outstanding works on audio and image generation using Generative Adversarial Networks (GANs) [Karras et al., 2020, Brock et al., 2019, Park et al., 2019, Engel et al., 2019, Nistal et al., 2020]. An open challenge in GANs is to learn comprehensible features that capture semantically meaningful properties of the data. This has been addressed to some extent in image generation tasks, where semantic control is achieved using semantic layouts [Park et al., 2019] or high-level attributes learned through unsupervised methods [Karras et al., 2020]. Other works achieve disentanglement of features in the data through regularization terms [Peebles et al., 2020] or by exploring the latent space of the GAN after being trained, in the search for human-interpretable factors of variation [Voynov and Babenko, 2020, Shen et al., 2020]. However, the great success of some of these approaches is partly enabled by the availability of large-scale image datasets containing rich semantic annotations [Deng et al., 2009, Caesar et al., 2018, Xiao et al., 2017]. Unfortunately, the situation is different in the musical audio domain, where datasets are scarce and often limited in size and availability of annotations.

Therefore, the work presented in this chapter studies whether limited annotations in audio datasets can be circumvented by taking a Knowledge Distillation (KD) approach (see Section 2.2). To that end, we utilize the soft labels generated by a pre-trained audio-tagging system for conditioning a GAN in an audio generation task. More precisely, we train the GAN on a subset of the NSynth dataset [Engel et al., 2017], which contains a wide range of instruments from acoustic, electronic, and synthetic sources. For that dataset we generate soft labels with a publicly available audio-tagging model [Kong et al., 2020b], pre-trained with attributes of the AudioSet ontology [Gemmeke et al., 2017]. This ontology contains a structured collection of sound events from many different sources and descriptions of around 600 attributes obtained from YouTube videos (e.g., "singing bowl", "sonar", "car", "siren", or "bird").

The soft labels produced by such audio tagging system indicate how much of the different characteristics are contained in a specific sound (e.g., a synthesizer sound may have some similarity with a singing bowl or a sonar pulse). There-

fore, it is theoretically possible that attributes that do not explicitly exist in the training data (e.g., "sonar", "singing bowl"), can still be somehow sparsely encoded across many examples. We hope that the generative model can distill such characteristics (e.g., the "essence" of a singing bowl sound) by looking at the soft labels to then be able to emphasize them at generation. The slight similarities to specific categories in data that can be distilled using soft labels were coined "Dark Knowledge" [Hinton et al., 2015]. Therefore, we call the proposed model DarkGAN.

The work contained in this chapter introduces a generic audio cross-task KD framework for transferring semantically meaningful features into a neural audio synthesizer. We implement this framework in DarkGAN, an adversarial audio synthesizer for comprehensible and controllable audio synthesis. We perform an experimental evaluation on the quality of the generated material and the semantic consistency of the learned attribute controls. Numerous audio examples are provided in the accompanying web page,[1] and the code is released for reproducibility.[2]

The content of this chapter is extracted from our paper:

> **Nistal, J.**, **Lattner, S.**, and **Richard, G.**. "DarkGAN: Exploiting Knoweldge Distillation for Comprehensible Audio Synthesis With GANs." In Proceedings of the 29th International Society for Music Information Retrieval (*ISMIR*), 2021.

In what follows, we first mention relevant state-of-the-art works in knowledge distillation, giving special attention to those works focused on audio (see Section 7.1). Next, in Section 7.2 we describe the AudioSet ontology and the pre-trained audio tagging system that we use as teacher model. We then present the experimental framework of DarkGAN (see Section 7.3). In Section 7.4 we provide a discussion of the results, and conclude in Section 7.5.

7.1 Previous Work

The Knowledge Distillation (KD) framework was briefly described in Section 2.2. As mentioned there, KD has been generally used as a model compression technique, although a few works employ it for different purposes [Papernot et al., 2017, Anil et al., 2018, Yuan and Peng, 2020]. For example, some works explore KD as a means to secure privacy of medical history training data, by releasing to the public models that are not explicitly trained on the sensible dataset, but on aggregated predictions of teacher ensembles [Papernot et al., 2017]. Other works employ KD on-the-fly as a distributed training framework to train very large models, and scale beyond the limits of distributed stochastic gradient descent [Anil et al., 2018]. An interesting line of research that is closely related to ours proposes cross-task knowledge distillation from image captioning and classification systems into an image synthesis generative neural-network [Yuan and Peng, 2018, 2020]. In audio, KD was extensively used on Automatic Speech Recognition (ASR) tasks in order to exploit large unlabelled datasets [Li et al., 2014],

[1] https://an-1673.github.io/DarkGAN.io/

[2] https://github.com/SonyCSLParis/DarkGAN

distill the knowledge from deep Recurrent Neural Networks (RNN) [Chan et al., 2015] or, inversely, to improve the performance of deep RNN models by distilling knowledge from simple models as a regularization technique [Tang et al., 2016]. Works related to ours use KD as a means to adapt a model to a different audio domain task [Asami et al., 2017] or even data modality (by distilling knowledge from a video classifier) [Aytar et al., 2016], where labeled datasets are scarce, and large models would easily overfit. Some works employ KD to fuse knowledge from different audio representations into a single compact model [Gao et al., 2020]. Finally, some works employed probability density distillation to reduce the computational complexity of WaveNet and allow parallel generation using standard feed-forward neural-networks [van den Oord et al., 2018a]. Here we employ knowledge distillation as a means to learn semantically meaningful controls in an adversarial audio synthesizer. To the best of our knowledge, this is the first time that such a task has been attempted in audio generation with GANs.

7.2 The AudioSet Ontology

AudioSet [Gemmeke et al., 2017] is a large-scale dataset containing audio data and an ontology of sound events that seeks to describe real-world sounds. It was created to set a benchmark in the development of automatic audio event recognition systems, similar to those in computer-vision, such as ImageNet [Deng et al., 2009]. The dataset consists of a structured vocabulary of 632 audio event classes and a collection of approximately 2M human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchy of categories with a maximum depth of 6 levels, covering a wide range of human and animal sounds, musical genres and instruments, and environmental sounds. We encourage the reader to visit the corresponding website for a complete description of the ontology.[3]

In this work, we do not employ all of the AudioSet attributes, as many of them refer to properties that are too vague for musical sounds or describe broader time-scale aspects of the sound (e.g., music, chatter, sound effect). Instead, we rank the attributes based on the geometric mean of their 90^{th} percentile (calculated on the predicted class probabilities for each attribute across the dataset), and the teacher's reported accuracy as $\sqrt{p^i_{90_{th}} \times acc^i}$. Then, we take the first 128 attributes according to this ranking.

7.2.1 Pre-trained AudioSet Classifier

In this work, we distill the knowledge from a pre-trained audio-tagging neural network (PANN) trained on raw audio recordings from the AudioSet collection [Kong et al., 2020b]. PANNs were originally proposed for transferring knowledge to other discriminative tasks. However, we use them to transfer the knowledge to a generative model and enable steering the generation process through a comprehensible vocabulary of attributes.

We employ the *CNN-14* model from the PANNs [Kong et al., 2020b]. *CNN-14* is built upon a stack of 6 convolution-based blocks containing 2 CNN layers with a

[3] `research.google.com/audioset/ontology/`

kernel size of 3x3. Batch Normalization is applied after every convolutional layer, and a ReLU non-linearity is used as activation function. After each convolutional block, they apply an average-pooling layer of size 2x2 for down-sampling. Global pooling is applied after the last convolutional layer to summarize the feature maps into a fixed-length vector. An extra fully-connected layer is added to extract embedding features before the output Sigmoid activation function. For more details on the architecture, please refer to Kong et al. [2020b].

7.3 Experiment Setup

In this section, details are given about the conducted experiments. We describe the AudioSet ontology, provide details about the teacher and student architectures, the metrics employed for evaluation, and the baselines used for comparison.

Dataset. For this work, we employ the NSynth dataset [Engel et al., 2017], described early on in Section 4.2. We employ all of the instrument classes (not only from the acoustic family) yielding to a subset of approximately 90k sounds with balanced instrument class distribution.

Audio representation. Following our previous work comparing representations for audio synthesis [Nistal et al., 2021c] we employ the Magnitude and Instantaneous Frequency of the STFT (*mag-if*) as it was shown to work well as a representation for tonal sounds. We use an FFT size of 2048 bins, an overlap of 75%, and a sample-rate of 16kHz.

Architecture. DarkGAN's architecture, illustrated in Fig. 4.1, follows the architecture of DrumGAN [Nistal et al., 2020] (see Chapter 6). The input to G is a concatenation of $n_{AS} = 128$ teacher-labeled AudioSet attributes $\boldsymbol{c_{AS}} \in [0, 1]^{128}$ (see Sec. 7.2), a one-hot vector $\boldsymbol{c_p} \in \{0, 1\}^{26}$ containing $n_p = 26$ pitch classes, and a random vector $\boldsymbol{z} \sim \mathcal{N}_{32}(0, 1)$ with $n_z = 32$ components. The resulting vector is placed as a column in the middle of a 4D tensor with $n_C = n_z + n_p + n_{AS} = 186$ convolutional maps. Then, it is fed through a stack of convolutional and box upsampling blocks to generate the output signal $\boldsymbol{x} = G(\boldsymbol{z}, \boldsymbol{c_p}, \boldsymbol{c_{AS}})$. We use $N = 6$ scale blocks, wherein each block the number of feature maps decreases from low to high resolution as {256, 128, 128, 128, 128, 64}. The discriminator D mirrors G's configuration and estimates the Wasserstein distance W_d between the real and generated distributions [Gulrajani et al., 2017], and predicts the AudioSet features accompanying the input audio in the case of a real batch, or those used for conditioning in the case of generated audio (see Fig. 7.1). In order to promote the usage of the conditioning information by G, we add to the objective function an auxiliary binary cross-entropy loss term for the distillation task and a categorical cross-entropy for the pitch classification task [Odena et al., 2017].

Evaluation. This work aims to learn semantically meaningful controls with DarkGAN by distilling knowledge from an audio-tagging system trained on attributes from the AudioSet ontology. Therefore, in addition to the evaluation metrics presented in Sec 4.3, we evaluate if changing an input attribute is reflected in the corresponding output of DarkGAN. To that end, we examine the change in the prediction of the teacher model (w.r.t. the output of DarkGAN) when changing a particular DarkGAN input attribute. A second property to assess is whether the *dark knowledge* helps DarkGAN learn well-formed representations of specific attributes and generalize to out-of-distribution input combinations. We

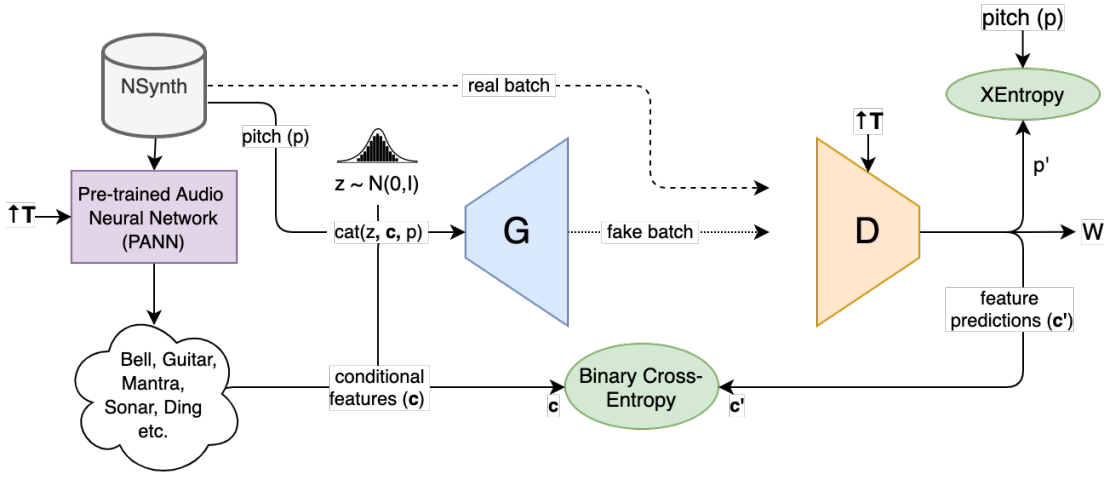


Figure 7.1 – Training diagram for DarkGAN. Note that the temperature value T is parametrizing a Sigmoid activation function in both the techer PANN and the student D, as explained in Section 2.2

.

compute these metrics for DarkGAN when trained under different temperature values in the distillation process (see Sec. 2.2), as well as for various baselines. To assess these two aspects, we perform the following tests:

1. *Attribute correlation*: we generate 10k samples using attribute vectors from the validation set as input to DarkGAN. The generated samples are fed to the teacher model to predict the attributes again. Then, for each attribute i, we compute the correlation across the 10k samples between the input vector $\boldsymbol{\alpha}$[4] and the predictions $\hat{\boldsymbol{\alpha}}$ as

$$\rho^i(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = \rho(F^i(G(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{\alpha})), \boldsymbol{\alpha_i}),$$

   where F^i is the classifier's prediction for the ith attribute, $\boldsymbol{p}$ is the pitch, and $\boldsymbol{z}$ is the random noise.

2. *Out-of-distribution Attribute Correlation*: for each attribute i exhibiting a positive correlation, i.e., $\mathcal{S} = \{\rho^i : \rho^i > 0\}$, test (1) is repeated 50 times, but using 1k samples instead of 10k. In each repetition, a specific attribute is progressively incremented by an amount $\delta_l := 10^{-3+l\frac{3.6}{50}}, l = 0, 1, ..., 50$* and we calculate

$$\overline{\rho}_{\delta_l} = \frac{1}{\mid \mathcal{S} \mid} \sum_{\mathcal{S}} \rho^i(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha} + \delta_l).$$

3. *Increment consistency*: being $\mathcal{A}$ the set containing the 50 attributes with the highest correlation, we compute

$$\overline{\Delta F_{\delta_k}} = \sum_{i \in |\mathcal{A}|} \sum_{j=1}^{100} \frac{F^i(G(\boldsymbol{z_j}, \boldsymbol{p_j}, \boldsymbol{\alpha_j} + \delta_k)) - F^i(G(\boldsymbol{z_j}, \boldsymbol{p_j}, \boldsymbol{\alpha_j}))}{50 \times 100 \times std(F^i(G(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{\alpha})))},$$

[4]In practice $\boldsymbol{\alpha} = \boldsymbol{c_{AS}}$. We employ $\boldsymbol{\alpha}$ in these explanations to remain as general as possible.

*The step of δ_l is defined to obtain more density of points in the range of variation of the attributes (i.e., [0, 1]) as well as $\delta_l > 1$.

where $\boldsymbol{\alpha_j}$ is the jth original feature vector from a set of 100 samples randomly picked from the validation set, and $\delta_k := \frac{k}{5}, k = 0, 1, ..., 25$. Intuitively, it is defined as the average difference of the predicted attributes of the generated audios (i.e., the difference before and after the attribute increment) as a function of the increment δ_k. We express the result in terms of standard deviations of the non-incremented generated examples as $\text{std}(G(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{\alpha}))$.

Baselines. We compare the evaluation metrics described above with *real data* to obtain a baseline for each metric. Also, GANSynth [Engel et al., 2019], the state-of-the-art on audio synthesis with GANs, is used for comparison.[6] As GANSynth generates 4-second long sounds, the waveform is trimmed down to 1 second for comparison with our models. Additionally, we examine the effect that KD has on these metrics by comparing against a model analogous to DarkGAN, but without using the AudioSet feature conditioning (*baseline*). Experiment results for DarkGAN are shown for different temperature values $T \in \{1, 1.5, 2, 3, 5\}$ (2.9) as part of the KD process (see Sec. 2.2.1), and we report separate results for conditional attributes obtained from the training (tr) and validation (val) set.

7.4 Results

In this section, we present the results from the evaluation procedure described in Sec. 7.3. We validate the quantitative results based on an informal assessment of the generated content.

7.4.1 Evaluation Metrics

Scores and Distances

Table 7.1 presents the metrics scored by DarkGAN$_T$, where $T \in \{1, 1.5, 2, 3, 5\}$ is the temperature value, and the baseline models, as described in Sec. 7.3. Note that we condition DarkGAN on attribute vectors randomly sampled from the validation set. Overall, DarkGAN$_{T\in\{1.5,2\}}$ obtains better results than the baselines and is close to *real data* in most metrics. All models score higher PIS than *real data*, with GANSynth in the first place, suggesting that the generated examples have a clear pitch and that the distribution of pitch classes follows that of the training data. This is not surprising, as all the models have explicit pitch conditioning. In contrast, we do not provide conditioning attributes for the instrument class. Therefore, we observe a slight drop in IIS for all models compared to *real data*. DarkGAN$_{T\in\{1.5,2\}}$ achieves the highest IIS, suggesting that the model captured the timbre diversity of the dataset and, also, that the generated sounds can be reliably classified into one of all possible instruments. In terms of KID, DarkGAN$_{T\in\{1.5,2\}}$ and *baseline* are on a par with *real data*. A KID equal to *real data* indicates that the Inception embeddings are similarly distributed for real and generated data. As our Inception classifier is trained on pitch and instrument classification and predicting AudioSet features, similarities in such an embedding space indicate common timbral and tonal characteristics between the

[6] https://github.com/magenta/magenta/tree/master/magenta/models/gansynth

Model	PIS		IIS		KID[a]		FAD	
real data	17.7		5.7		6.7		0.1	
GANSynth [Engel et al., 2019]	**19.6**		4.0		7.1		4.5	
baseline	18.5		4.3		**6.7**		0.8	
DarkGAN$_T$	tr	val	tr	val	tr	val	tr	val
$T = 1$	18.4	18.3	4.0	4.0	6.8	6.8	0.7	0.7
$T = 1.5$	19.0	19.0	**4.5**	**4.5**	**6.7**	**6.7**	0.7	0.7
$T = 2$	19.1	19.0	4.2	4.1	**6.7**	6.8	**0.6**	**0.6**
$T = 3$	19.1	19.1	4.2	4.1	6.8	6.8	0.8	0.8
$T = 5$	19.2	19.1	4.0	4.0	6.8	6.8	0.8	0.8

[a] $\times 10^{-4}$

Table 7.1 – PIS, IIS, KID and FAD (see Sec. 4.3)

generated and the real audio data distribution. This trend is maintained in the case of the FAD, where DarkGAN$_{T=2}$ obtains the best scores followed closely by DarkGAN$_{T\in\{1,1.5\}}$.

From the results discussed above, we can conclude that distilling knowledge from the AudioSet classifier helps DarkGAN learning the real data distribution. Furthermore, using slightly higher temperatures in the distillation process yields an improvement over the *baseline* without feature conditioning. We speculate that the additional supervised information that the teacher model provides to DarkGAN's discriminator results in a more meaningful gradient for the generator. Also, attribute conditioning (i.e., attribute vectors sampled from the validation set) may help the generator synthesize diverse samples closer to the training data distribution.

Attribute Coherence

Note that the metrics discussed in this section are not guaranteed to relate directly to human perception, but we consider them suitable indicators of whether the model responds coherently to the input conditioning. There exists the threat of the generator producing adversarial examples, but we argue that this is prevented by the discriminator having to satisfy the Wasserstein criterion (as adversarial examples would exhibit out-of-distribution artifacts). This assumption is also supported by informal listening tests where we find that the metrics correlate with our perception (see Sec. 7.4.2).

Table 7.2 shows some of the results for the *attribute correlation* $\rho^i(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$ for conditional feature vectors $\boldsymbol{\alpha} = \boldsymbol{c_{AS}}$ sampled from the dataset (see Sec. 7.3). The complete table can be found in Appendix B. At the top of the table, we show a few attributes corresponding to classes represented in the NSynth dataset (e.g., "guitar", "trumpet"). In the middle, we show attributes that, while not being present in the dataset (e.g., "siren", "tuning fork"), still exhibit (relatively) high correlation. At the bottom, attributes that obtain low correlations are presented (e.g., "cat", "insect"). We can observe that models trained with $T \in \{1.5, 2, 3\}$ generally obtain better results than $T \in \{1, 5\}$ in most attributes. Specifically,

Attribute	T=1	T=1.5	T=2	T=3	T=5
Accordion	0.1	0.25	0.31	**0.32**	0.10
Acoustic guitar	0.20	0.36	**0.39**	0.23	0.10
Bass guitar	0.30	0.38	**0.46**	0.38	0.19
Brass Instrument	0.28	**0.49**	0.38	0.26	0.00
Cello	0.24	**0.29**	0.26	0.17	0.00
Chime	0.15	0.33	**0.39**	0.31	0.03
Clarinet	0.12	0.29	0.37	**0.39**	-
Guitar	0.28	0.37	**0.42**	0.34	0.13
Harp	0.11	0.37	**0.41**	0.17	-
Inside, small room	0.24	**0.30**	**0.30**	0.19	-
Orchestra	0.30	**0.53**	0.47	-	
Plucked string	0.27	0.37	**0.42**	0.32	0.11
Saxophone	0.25	**0.41**	**0.41**	**0.41**	0.03
Trombone	0.18	**0.41**	0.29	0.16	0.00
Trumpet	0.16	**0.46**	0.36	0.25	0.00
Wind instrument	0.21	0.36	**0.40**	0.39	0.10
...			...		
Bicycle bell	0.11	0.16	0.08	**0.23**	0.01
Civil defense siren	0.10	0.16	**0.23**	0.09	0.06
Didgeridoo	0.06	0.16	**0.21**	0.20	0.08
Drum	0.05	0.21	**0.24**	0.12	0.01
Electronic tuner	0.35	0.44	**0.50**	0.29	0.13
Percussion	0.04	0.19	**0.30**	0.14	0.08
Sine wave	0.28	**0.32**	0.27	0.17	0.10
Singing bowl	0.08	0.20	**0.24**	0.21	0.03
Siren	0.13	0.19	**0.24**	0.10	0.08
Tuning fork	0.22	0.29	**0.35**	0.29	0.10
Zither	0.03	0.18	**0.19**	0.07	-0.01
...			...		
Cat	-0.01	-0.01	-0.01	-0.01	0.00
Chicken, rooster	0.00	-0.06	-0.02	-0.01	-0.01
Domestic animals, pets	-0.01	-0.02	-0.02	0.00	0.00
Fowl	-0.01	-0.07	-0.02	-0.02	-0.01
Frog	0.00	0.03	0.07	0.06	-0.03
Insect	0.00	-0.02	-0.02	-0.02	-0.01
Speech	-0.04	-0.10	-0.07	-0.05	0.01

Table 7.2 – A few examples of attribute correlation coefficients $\rho^i(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$ (see Sec. 7.3). The whole table can be found in Appendix B.

DarkGAN$_{T=2}$ yields the highest correlations, followed by DarkGAN$_{T=1.5}$. Note that temperatures higher than 1 also improve the correlation for attributes that do not have corresponding classes in the dataset (e.g., "didgeridoo", "percussion", "singing bowl"). This suggests that DarkGAN can extract dark knowledge (which is emphasized by increasing T) from the soft labels. The soft labels indicating the presence of (potentially just slight) timbral characteristics in various sounds are helping the model to learn linearly dependent feature controls for those attributes.

A more in-depth analysis of feature errors and the distribution of features in the dataset would be required to further characterize the results for each attribute. However, it is reasonable that those classes obtaining higher correlations share some timbral features with the training data (e.g., clearly, "violins" are contained in the data set, and a "tuning fork" is similar to a "mallet"). In contrast, those attributes obtaining low correlations may be related to underrepresented features in the training set or features that the model failed to capture.

Fig. 7.2 shows the correlation coefficient when increasing each attribute by a value δ_l in the input conditioning. The plot reveals that the trend of Table 7.2 is maintained throughout an ample range of variation of the attributes. Interestingly, while the correlation of DarkGAN$_{T=1}$ considerably declines after an increase $\delta_l > 10^{-0.8}$, using a temperature $T \in \{1.5, 2, 3\}$ the decline is more moderate, and we observe some correlation even for a $\delta_l > 1$, which is outside the range of the attributes.

As the correlation coefficient provides normalized results (regarding scale and offsets), we evaluate the attribute control using the *increment consistency* metric $\overline{\Delta F}_{\delta_k}$ (see Fig. 7.3). We observe that for low increments of the features ($\delta_k < 1$) temperatures $T \in \{1, 1.5, 2\}$ yield comparable input-output relationships of the features. A temperature $T = 1.5$, however, yields more consistent feature differences for increments $\delta_k > 1$ of the conditional input features. In conclusion, while $DarkGAN_{T=2}$ yields better correlation over all the data (i.e., conditional and predicted attributes are more strongly dependent), for attributes with particularly high correlation, $DarkGAN_{T=1.5}$ performs best in over-emphasizing dark knowledge contained in the data (i.e., the degree of change is higher, especially for $\delta_k > 1$).

7.4.2 Informal Listening

In the accompanying website,[7] we show sounds generated under various conditioning settings, including generations with feature combinations randomly sampled from the validation set, generations where we fix $\boldsymbol{\alpha}$ and $\boldsymbol{p}$ while changing $\boldsymbol{z}$, timbre transfer, scales, and more. Overall, we find the results of PIS, IIS, KID, and FAD, discussed in Sec. 7.4.1, to align well with our perception. The quality of the generated audio is acceptable for all models. Also, we find the generated examples to be diverse in terms of timbre, and the tonal content is coherent with the pitch conditioning. Moreover, we perceive that most of the attributes exhibiting high correlations (see Table 7.2) are audible in the generated output, particularly in the case of DarkGAN$_{T\in\{1,1.5,2\}}$. For higher temperatures $T \in \{3, 5\}$, the model's responsiveness to the attribute conditioning drops substantially. We find the model to be particularly responsive to attributes such as "drum", "tuning

[7] https://an-1673.github.io/DarkGAN.io/

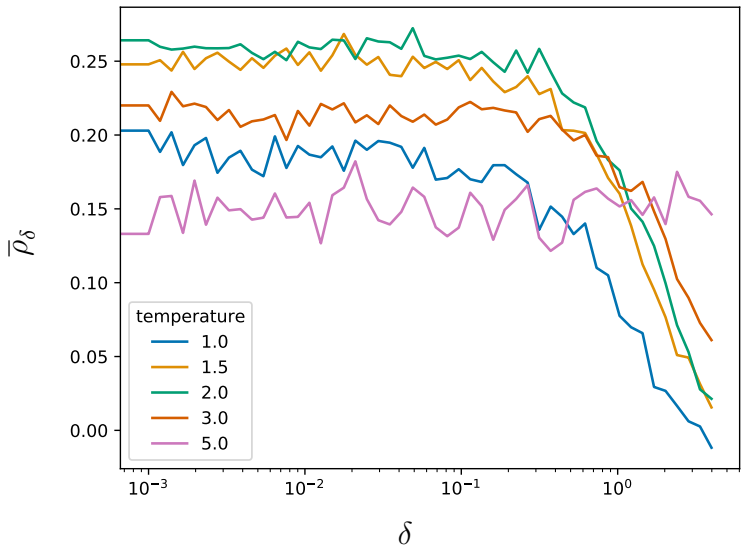


Figure 7.2 – Out-of-distribution average attribute correlation $\overline{\rho}_\delta$ (see Sec. 7.3)

fork", "theremin", "choir", or "cowbell". To other attributes (e.g., "accordion", "piano", or "organ"), even though the analysis yields moderate correlations, the model does not seem to produce perceptually satisfactory outputs.

7.5 Conclusion

In this work, we distilled knowledge from a large-scale audio tagging system into DarkGAN, an adversarial synthesizer of tonal sounds. The goal was to enable steering the synthesis process using attributes from the AudioSet ontology. A subset of the NSynth dataset was fed to a pre-trained audio tagging system to obtain AudioSet predictions. These predictions were then used to condition DarkGAN. The proposed Knowledge Distillation (KD) framework was evaluated by comparing different temperature settings and employing a diverse set of metrics. Results showed that DarkGAN can generate audio resembling the true dataset and enables moderate control over a comprehensible vocabulary of attributes. By slightly increasing the temperature during the distillation process, we can further improve the responsiveness of the attribute controls. It is also notable that KD can be performed even when the original dataset (i.e., the AudioSet collection) is not involved.

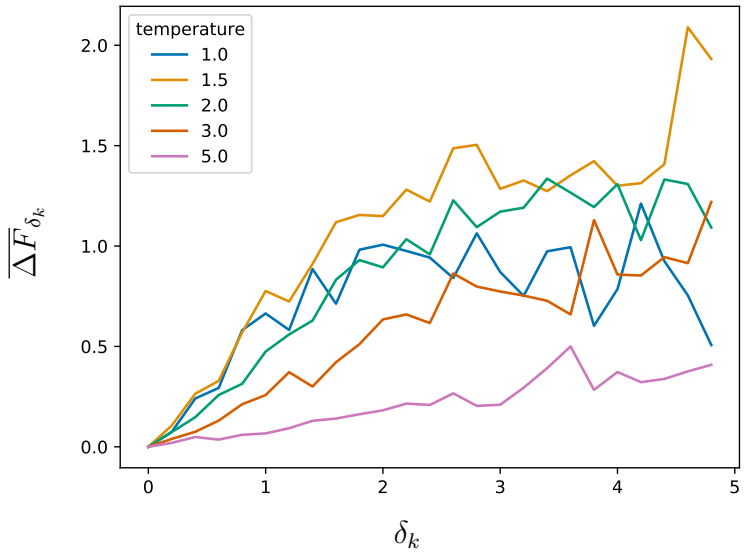


Figure 7.3 – Increment consistency $\overline{\Delta F}_{\delta_k}$ (see Sec.7.3)

Chapter 8

VQCPC-GAN: Variable-Length Adversarial Audio Synthesis Using Vector-Quantized Contrastive Predictive Coding

In recent years, Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] have shown outstanding results in image and audio synthesis tasks [Karras et al., 2017, 2020, Engel et al., 2019, Nistal et al., 2020, Binkowski et al., 2020]. As most (initial) studies on GANs focused on generating images, the resulting architectures are now often adopted for the musical audio domain, using fixed-size two-dimensional spectrogram representations as the "image data". However, while it is a natural choice to use data of fixed dimensionality in the visual domain, fixing the length of musical audio content in generation tasks poses a significant limitation. As a result, GANs are currently mainly used to generate short audio content in the musical audio domain, like single notes of a tonal instrument or single percussion samples [Engel et al., 2019, Nistal et al., 2020].

We have seen in Chapter 3 that when dealing with variable-length sequence generation, commonly utilized models are Transformer architectures [Hadjeres and Crestel, 2020], causal convolutional architectures (causal CNNs) [van den Oord et al., 2016a], and recurrent neural networks (RNNs) [Fan et al., 2014]. However, those models suffer various problems like high computational cost (autoregressive), missing look-back capabilities (recurrent), and, typically, they cannot be parallelized at test time. In contrast, GANs are relatively efficient in generating high-dimensional data, as the conditioning on a single noise vector determines the values of all output dimensions at once. Therefore, it seems reasonable to also adopt the GAN paradigm for generating variable-length musical audio content. It has been shown in text-to-speech translation [Binkowski et al., 2020] that GANs can be successful in generating coherent variable-length audio when conditioned on meaningful sequences of symbols (i.e., linguistic and pitch features), while the input noise z accounts for the remaining variability.

We adopt a similar strategy by first learning sequences of symbolic audio descriptors, serving as conditional inputs to a GAN architecture. These descriptors are discrete tokens learned through self-supervised training, using Vector-Quantized Contrastive Predictive Coding (VQCPC) [Hadjeres and Crestel, 2020]

as explained in Section 2.3. In VQCPC, discrete representations are learned through contrastive learning, by confronting positive and negative examples. In contrast to reconstruction-based VQ-VAEs, introduced by van den Oord et al. [2017], VQCPC allows to control to some extent which aspects of the (sequential) data are captured in the tokens, by carefully designing a negative sampling strategy, thus defining the so-called "pretext" task. In this work, the tokens are trained to represent *time-varying features* (i.e., something close to the envelope) of single, pitched audios of different instruments. The proposed model is conditioned on such envelope feature sequences, on the noise vector $\boldsymbol{z}$ (static, representing the "instrument"), and on pitch information (static). This approach of sequence generation with GANs using discrete tokens is promising for future, more elaborate applications. While in this work, we are simply up-sampling token sequences to generate longer sounds, one could also *generate* plausible token sequences. Such a system could then be used to hold sounds for an arbitrary time in real-time performance with a MIDI input device. Also, token sequences could be generated conditioned on MIDI information, to represent the dynamics of a target instrument. The resulting system could then be used for naturalistic rendering of MIDI files. Furthermore, training tokens to also represent pitch information would result in a more general variable-length audio generation framework. To the best of our knowledge, this is the first work implementing a variable-length GAN for *musical* audio synthesis.

The content of this chapter is extracted from our paper:

> **Nistal, J.**, **Auoameur, C.**, **Lattner, S.**, and **Richard, G.** "VQCPC-GAN: Variable-Length Adversarial Audio Synthesis using Vector-Quantized Contrastive Predictive Coding." In Workshop on Applications of Signal Processing for Audio and Acoustics (*WASPAA*), 2021.

The rest of this chapter is organized as follows. First, in Section 8.1, we summarize previous works on time-series GANs and Contrastive Predictive Coding. In Section 8.2 we describe in detail the proposed framework. Section 8.3 describes the experiment setup. Next, in Section 8.4, we evaluate the proposed method and compare results with previous work and other baselines. Finally, in Section 8.5, we draw some conclusions and discuss future directions.

8.1 Previous Work

In addition to the works presented in Chapter 3, in the following, we review some of the most important works on variable-length time-series generation using GANs and contrastive learning of sequences. We pay special attention to those works focused on audio data.

8.1.1 Time-Series GAN

Several studies have adopted the GAN framework within the sequential setting. Early approaches used recurrent neural networks (RNN) for both the generator and discriminator's architecture. The first work modeled discrete sequential musical data [Yu et al., 2017] and applied a policy gradient method to cope with the

discrete nature of the symbolic representation, using the discriminator to compute a reward judged over complete sequences. In contrast to this, C-RNN-GAN [Mogren, 2016] uses a continuous-valued representation, enabling standard backpropagation, to train the whole model end-to-end. Data is generated recurrently using an LSTM-based architecture, taking as inputs a noise vector and the previous step's generated data. Follow-up work improves C-RNN-GAN by eliminating the recursive conditional input from previous time-steps and generating a time series with just a random input vector [Esteban et al., 2017]. Most of these approaches rely only on the binary adversarial feedback for learning, which by itself may not be sufficient for the network to capture the temporal dynamics in the training data efficiently. TimeGAN [Yoon et al., 2019] is a recent work for continuous time series generation that combines the unsupervised learning framework of a GAN with an autoregressive supervised loss. The supervised objective allows for better capturing the temporal behavior of the generated time series training data. Similarly to our approach, in TimeGAN the generator is conditioned on *static* and *dynamic* sequential random vectors to account for global and temporal features. Similarly, GAN-TTS [Binkowski et al., 2020] synthesizes variable-length speech by conditioning the generator on sequential linguistic and pitch features, as well as a global random vector and a speaker ID. Taking inspiration from previous approaches, in this work, we perform variable-length audio synthesis by conditioning the generator on *static* and *dynamic* prior information. The *static* information is represented by a random vector and a pitch class, whereas the *dynamic* information is captured by a sequence of discrete tokens learned through self-supervised techniques.

8.1.2 Contrastive Predictive Coding

Contrastive Predictive Coding (CPC) [van den Oord et al., 2018b] is a self-supervised framework used to learn general features from an unlabeled dataset of sequences by contrasting *positive* and *negative* examples in a so-called pretext task. CPC has been actively studied for speech tasks [van den Oord et al., 2018b, Schneider et al., 2019, Baevski et al., 2020], where it was shown to improve the performance of ASR systems when used as front-end in replacement of spectrograms [Schneider et al., 2019]. Introducing a VQ bottleneck to the CPC improved the system's performance by discarding irrelevant information [Baevski et al., 2020, van Niekerk et al., 2020]. In contrast to previous works exploiting VQCPC for discriminative downstream tasks [van den Oord et al., 2018b, Schneider et al., 2019, Baevski et al., 2020], recent approaches explore small codebook sizes to learn compact, discrete representations of symbolic music from which to generate variations of any music piece [Hadjeres and Crestel, 2020]. We follow a similar strategy in this work and use VQCPC to condition a GAN on such discrete codes for synthesizing variable-length audio.

8.2 VQCPC-GAN

In Sec. 2.3 we briefly reviewed the theory of Contrastive Predictive Coding (CPC) and it's Vector-Quantized (VQ) variant. In what follows we describe the implementation details of the two building blocks of VQCPC-GAN, the VQCPC

encoder and the GAN.

8.2.1 VQCPC Encoder

The VQCPC schematic, depicted in Fig. 8.1, is similar to that presented in Section 2.3 but with an initial Constant-Q Transform at the input. The encoder f_{enc} is a stack of 4 convolutional blocks operating frame-by-frame. Each block is composed of a 1D CNN (in time) with a kernel size of 1 and number of channels (512, 512, 256, d_z) respectively for each block in the stack. Each CNN, except the last one, is followed by a ReLU activation function. As opposed to [Chen et al., 2020], there is no projection head. VQ is trained with a squared $\mathcal{L}_2$ loss with a commitment component [van den Oord et al., 2017]. We choose a codebook $\mathcal{C}$ containing $C = 16$ centroids, and where $d_c = d_z = 32$. The codebook size is chosen relatively small, enforcing an information bottleneck that only lets through the most salient information needed to discriminate between positive and negative examples [Hadjeres and Crestel, 2020]. The autoregressive model f_{ar} is a 2-layer GRU with a hidden size of 256 and an output size of 512, and we use its output at timestep t as the context vector h_t to predict $K = 5$ timesteps into the future. The overall training objective is the VQ and the InfoNCE loss (2.11).

As mentioned earlier, an important choice in contrastive learning is the design of the negative sampling strategy, as this controls what features are represented by the encoding. Usually, the proposal distribution for the negative samples is chosen to be uniform over the training set [Hadjeres and Crestel, 2020, van den Oord et al., 2018b, Hénaff et al., 2019]. However, in this work, we sample 16 negative examples in an intra-sequence fashion: given an audio excerpt $\boldsymbol{x}$, the negative examples are all drawn from a uniform distribution over $\boldsymbol{x}$ (i.e., the same audio excerpt). This intra-sequence sampling forces the network to encode only information which varies *within a sample* (i.e., dynamic information such as onset, offset, loudness change, vibrato, tremolo, etc.), while ignoring static information like instrument type and pitch. This shows that VQCPC provides a convenient way to control what should be represented by the discrete representations. In this work, the remaining information (instrument type and pitch) is represented by the GAN's input noise and the explicit pitch conditioning.

8.2.2 GAN Architecture

The proposed WGAN is inherited from DrumGAN [Nistal et al., 2020] although it slightly differs from that presented in Section 4.1. We adapt the architecture to a sequential scheme by conducting two major changes. First, the input tensor to the generator G is a sequence containing *static* and *dynamic* information. The *static* information refers to the global context and accounts for the pitch class, a one-hot vector $\boldsymbol{p} \in \{0, 1\}^{26}$ with 26 possible pitch values, as well as a noise vector $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^{128}$ sampled from a standard normal distribution with zero mean and unit variance $\mathcal{N}(0, \mathbf{I})$. The *dynamic* information provides local frame-level context and is composed of a sequence of discrete, one-hot vectors $\boldsymbol{c} = [c^1, ..., c^L]$ where $c^l \in \{0, 1\}^{16}$ and L is the number of frames in the sequence. The tensor $\boldsymbol{c}$ identifies a sequence of spectrogram clusters obtained by encoding real audio using VQCPC (see Sec. 2.3). At training time, L is set to 32 frames, which cor-

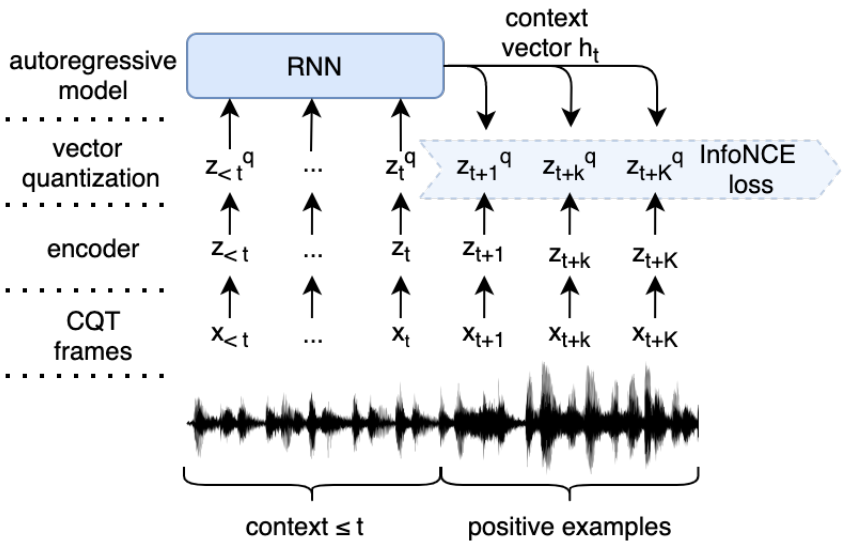


Figure 8.1 – Updated schematic of VQCPC incorporating the Constant-Q Transform (CQT).

responds to approximately 1 second of audio given the pre-processing parameters (see Sec. 8.3). The *static* vectors $\boldsymbol{p}$ and $\boldsymbol{z}$ are repeated across the sequence dimension L of the *dynamic* information $\boldsymbol{c}$, resulting in a tensor $\boldsymbol{v} \in \mathbb{R}^{L \times 160}$. This tensor is unsqueezed, reshaped to $(160 \times 1 \times L)$ and fed through a stack of convolutional and nearest-neighbour up-sampling blocks to generate the output signal $\boldsymbol{x} = G(\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{p})$. In order to turn the input tensor into a spectrogram-like convolutional input, it is first zero-padded in the frequency dimension. As depicted in Fig. 8.2, the generator's input block performs this zero-padding followed by two convolutional layers with ReLU non-linearity. Each scale block is composed of one nearest-neighbour up-sampling step at the input and two convolutional layers with filters of size $(3, 3)$. The number of feature maps decreases from low to high resolution as $\{512, 256, 256, 256, 256, 128\}$. We use Leaky ReLUs as activation functions and apply pixel normalization.

The second major change is the use of two discriminators (see Fig. 8.2). A local discriminator D_l, implemented in a fully convolutional manner, estimates $\boldsymbol{W_{local}}$ which is the Wasserstein distance [Gulrajani et al., 2017] between real and generated distributions at a frame-level (i.e. using batches of frames instead of batches of full spectrograms). Additionally, to encourage G to consider the conditional sequence of VQCPC tokens, D_l performs an auxiliary classification task where each input spectrogram frame is assigned to a VQCPC token c_l. As illustrated in Fig. 8.3, we add an additional cross-entropy loss term for D_l's objective [Odena et al., 2017]. A global discriminator D_g with two dense layers in its output block estimates W_{global} over complete sequences of $L = 32$ spectrogram frames and predicts the pitch class. As in D_l, we add an auxiliary cross-entropy loss term to D_g's objective function for the pitch classification task [Odena et al., 2017].

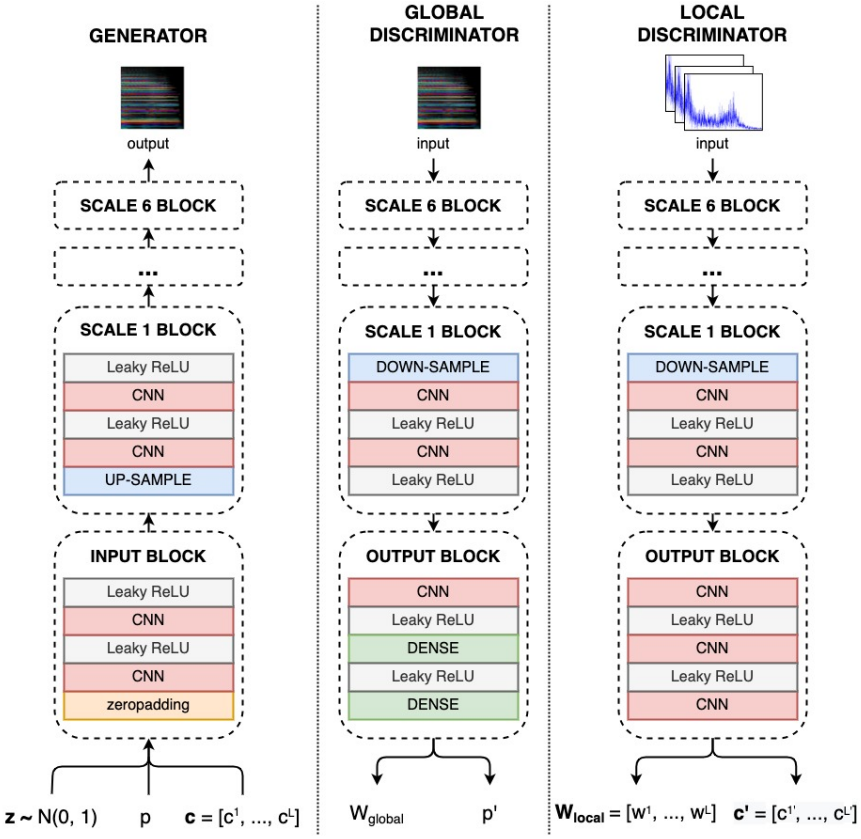


Figure 8.2 – Proposed architecture for VQCPC-GAN (see Sec. 8.2.2).

8.3 Experiment Setup

In this work, we employ a VQCPC encoder (see Sec. 2.3) to learn discrete sequences of high-level features from a dataset of tonal sounds (see Sec. 4.2). As described in Sec. 4.1, we condition a GAN on such discrete sequential representations in order to perform audio synthesis. Variable-length audio is achieved by up/down-sampling, respectively for longer or shorter sounds, of the conditional VQCPC sequence. In the following, we present the training dataset, the evaluation metrics and the baselines.

Dataset. As explained in Section 4.2, we employ subset of audio excerpts obtained from the NSynth dataset [Engel et al., 2017].

Audio representation. As in DarkGAN, we preprocess the data to obtain magnitude and IF spectrograms. We employ an FFT size of 2048 bins and an overlap of 75%. For the VQCPC encoder (see 8.2.1), we rely on the Constant-Q Transform (CQT) spanning 6 octaves with 24 bins per octave. We use a hop-length of 512 samples for the output token sequence to match the temporal resolution of the data used to train the GAN.

Evaluation. We compare the metrics described in Sec. 4.3 with a few baselines and include results scored by *real data* to delimit the range of each metric. Specifically, we compare with GANSynth [Engel et al., 2019], obtained from Google Magenta's github,[1] and two baselines that we train using the same architecture of VQCPC-GAN but removing the sequence generation scheme, i.e.,

[1] https://github.com/magenta/magenta/tree/master/magenta/models/gansynth

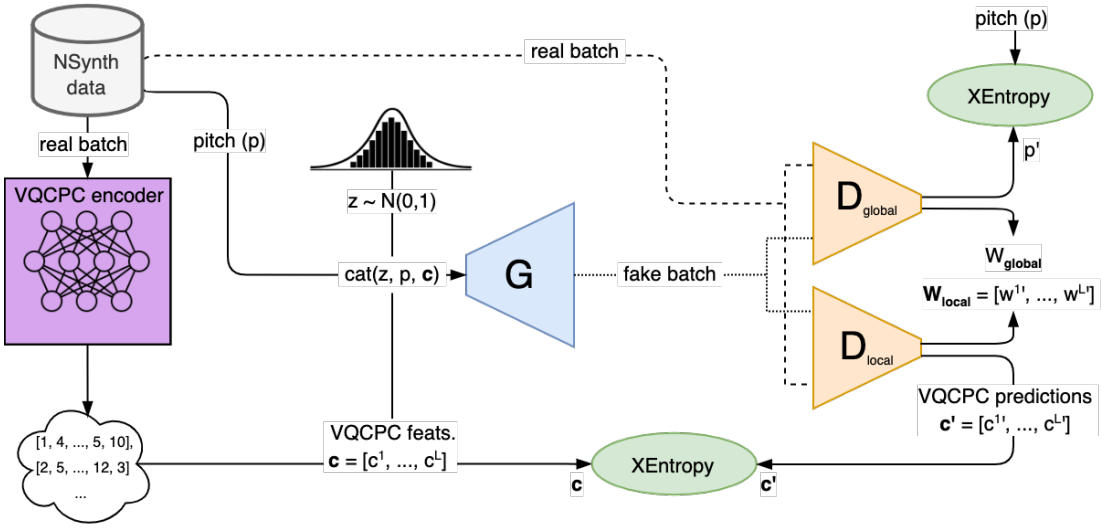


Figure 8.3 – Proposed architecture for VQCPC-GAN (see Sec. 8.2.2).

without VQCPC conditioning nor the local D. We train the two baseline models, WGAN_{1s} and WGAN_{4s}, on $1s$ and $4s$-long audio excerpts, respectively, whereas GANSynth is originally trained on $4s$ audio excerpts. As mentioned early on in this section, we condition VQCPC-GAN on varying-length VQCPC sequences in order to generate audio with different duration. To do so, we just up/down-sample the VQCPC sequence accordingly to obtain the desired number of output frames. In particular for these experiments, we take the original VQCPC sequences of length 32 (i.e., $1s$-long) and perform nearest-neighbour up-sampling by a factor of 4 to obtain 128 tokens (i.e., $4s$-long).

8.4 Results

In this section, we present the results from the evaluation metrics described in Sec. 4.3. We informally validate these quantitative results by listening to the generated content and sharing our assessment.

8.4.1 Evaluation Metrics

Table 8.1 presents the metrics scored by our proposed VQCPC-GAN and the baselines. Overall, our WGANs score closest to those of *real data* in most metrics, or even better in the case of the PIS. GANSynth follows closely and VQCPC-GAN obtains slightly worse results. VQCPC-GAN performs particularly good in terms of PIS, which suggests that the generated examples have an identifiable pitch content and that the distribution of pitch classes follows that of the training data. This is not surprising given that the model has explicit pitch conditioning, making it trivial to learn the specific mapping between the pitch class and the respective tonal content. Conversely, results are worse in the case of IIS, suggesting that the model failed to capture the timbre diversity existent in the dataset and that generated sounds cannot be reliably classified into one of all possible instrument types (i.e. mode failure). Turning now our attention to the KID, VQCPC-GAN scores results very similar to GANSynth and slightly worse than WGAN_{1s}. A low

	IIS		PIS		KID[a]		FAD	
duration (s)	1	4	1	4	1	4	1	4
real data	6.3	4.5	17.9	18.0	6.7	6.6	0.0	0.0
GANSynth [Engel et al., 2019]	-	4.1	-	19.7	-	7.0	-	2.1
WGAN_{1s}	**4.5**	-	**19.0**	-	**6.8**	-	**0.8**	-
WGAN_{4s}	-	**4.5**	-	**20.1**	-	**6.9**	-	**1.0**
VQCPC-GAN	3.0	2.9	18.5	17.2	7.3	7.1	5.6	5.4

[a] $\times 10^{-4}$

Table 8.1 – IIS, PIS, KID, and FAD (Sec. 4.3), scored by VQCPC-GAN and baselines. The metrics are computed over 25k samples.

KID indicates that the Inception embeddings are similarly distributed for real and generated data. Our Inception classifier is trained on several discriminative tasks of specific timbral attributes, including pitch and instrument classification. Therefore, we can infer that similarities in such embedding space indicate shared timbral and tonal characteristics, from a statistical point of view, between real and generated audio data. This trend is not as evident in the case of the FAD, where VQCPC-GAN obtains considerably worse results than the baselines, particularly in the case of WGAN_{1s}. This could indicate the existence of artefacts as FAD was found to correlate well with several artificial distortions [Kilgour et al., 2018].

To wrap up: despite the architectural changes introduced for sequential generation, VQCPC-GAN exhibits results comparable to GANSynth, the SOTA on adversarial audio synthesis of tonal sounds, as well as two strong baselines $\text{WGAN}_{1,4s}$ trained on 1 and 4-second long audio respectively. Notably, our WGAN_{4s} baseline scores better results than GANSynth in all metrics. In the following section, we informally validate these quantitative results by sharing our assessment when listening to generated audio material.

8.4.2 Informal Listening

The accompanying website contains audio examples generated under different settings (e.g. latent interpolations, pitch scales, generation from MIDI files) and different duration (0.5, 1, 2 and 4 seconds). Synthesis of variable-length audio is achieved by up/down-sampling of the conditional VQCPC sequence. Overall, we find the results discussed in Sec. 8.4.1 to align well with our perception. The range of instruments is narrow, and only a few from the most homogeneous and populated classes in the dataset can be identified (e.g., mallet, guitar, violin), hence the low IIS. In the *pitch scale* examples, we can perceive that the pitch content responds nicely to the conditional signal, and it is consistent across the generation time span, which explains the higher PIS. Although we eventually obtain some artifacts when using certain VQCPC token combinations as conditional input, the overall quality is acceptable. This is aligned with having a low FAD but a KID comparable to the baselines.

8.5 Conclusion

In this work, we presented VQCPC-GAN, an adversarial model capable of performing variable-length sound synthesis of tonal sounds. We adapted the WGAN architecture found in previous works [Engel et al., 2019, Nistal et al., 2020] to a sequential setting by conducting two major architectural changes. First, we condition G on *dynamic* and *static* information captured, respectively, by a sequence of discrete tokens learned through VQCPC, and a global noise $\boldsymbol{z}$. Additionally, we introduce a secondary fully-convolutional D that discriminates between real and fake data distributions at a frame level and predicts the VQCPC token associated with each frame. Results showed that VQCPC-GAN can generate variable-length sounds with controllable pitch content while still exhibiting results comparable to previous works generating audio with fixed-duration. We provide audio examples in the accompanying website. As future work, we plan on investigating hierarchical VQCPC tokens to condition the GAN on longer-term, compact representations of audio signals.

Chapter 9

Stochastic Restoration of Heavily Compressed Musical Audio using GANs

An exciting direction for future research is using GANs to learn rather complex musical relationships between input-output audio pairs, e.g., generating a drum loop given some preexisting recording of a bass-line as musical context. This task can be framed as a domain translation problem. However, to train a GAN on such a task, large amounts of multi-track musical audio data would be required, i.e., a dataset where each of the individual audio tracks that compose a music piece is available as separate audio files. Gathering a large-scale dataset with such characteristics is challenging. Therefore, in this chapter, we address a pretext task that is simpler and for which we can create the audio data pairs artificially: audio enhancement of compressed musical audio. Initially, the audio enhancement field aimed to bridge legacy technologies for music storage, processing, and transmission with current audio quality standards. Here we present a first attempt at learning musical audio transformations with GANs by performing musical audio restoration of heavily compressed MP3 music excerpts. We believe such a task is a feasible first step towards modeling more complex musical audio relationships in the future. An important reason is that we can artificially create the dataset by gathering high-quality musical audio and compressing it. Also, the task is considerably more straightforward than the end goal as the network is provided with a denser context (the compressed audio data) while still exhibiting some of the fundamental challenges: generating some missing time-frequency content that is musically coherent with the conditional audio data.

The introduction of MP3 (i.e., MPEG-1 layer 3 [Brandenburg and Stoll, 1994]) was transformative in how music was stored, transmitted, and shared in digital devices and on the internet. MP3 players, sharing platforms, and streaming resulted directly from the possibility to compress audio data without noticeable perceptual compromises. Compared to *lossless* audio coding formats, which allow for a perfect reconstruction of the original PCM audio signal, *lossy* formats (like MP3) typically lead to better compression by ignoring the parts of the signal to which humans are less sensitive. This process is also called *perceptual coding* which takes into account the physio- and psychological abilities of the human auditory perception, resulting in so-called psychoacoustic models [Brandenburg,

1999].

While several different lossy audio codecs (e.g., AAC, Opus, Vorbis, AMR) exist, MP3 is undoubtedly the most commonly used. It is built upon an analysis filter bank and the modified discrete cosine transform (MDCT). In parallel, the signal is analyzed based on a perceptual model that exploits the psychoacoustic phenomena of *auditory masking* to determine sound events in the audio signal that are considered to be beyond human hearing capabilities. Based on this information, the spectral components are quantized with specific resolution and coded with variable bit allocation while keeping the noise introduced in this process below the masking thresholds [Musmann, 2006]. This process may introduce a variety of deficiencies when configured with incorrect or very extreme parameters. For example, under large compression rates, high-frequency content is susceptible to being removed, resulting in a *bandwidth loss*. *Pre-echoes* can occur when decoding very sudden sound events for which the quantization noise spreads out over the synthesis window and consequently precede the event causing the noise. Other common artifacts are so-called *swirlies* [Corbett, 2012], characterized by fast energy fluctuations in the low-level frequency content of the sound. Furthermore, there are other problems related to MP3 compression such as *double-speak* as well as a general loss of transient definition, transparency, loss of detail clarity, and more [Corbett, 2012].

Many works exist which tackle the problem of audio enhancement, including the removal of compression artifacts. The most common recent methods used for these types of problems are based on deep learning. Typically, they focus on specific types of impairments present in the audio signals (e.g., reverberation [Williamson and Wang, 2017], bandwidth loss [Kumar et al., 2020], or audio codec artifacts [Zhao et al., 2019, Fisher and Scherlis, 2016, Skoglund and Valin, 2020, Biswas and Jia, 2020, Porov et al., 2018]). Also, different types of neural network architectures have been studied for these tasks. For example, Convolutional Neural Networks (CNNs) [Park and Lee, 2017], WaveNet-like architectures [Fisher and Scherlis, 2016, Gupta et al., 2019], and UNets [Isik et al., 2020, Hu et al., 2020]. However, most of the works in this line of research tackle the enhancement of *speech* signals [Zhao et al., 2019, Skoglund and Valin, 2020, Gupta et al., 2019, Biswas and Jia, 2020, Fisher and Scherlis, 2016, Park and Lee, 2017, Kontio et al., 2007, Isik et al., 2020, Hu et al., 2020, Li and Lee, 2015, Xu et al., 2015], and only a few publications exist for *musical* audio restoration [Lagrange and Gontier, 2020, Miron and Davies, 2018, Porov et al., 2018, Deng et al., 2020]. Given the wide range of speech enhancement techniques in telephony, automatic speech recognition, and hearing aids, this focus on speech is understandable. Also, compared to musical audio signals, speech signals are easier to study, as they are more homogeneous, narrow-band, and usually monophonic. In contrast, musical audio signals, particularly in the popular music genre, are highly varied. It typically consists of multiple superimposed sources, which can be of any type, including (polyphonic) tonal instruments, percussion, (singing) voice, and various sound effects. In addition, music is typically broad-band, containing frequencies spanning over the entire human hearing range.

Given that studies on deep learning-driven audio codec artifact removal for musical audio data are underrepresented in audio enhancement research, in this work, we attempt to provide some more insights into this task. We investigate the

limits of a generative neural network model when dealing with a general popular music corpus comprising music released in the last seven decades. In particular, we are interested in the ability of the model to regenerate lost information of heavily compressed musical audio signals using a *stochastic generator* (which is not very common in audio enhancement, with [Biswas and Jia, 2020, Maiti and Mandel, 2019] being some exceptions). This work is not only relevant for the restoration of MP3 data in existing (older) music collections. In the light of current developments in musical audio generation, where full songs can already be generated from scratch [Dhariwal et al., 2020], musical audio enhancement may soon possess a much more *generative* aspect. It has already been shown that strong generative models can enhance heavily corrupted speech through resynthesis with neural vocoders [Maiti and Mandel, 2019]. Along these lines, examining a *generative* (i.e., stochastic) decoder for heavily compressed audio signals may contribute to insights about more efficient musical data storage and transmission. Today, music streaming is increasingly common, which poses issues regarding energy consumption and environmental sustainability. When accepting deviations from the original recording, higher compression rates could be reached with a generative decoder without perceptual compromises in the listening experience. Moreover, there is no single best solution for heavily compressed audio signals to recover the original version. Therefore, it may be interesting for users to generate multiple recoveries and pick the one they like most.

We introduce a Generative Adversarial Network (GAN) [Goodfellow et al., 2014] architecture for the restoration of MP3-encoded musical audio signals. We train different stochastic (with $\mathbf{z} \sim \mathcal{N}(\mu = 0, \sigma^2 = \mathbf{I})$ input) and deterministic generators on MP3s with different compression rates. Using these models, we investigate if 1) restorations of the models considerably improve the MP3 versions, 2) if we can systematically pick samples among the outputs of the stochastic generators which are closer to the original in comparison to samples drawn from the deterministic generators, and 3) if the stochastic generators generally output higher-quality restorations than the deterministic generators. To that end, we perform an extensive evaluation of the different experiment setups utilizing objective metrics and listening tests. We find that the models are successful in points 1 and 2, but the *random* outputs of the stochastic generators are approximately on par (i.e., do not improve) the overall quality compared to the deterministic models (point 3).

The content of this chapter is extracted from our paper:

> **Lattner, S.**, and **Nistal, J.** "Stochastic Restoration of Heavily Compressed Musical Audio Using Generative Adversarial Networks." MDPI, Electronics 10, no. 11: 1349, 2021.

The rest of this chapter is organized as follows. In Section 9.1 we revise previous works in bandwidth extension and audio enhancement. In Section 9.2 we provide a brief description of the proposed GAN architecture and the experimental setup. Finally, in Section 9.3 we present and discuss the results and conclude with suggestions for future work in Section 9.4. Audio examples of the work are provided in the accompanying website.[1]

[1] `https://sonycslparis.github.io/restoration_mdpi_suppl_mat/`

9.1 Related Work

This work employs Generative Adversarial Networks (GANs) to restore MP3-compressed musical audio signals to their original high-quality versions. This task falls into the intersection of audio enhancement and bandwidth extension. Therefore, we review works on both these domains.

9.1.1 Bandwidth Extension

Low-resolution audio data (i.e., audio signals with a sample rate lower than 44.1kHz) is generally preferable for storage or transmission over band-limited channels, like streaming music over the internet. Also, lossy audio encoders can significantly reduce the amount of information by removing high-frequency content, but at the expense of potentially hampering the perceived audio quality. In order to restore the quality of such truncated audio signals, bandwidth extension (BWE) methods aim to reconstruct the missing high-frequency content of an audio signal given its low-frequency content as input [Larsen and Aarts, 2005]. BWE is alternatively referred to as *audio re-sampling* or *sample-rate conversion* in the field of Digital Signal Processing (DSP), or as *audio super-resolution* in the Machine Learning (ML) literature. Methods for BWE have been extensively studied in areas like audio streaming and restoration, mainly for legacy speech telephony communication systems [Bansal et al., 2005, Gupta et al., 2019, Kontio et al., 2007, Li and Lee, 2015] or, less commonly, for degraded musical material [Lagrange and Gontier, 2020, Miron and Davies, 2018].

Pioneering works to speech BWE were originally algorithmic and operated based on a source-filter model. In such approaches, the problem of regenerating a wide-band signal is divided into finding an upper-band source and the corresponding spectral envelope, or filter, for that upper band. While methods for source generation were based on simple modulation techniques such as spectral folding and translation of a so-called low-resolution baseband [Makhoul and Berouti, 1979], the efforts focused on estimating the filter or spectral envelope [Dietz et al., 2002]. These works introduced the so-called spectral band replication (SBR) method, where the lower frequencies of the magnitude spectra are duplicated, transposed, and adjusted to fit the high-frequency content. Because in most use-cases for speech BWE the full transmission stack is controlled, most of these algorithmic methods rely on side information about the spectral envelope, obtained at the encoder from the full wide-band signal, and then transmitted within the bitstream for subsequent reconstruction at the decoder.

Learning-based approaches to speech BWE rely on large models to learn dependencies across the lower and higher end of the frequency spectrum, reducing the need for side information in the transmitted bitstream (i.e., blind BWE). Methods based on Non-negative Matrix Factorization (NMF) treat the spectrogram as a fixed set of non-negative bases learned from wide-band signals [Bansal et al., 2005]. These bases are fixed at test time and used to estimate the activation coefficients that best explain the narrow-band signal. The wide-band signal is then reconstructed by a linear combination of the base vectors weighted by the activations. These methods efficiently up-sample speech audio signals up to 22.05kHz but are sensitive to non-linear distortions due to the linear-mixing

assumption. Dictionary-based methods can significantly improve the speech quality over the NMF approach by reconstructing the high-resolution audio signals as a non-linear combination of units from a pre-defined clean dictionary [Mandel and Cho, 2015], or by casting the problem as an l1-optimization of an analysis dictionary learned from wide-band data [Dong et al., 2015].

Early works on speech BWE using neural networks inherited the source-filter methodology found in previous works. By employing spectral folding to regenerate the wide-band signal, a simple NN is used to adjust the spectral envelope of the generated upper-band [Kontio et al., 2007]. Direct estimation of the missing high-frequency spectrum was not extensively studied until the introduction of deeper architectures [Li and Lee, 2015]. Advances in computer vision [Dong et al., 2016, Isola et al., 2017] inspired the usage of highly expressive models to audio BWE, leading to significant improvements in the up-sampling ratio and quality of the reconstructed audio signal. Different approaches followed: by generating the missing time-domain samples in a process analogous to image super-resolution [Kuleshov et al., 2017], by inpainting the missing content in a time-frequency representation [Miron and Davies, 2018], or by combining information from both domains, preserving the phase information [Lim et al., 2018]. Powerful auto-regressive methods for raw audio signals based on SampleRNN [Ling et al., 2018], or WaveNet [Gupta et al., 2019] are able to increase the maximum resolution to 16 kHz and 24 kHz sample-rate, respectively, without neglecting phase information, as it is the case in most works operating in the frequency domain [Miron and Davies, 2018, Lagrange and Gontier, 2020, Bansal et al., 2005, Li and Lee, 2015, Kumar et al., 2020]. Most recent techniques using sophisticated transformer-based GANs can up-sample speech to full resolution audio at 44.1 kHz sample-rate [Kumar et al., 2020].

9.1.2 Audio Enhancement

Audio signals may suffer from a wide variety of environmental adversities: e.g., sound recordings using low-fidelity devices or in noisy and reverberant spaces; degraded speech in mobile or legacy telephone communications systems; musical material from old recordings, or heavily compressed audio signals for streaming services. Audio enhancement improves the quality of corrupted audio signals by removing noisy additive components and restoring distorted or missing content to recover the original audio signal. The field was first introduced for applications in noisy communication systems to improve the quality and intelligibility of speech signals [Loizou, 2007]. Many studies have been carried out on speech audio enhancement, e.g., for speech recognition, speaker identification and verification [Ortega-Garcia and Gonzalez-Rodriguez, 1996, Seltzer et al., 2013, Kolbæk et al., 2016], hearing assistance devices [Yang and Fu, 2005, Chen et al., 2016a], de-reverberation [Williamson and Wang, 2017], and so on. In the specific case of audio codec restoration, many different techniques exist for improvement of speech signals [Zhao et al., 2019, Fisher and Scherlis, 2016, Skoglund and Valin, 2020, Biswas and Jia, 2020], yet only few works attempt the restoration of heavily compressed *musical* audio signals [Porov et al., 2018, Deng et al., 2020].

Classic speech enhancement methods follow multiple approaches, primarily based on analysis, modification, and synthesis of the noisy signal's magnitude

spectrum and often omitting phase information. Popular strategies are categorized into spectral subtraction methods [Boll, 1979], Wiener-type filtering [Jae Lim and Oppenheim, 1978], statistical model-based [Ephraim, 1992] and subspace methods [Dendrinos et al., 1991]. These approaches have proven successful when the additive noise is stationary. However, they introduce artificial residual noise under highly non-stationary noise or reduced signal-to-noise ratios (SNR).

Recent deep learning approaches to speech enhancement outperform previous methods in terms of perceived audio quality, effectively reducing both stationary and non-stationary noise components. Popular methods learn non-linear mapping functions of noisy-to-clean spectrogram signals [Xu et al., 2015] or learn masks in a time-frequency domain representation [Williamson and Wang, 2017, Isik et al., 2020, Williamson et al., 2016]. Many architectures have been proposed: basic feed-forward DNNs [Xu et al., 2015], CNN-based [Park and Lee, 2017], RNN-based [Erdogan et al., 2015], and more sophisticated architectures based on WaveNet [Fisher and Scherlis, 2016] or U-Net [Isik et al., 2020]. GANs are also increasingly popular in speech enhancement [Pascual et al., 2017, 2019, Li et al., 2018, Donahue et al., 2018]. Pioneering works using GANs operated either on the waveform domain [Pascual et al., 2017] or on the magnitude STFT [Michelsanti and Tan, 2017]. Subsequent works mainly focused on the latter representation due to the reduced complexity compared to time-domain audio signals [Li et al., 2018, Donahue et al., 2018, Fu et al., 2019]. Recent works operating directly on the raw waveform were able to consider a broader type of signal distortions [Pascual et al., 2019] and to improve the reduction of artifacts over previous works [Phan et al., 2020]. Successive efforts were made to further reduce artefacts by, for example, taking into consideration human perception. Some works directly optimize over differentiable approximations of objective metrics such as PESQ [Fu et al., 2019]. However, these metrics correlate poorly with human perception, and some works defined the objective metric in embedding spaces from related tasks [Germain et al., 2019] or by matching deep features of real and fake batches in the discriminator's embedding space [Su et al., 2020].

The vast majority of the speech audio enhancement approaches mentioned above operate on the magnitude spectrum and ignore the phase information [Li et al., 2018, Deng et al., 2020, Miron and Davies, 2018, Donahue et al., 2018]. Researchers often reuse the phase spectrum from the noisy signal at synthesis, introducing audible artifacts that would be particularly annoying in musical audio signals. To address this, phase-aware models for speech enhancement use a complex ratio mask [Williamson et al., 2016], or, as we have seen, operate directly in the waveform domain [Pascual et al., 2019, Phan et al., 2020]. Inspired by a recent work demonstrating that DNNs implementing complex operators [Trabelsi et al., 2018] may outperform previous architectures in many audio-related tasks, new state-of-the-art performances were achieved on speech enhancement using complex representations of audio data [Isik et al., 2020, Hu et al., 2020]. Recent work was able to further improve these approaches by introducing a complex convolutional block attention module (CCBAM), and a mixed loss function [Zhao et al., 2021].

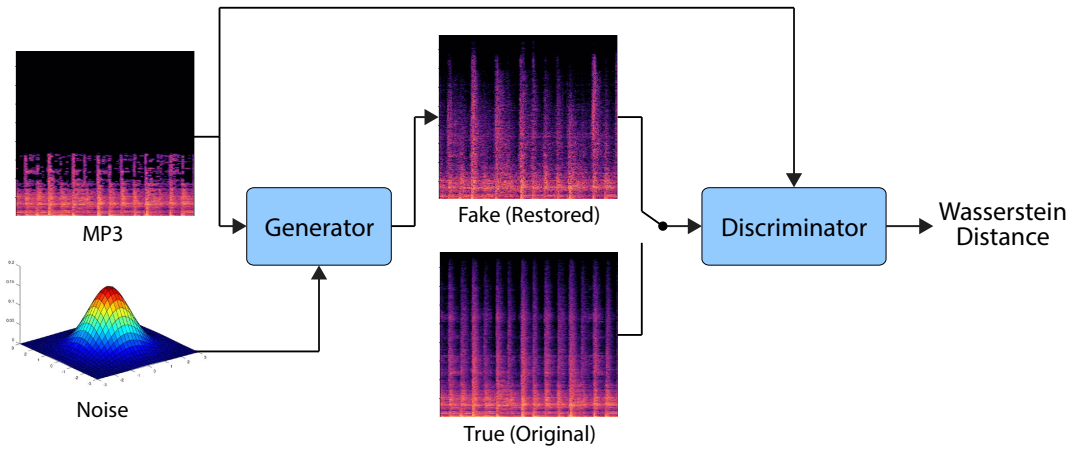


Figure 9.1 – Schematic depiction of the architecture and training procedure.

9.2 Experiment Setup

Following, we describe the experiment setup, including the model architecture, training procedure, data, and objective and subjective evaluation methods.

Dataset. The model is trained on the dataset described in Section 4.2, which contains pairs of audio data, where one part is the MP3 version, the other part is a high-quality (44.1 kHz) version of the signal. We use a dataset of approximately 64 hours of Nr 1 songs of the US charts between 1950 and 2020. The high-quality data is then compressed to 16kbit/s, 32kbit/s and 64kbit/s mono MP3 using the LAME MP3 codec, version 3.100.[2] The total number of songs is first divided into train, eval, and test sub-sets with a ratio of 80%, 10%, 10%, respectively. We then split each song into 4-second-long segments with 50% overlap for training and validation. For the subjective evaluation described below in this section, we split the songs into segments of 8 seconds.

Audio representation. The main representation used in the proposed method are the complex STFT components of the audio data $h_{j,k} \in \mathbb{C}^{JK}$, as it has been shown that this representation works well for audio generation with GANs in [Nistal et al., 2021c]. The STFT is computed with window size 2048, and a hop size of 512. In addition, we perform non-linear scaling to all complex components, in order to obtain a scaling which is closer to human perception than when using the STFT components directly. This is, we transform each complex STFT coefficient $h_{j,k} = a_{j,k} + i\, b_{j,k}$ by taking the signed square-root of each of its components $h^{\sigma}_{j,k} = \sigma(a_{j,k}) + i\, \sigma(b_{j,k})$, where the signed square-root is defined as

$$\sigma(r) = \text{sign}(r)\sqrt{|r|}. \tag{9.1}$$

Architecture. The model employed in this work follows the training framework described in Sec. 4.1 although the specific architecture implementation deviates from the one described there. Concretely, G receives as input an excerpt of an MP3-compressed musical audio signal in spectrogram representation y (i.e., non-linearly scaled complex STFT components described above) and learns to output a restored version $\hat{x}$ of that excerpt (i.e., the fake data), approximating the original, high-quality signal x (see Figure 9.1 for an overview on architecture

[2] `https://lame.sourceforge.io/` (accessed on 31 May 2021)

and training). D learns to distinguish between such restorations $\hat{x}$ and original high-quality versions of the signal x (i.e., the true data). In addition to the true/fake data, D also receives the MP3 versions of the respective excerpts. That way, it is ensured that the information present in the MP3 data is faithfully preserved in the output of G. We test *stochastic* and *deterministic* generators in our experiments. For the *stochastic* models, we also provide some noise input $z \sim \mathcal{N}(0, \mathbf{I})$, resulting in different restorations for a given MP3 input, whereas for the *deterministic* models we only provide the compressed audio. As the training criterion, we use the WGAN loss [Arjovsky et al., 2017] described in Sec. 2.1.4. The full architecture is described in Table 9.1. Both G and D are based on dilated convolutions with skip connections, combined with a novel concept which we call Frequency Aggregation Filters. These are convolutional filters spanning the whole frequency range, which contribute to the stability of the training and constitute a consequent take on the problem of non-local correlations in the frequency spectrum. We also find that using so-called self-gating considerably reduces the memory requirement of the architecture by halving the number of input maps to each convolutional layer without degradation of the results. In order to prevent mode collapse, we propose a regularization that enforces a correlation between differences in the noise input and differences in the model output. As opposed to most other works (but in line with our previous work [Nistal et al., 2021c] and other U-Net-based architectures [Isik et al., 2020, Hu et al., 2020]), we input (and output) directly the (non-linearly scaled) complex-valued spectrum to the generator, eliminating the need to deal with phase information separately. For further details about the architecture, we encourage the reader to revise our original paper [Lattner and Nistal, 2021].

Training. Each model is trained for 40k iterations and a batch size of 12, which takes about 2 days on two NVIDIA Titan RTX with 24GB memory each. We use the ADAM optimizer [Kingma and Ba, 2015] with a learning rate of 1e-3 and gradient penalty loss, to restrict the gradients of D to 1 Lipschitz [Gulrajani et al., 2017]. We also use a loss term that penalizes the magnitudes of the output of D for *real* input data, preventing the loss from drifting.

Evaluation. For this specific work, we deviate from the evaluation methodology presented in Sec. 4.3. The main goal here is to assess the similarity between the reference signals (i.e., the high-quality signals) and the signal approximations (i.e., MP3 versions of the audio excerpts or outputs of the proposed model). The employed objective metrics are standard in the audio enhancement literature: Log-Spectral Distance (LSD), Mean Squared Error (MSE), Signal-to-Noise Ratio (SNR), Objective Difference Grade (ODG), and Distortion Index (DI). We also perform a subjective evaluation in the form of the Mean Opinion Score (MOS).

- **Objective Difference Grade and Distortion Index**. The Objective Difference Grade (ODG) is a computational approximation to subjective evaluations (i.e., the *subjective difference grade*) of users when comparing two signals. It ranges from 0 to -4, where lower values denote worse similarities between the signals. The Distortion Index (DI) is a metric that is differently scaled but correlated to the ODG and can be seen as the amount of distortion between two signals. Both the ODG and DI are based on a highly non-linear psychoacoustic model, including filtering and masking to approximate the human auditory perception. They are part of the

Layer	In Maps	Out Maps	Kernel Size	Dilation	Padding	Non-linearity	Output Size
`Input`	-	-	-	-	-	-	$2 \times 1024 \times (336)[212]$
`Conv1`	2	18	3×3	1	$1, 1$	PReLU	$18 \times 1024 \times (336)[212]$
`Conv2`	18	38	3×3	2	$2, 2$	PReLU	$38 \times 1024 \times (336)[212]$
`Conv3`	38	38	3×3	4	$4, 4$	PReLU	$38 \times 1024 \times (336)[212]$
`Conv4`	38	4096	1024×1	1	$0, 0$	PReLU	$4096 \times 1 \times (336)[212]$
`Reshape1`		-	-	-	-	-	$128 \times 32 \times (336)[212]$
`ReMap`	128	256	1×1	1	$0, 0$	PReLU	$256 \times 32 \times (336)[212]$
`Conv5`	256	256	3×3	1	$1, (0)[1]$	PReLU	$256 \times 32 \times (334)[212]$
`(Noise Concat)`	-	-	-	-	-	-	$320 \times 32 \times 334$
`Conv6`	(320)[256]	256	3×3	2	$2, (0)[2]$	PReLU	$256 \times 32 \times (330)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (330)[212]$
`Conv7`	128	256	3×3	4	$4, (0)[4]$	PReLU	$256 \times 32 \times (322)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (322)[212]$
`Conv8`	128	256	3×3	8	$8, (0)[8]$	PReLU	$256 \times 32 \times (306)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (306)[212]$
`Conv9`	128	256	3×3	16	$16, (0)[16]$	PReLU	$256 \times 32 \times (274)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (274)[212]$
`Conv10`	128	256	3×3	1	$1, (0)[1]$	PReLU	$256 \times 32 \times (272)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (272)[212]$
`Conv11`	128	256	3×3	2	$2, (0)[2]$	PReLU	$256 \times 32 \times (268)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (268)[212]$
`Conv12`	128	256	3×3	4	$4, (0)[4]$	PReLU	$256 \times 32 \times (260)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (260)[212]$
`Conv13`	128	256	3×3	8	$8, (0)[8]$	PReLU	$256 \times 32 \times (244)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times (244)[212]$
`Conv14`	128	256	3×3	16	$16, (0)[16]$	PReLU	$256 \times 32 \times (212)[212]$
`SelfGating`	-	-	-	-	-	-	$128 \times 32 \times 212$
`(Reshape2)`		-	-	-	-	-	$4096 \times 1 \times 212$
`(DeConv4)`	38	4096	1024×1	1	$0, 0$	PReLU	$38 \times 1024 \times 212$
`(DeConv3)`	38	38	3×3	4	$4, 4$	PReLU	$38 \times 1024 \times 212$
`(DeConv2)`	18	38	3×3	2	$2, 2$	PReLU	$18 \times 1024 \times 212$
`(DeConv1)`	2	18	3×3	1	$1, 1$	PReLU	$2 \times 1024 \times 212$
`(Output)`		-	-	-	-	-	$2 \times 1024 \times 212$
`[Conv15]`	128	256	3×3	1	$1, 1$	PReLU	$256 \times 32 \times 212$
`[Conv16]`	256	1	32×1	1	$0, 0$	-	$1 \times 1 \times 212$

Table 9.1 – Architecture details of generator G and discriminator D for 4-second-long excerpts (i.e., 336 spectrogram frames), where $(\cdot)$-brackets mark information applying only to G, and information in $[\cdot]$-brackets applies only to D. During training, no padding is used in the time dimension for G resulting in a shrinking of its output to 212 time steps.

Perceptual Evaluation of Audio Quality (PEAQ) ITU-R recommendation (BS.1387-1, last updated 2001) [Thiede et al., 2000]. We use an openly available implementation of the basic version (as defined in the ITU recommendation) of PEAQ[3], including ODG and Distortion Index (DI). Even though PEAQ was initially designed for evaluating audio codecs with *minimal* coding artifacts, we found that the results correlate well with our perception.

- **Log-Spectral Distance**. The log-spectral distance (LSD) is the Euclidean distance between the log-spectra of two signals and is invariant to phase information. Here, we calculate the LSD between the spectrogram of the reference signal and that of the signal approximation. This results in the equation

$$LSD = \frac{1}{L}\sum_{l=0}^{L-1}\sqrt{\frac{1}{W}\sum_{f=0}^{W-1}\left[10\log_{10}\frac{P(l,f)}{\hat{P}(l,f)}\right]^2}, \tag{9.2}$$

  where P and $\hat{P}$ are the power spectra of x and $\hat{x}$, respectively, L is the total number of frames, and W is the total number of frequency bins.

- **Mean Squared Error**. The LSD described above is particularly high when comparing MP3 data with high-quality audio data. This is because it is a standard practice in many MP3 encoders (including the one we use) to perform a high-cut, removing most frequencies above a specific cut-off frequency. For values close to zero, a log-scaling introduces negative numbers with very high magnitudes. Therefore, when comparing log-scaled power spectra of MP3 and PCM, we obtain particularly high distances. This generally favors algorithms that add frequencies in the upper range (like the proposed method). In this regard, a fairer comparison is the Mean Squared Error (MSE) between the *square-root* of the power spectra P of the two signals:

$$MSE = \frac{1}{L}\sum_{l=0}^{L-1}\frac{1}{W}\sum_{f=0}^{W-1}\left[\sqrt{P(l,f)}-\sqrt{\hat{P}(l,f)}\right]^2. \tag{9.3}$$

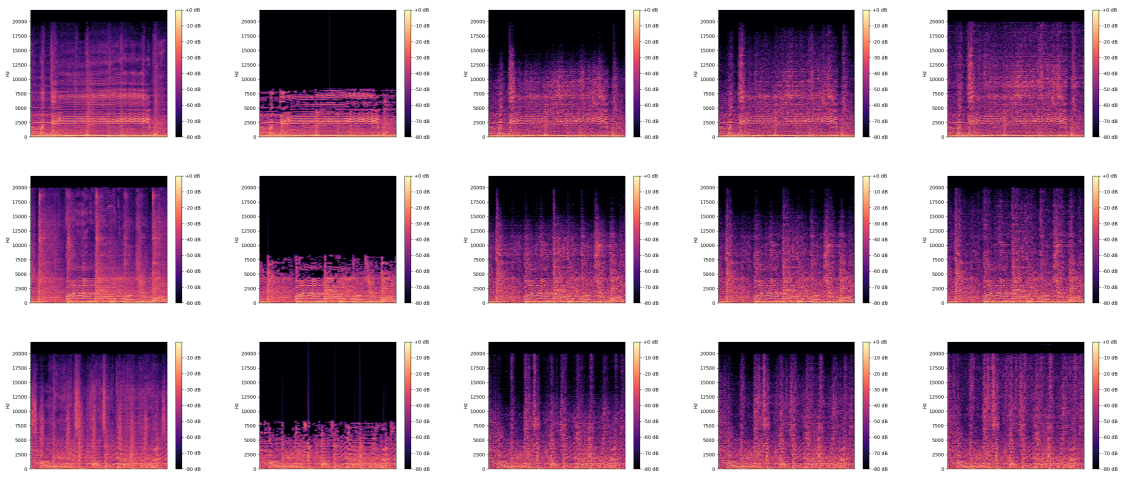
- **Signal-to-Noise Ratio**. The signal-to-noise ratio (SNR) measures the ratio between a reference signal and the approximation residuals. As it is computed in the *time domain*, it is highly sensitive to phase information. The SNR is calculated as

$$\mathrm{SNR} = 10\log_{10}\frac{\|s\|_2^2}{\|s-\hat{s}\|_2^2}, \tag{9.4}$$

  where s is the reference signal, and $\hat{s}$ is the signal approximation.

- **Mean Opinion Score**. We ask 15 participants (mostly expert listeners) to provide absolute ratings (i.e., no reference audio excerpts) of the perceptual quality of isolated musical excerpts. At the beginning of the test, the

[3] `https://github.com/akinori-ito/peaqb-fast` (accessed on 31 May 2021)



(a) (b) (c) (d) (e)

Figure 9.2 – Spectrograms of (a) original audio excerpts, (b) corresponding 32kbit/s MP3 versions, and (c), (d), (e) restorations with different noise $\boldsymbol{z}$ randomly sampled from $\mathcal{N}(0, \mathbf{I})$.

participants had a training phase where high-quality, MP3 and generated audio examples are presented. The listening test is performed with random, 8 second-long audio excerpts of the test set that could be listened as many times the listener wished. We present to the listeners 5 high-quality audio excerpts, 15 MP3s ($5 \times$ 16kbit/s, $5 \times$ 32kbit/s and $5 \times$ 64kbit/s) and 50 restored versions (using 25 stochastic restorations with random noise $\mathbf{z}$ and 25 deterministic restorations). Among these 25 restorations per model we restored $10 \times$ 16kbit/s, $10 \times$ 32kbit/s and $5 \times$ 64kbit/s MP3s. All together this results in 70 ratings per user. The participants were asked to give an overall quality score and instructed to consider both the extent of the audible frequency range and noticeable, annoying artifacts. They provided their rating using a Likert-scale slider with 5 quality levels (1) *very bad*, 2) *poor*, 3) *fair*, 4) *good* and 5) *excellent*). From these results, we compute the Mean Opinion Score (MOS) [International Telecommunications Union–Radiocommunication (ITU-T)].

Baselines. As mentioned in the introduction of this chapter, we compare stochastic and deterministic generators. Additionally, we use as reference quality the corresponding MP3 versions for each compression rate.

9.3 Results

In the following, we present the results of the performed evaluations. In Section 9.3.1 we discuss the results of the objective metrics and in Section 9.3.2 we discuss the subjective evaluation. Figure 9.2 provides a visual impression of the model output by comparing the spectrograms of some high-quality audio segments, the corresponding MP3 versions, and some restorations.

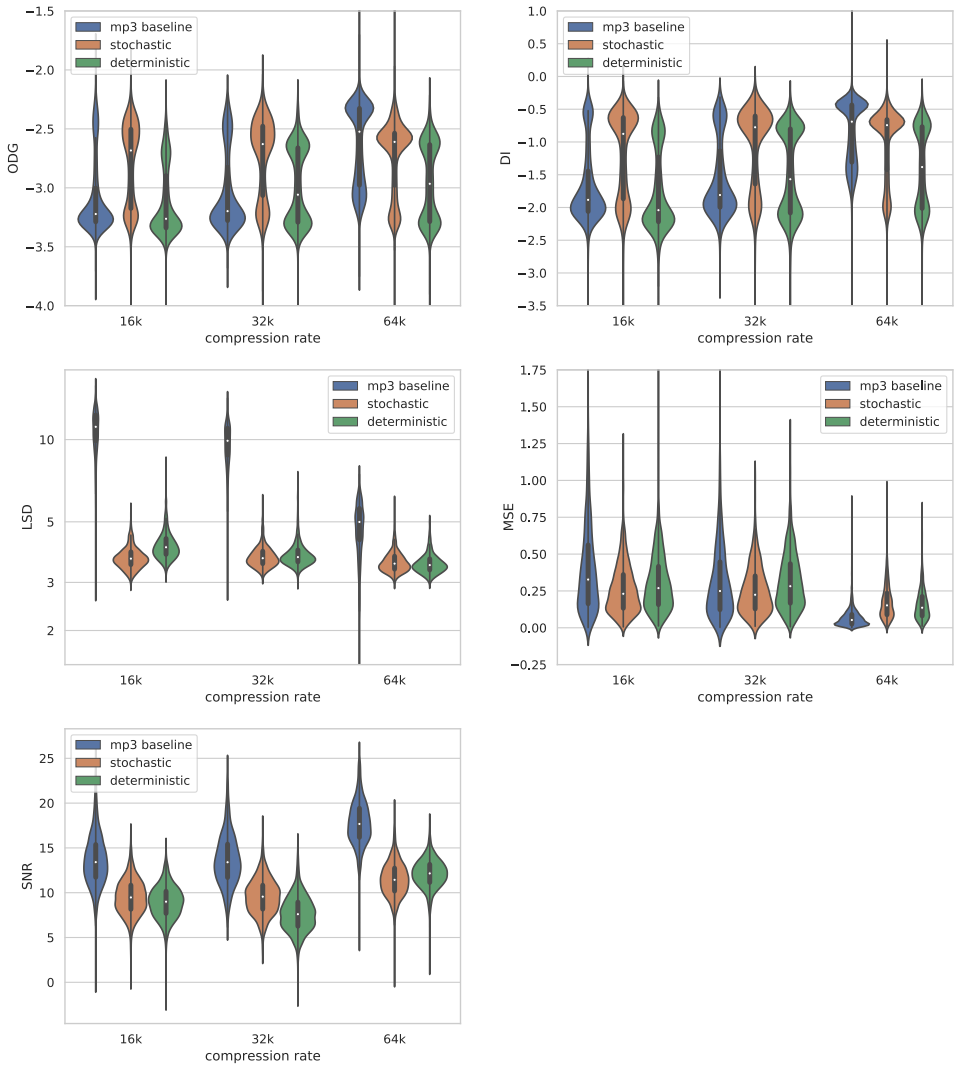


Figure 9.3 – Violin plots of objective metrics for stochastic (`sto`), deterministic (`det`) models and MP3 baselines (`mp3`), for different compression rates (16 kbit/s, 32kbit/s, 64kbit/s). Higher values are better for ODG, DI and SNR; lower values are better for LSD and MSE.

	ODG	DI	LSD	MSE	SNR
mp3_16k	-3.08	-1.67	10.98	0.40	**13.69**
det_16k	-3.12	-1.77	4.15	0.30	8.95
sto_16k	**-2.80**	**-1.19**	**3.72**	**0.26**	9.51
mp3_32k	-3.04	-1.56	9.75	0.31	**13.67**
det_32k	-2.99	-1.48	3.83	0.32	7.66
sto_32k	**-2.74**	**-1.07**	**3.75**	**0.26**	9.57
mp3_64k	**-2.64**	**-0.86**	4.89	**0.07**	**17.85**
det_64k	-2.95	-1.40	**3.54**	0.16	12.13
sto_64k	-2.74	-1.02	3.59	0.17	11.51

Table 9.2 – Results of objective metrics for stochastic (sto), deterministic (det) models and MP3 baselines (mp3), for different compression rates (16kbit/s, 32kbit/s, 64kbit/s). Higher values are better for ODG, DI and SNR; lower values are better for LSD and MSE.

9.3.1 Objective Evaluation

We test the method for three different MP3 compression rates (16kbit/s, 32kbit/s and 64kbit/s) as input to the generator. Moreover, as stated above, we assume multiple valid solutions for an MP3 to be restored with very high compression rates. This would also mean that when using a stochastic generator, some of all possible samples should be closer to the original than when only using a deterministic generator. In order to test this hypothesis, for each compression rate, we train a stochastic generator (with noise input $\boldsymbol{z}$) and a deterministic generator (without noise input). Then, for any input $\boldsymbol{y}$ taken from the test set, we sample 20 times with the corresponding generator using $\boldsymbol{z_i} \sim \mathcal{N}(0, \mathbf{I})$, and for each objective metric, we take the best value of that set. Note that all objective metrics are computed by comparing the restored data with the original versions. Therefore, when picking samples to optimize a specific metric, we do not pick the sample with the best "quality", but rather the restoration that best approximates the original.

Table 9.2 and Figure 9.3 show the results (i.e., the comparison to the high-quality data) for the stochastic and the deterministic models, and the respective MP3 baselines. For high compression rates (i.e., 16kbit/s and 32kbit/s), the best reconstructions of the stochastic models generally perform better than the baseline MP3s in most metrics and improve over the outputs of the deterministic models. This indicates that the facilitation of a stochastic generator is actually useful for restoration tasks. For some metrics (except LSD), the deterministic models perform on par with the MP3 baselines. That is reasonable, as there are many different ways to restore the original version, and it is unlikely that a deterministic model outputs a close approximation. In Figure 9.3 the strong violin-shaped forms in the figures indicate that the restorations form two groups in the ODG and DI metrics. From visual inspection of the respective data, it becomes clear that those excerpts in the lower (worse) groups are such without percussion instruments, indicating that the models cannot add meaningful high-

frequency content for, e.g., singing voice or tonal instruments. The SNR is always worse for the restorations (compared to the MP3 baselines), which shows that the phase information is not faithfully regenerated. Given the wide variety of possible phase information in the high-frequency range, particularly for percussive sounds, this is not surprising but also does not hamper the perceived audio quality.

For the 64kbit/s MP3s, we see that the reconstructions are worse than the MP3 itself, except in the LSD metric. Note that 64kbit/s mono MP3s are already close to the original. The fact that the generator performs worse on these data indicates that in addition to adding high frequency content (which is mostly advantageous, as can be seen in the LSD results), it also introduces some undesirable artifacts in the reconstruction of the MP3 information.

Frequency Profiles

In order to test the influence of the input noise $\boldsymbol{z}$ onto the generator output, we input random MP3 examples and restore them while keeping the noise input fixed. Then, we calculate the frequency profiles of the resulting outputs by taking the mean over the time dimension. Figure 9.4 shows examples of this experiment, which makes it clear that a specific $\boldsymbol{z}$ causes a characteristic frequency profile consistently over different examples. This is advantageous when $\boldsymbol{z}$ is chosen manually to control the restoration of an entire song, where a consistent characteristic is desired throughout the whole song.

9.3.2 Subjective Evaluation

In this section, we describe our own assessment when listening to the restored audio excerpts (Section 9.3.2), and then we provide results of the Mean Opinion Score (MOS) where we evaluate the restorations in a listening test with expert listeners.

Informal Listening

For sound examples of the proposed method, please refer to the accompanying website.[4] When listening to the restored audio excerpts compared to the MP3 versions, the overall impression is a richer, higher bandwidth sound that could be described as "opening up". Also, we notice that the model can remove some MP3 artifacts, particularly *swirlies*, as described in the introduction (see also [Corbett, 2012]). It is clearly audible that the model adds frequency content which got lost in the MP3 compression. When comparing the restorations directly to the high-quality versions, it is noticeable that the level of detail in the high frequencies is considerably lower in the restorations. When inspecting the restorations closer, we can hear that for specific sound events, the model performs particularly well (i.e., adds convincing high-frequency content and removes specific compression artifacts), other sources do not undergo a considerable improvement, and some events tend to cause undesired, audible artifacts.

Among the sound events which are generally improved very well are percussive elements like snare, crash, hi-hat, and cymbal sounds, but also other onsets

[4] `https://sonycslparis.github.io/restoration_mdpi_suppl_mat/`

	mean	std
original	2.81	0.94
mp3_16k	0.74	0.79
det_16k	1.33	0.82
sto_16k	**1.40**	0.89
mp3_32k	0.80	0.71
det_32k	**1.43**	0.84
sto_32k	1.28	0.82
mp3_64k	**2.92**	0.95
det_64k	2.49	0.86
sto_64k	2.65	0.74

Table 9.3 – Mean Opinion Score (MOS) of absolute ratings for different compression rates. We compare the stochastic (*sto*) versions against the deterministic baselines (*det*), the MP3-encoded lower anchors (*mp3*) and the *original* high-quality audio excerpts.

with steep transients and non-harmonic high-frequency content, like the strumming of acoustic guitars or sibilants or plosives ('s' and 't') in a singing voice. Also, sustained electric guitars undergo considerable improvement. Note that all these sound types do not possess harmonics but instead require the addition of high-frequency noise in the restoration process. Considering the nature of percussive sounds and the wide variety of sources in the training data, this is a reasonable outcome. Conversely, percussive sounds dominate other sources in the higher frequency range, which constitutes the main difference between MP3 and high-quality versions of the audio excerpts. On the other hand, harmonic sources are highly varied, and their harmonics are of different characteristics. In addition, harmonics are rarely found above 10kHz, which is the range in which the discriminator can best determine the difference between MP3 and high-quality audio signals.

Sometimes, the generator adds undesired, sustained noise, mainly when the audio input is very compressed or when there are rather loud, single tonal instruments or singing voice. Other undesired artifacts added by the generator are mainly "phantom percussions", like hi-hats that do not have meaningful rhythmic positions, triggered by events in the MP3 input that get confused with percussive sources. Also, the generator sometimes overemphasizes 's' or 't' phonemes of a singing voice. However, in some cases, percussive sounds not present in the original audio signals are added, which are rhythmically meaningful. In general, the overall characteristics of the percussion instruments are often different in the restorations compared to the high-quality versions. This is reasonable, as the lower frequencies present in the MP3 do not provide information about their characteristics in the higher frequency range, wherefore, the characteristic needs to be regenerated by the model (dependent on the input noise $\boldsymbol{z}$).

Formal Listening

Table 9.3 shows the results of the listening test (i.e., MOS ratings). Overall, the *original* and the 64kbit/s MP3s (`mp3_64k`) obtain the highest ratings and the restored 64kbit/s MP3s (`det_64k` and `sto_64k`) perform slightly worse. The ratings for the restored 16kbit/s and 32kbit/s (`det_16k`, `sto_16k`, `det_32k` and `sto_32k`) are considerably better than the MP3 versions (`mp3_16k` and `mp3_-32k`). This shows that the proposed restoration process indeed results in better perceived audio quality. However, the random samples from the stochastic generators are not assessed better than the outputs of the deterministic generators (the differences are not significant, as detailed below). We note that for the high compression rates, we reach only about half the average rating of the high-quality versions (but about double the rating of the MP3 versions). While overall, a restored MP3 version possesses a broader frequency range, weak ratings may result from off-putting artifacts, like the above-mentioned "phantom percussions". In 8-second-long excerpts, only one irritating artifact can already lead to a relatively weak rating for the whole example.

As the variance of the ratings is rather high, we also compute t-tests for statistical significance comparing responses to the different stimuli. We obtain p-values < 0.05 ($< 10^{-5}$) when comparing `det` and `sto` to `mp3` for compression rates below 64kbit/s. Conversely, we observe no statistically significant differences between ratings of `det` and `sto` for all compression rates (p-values > 0.15). Responses to *original* and `mp3_64k` also show no statistically significant differences (p-value $= 0.49$). We also observe no statistical significance between responses to `mp3_64k` and `det_64k` (p-value $= 0.06$), whereas there is a significant difference between ratings of `sto_64k` and `mp3_64k` (p-value $= 0.04$).

9.4 Conclusion

This chapter presented a GAN architecture for the stochastic restoration of high-quality musical audio signals from highly compressed MP3 versions. We tested 1) if the output of the proposed model improves the quality of the MP3 inputs, 2) if a stochastic generator improves (i.e., can generate samples closer to the original) over a deterministic generator, and 3) if the outputs of the stochastic variants are generally of higher quality than deterministic baseline models.

Results show that the restorations of the highly compressed MP3 versions (16kbit/s and 32kbit/s) are generally better than the MP3 versions themselves, which is reflected in a thorough objective evaluation, and confirmed in perceptual tests by human experts. We also tested weaker compression rates (64kbit/s mono), where we found that the proposed architecture results in slightly worse results than the MP3 baseline. We could also show in the objective metrics that a stochastic generator can indeed output samples closer to the original than when using a deterministic generator. However, the perceptual tests indicate that when drawing random samples from the stochastic generator, the results are not assessed significantly better than the results of the deterministic generator.

Due to the wide variety of popular music, the task of generating missing content is very challenging. However, the proposed models succeeded in adding high-frequency content for particular sources resulting in an overall improved

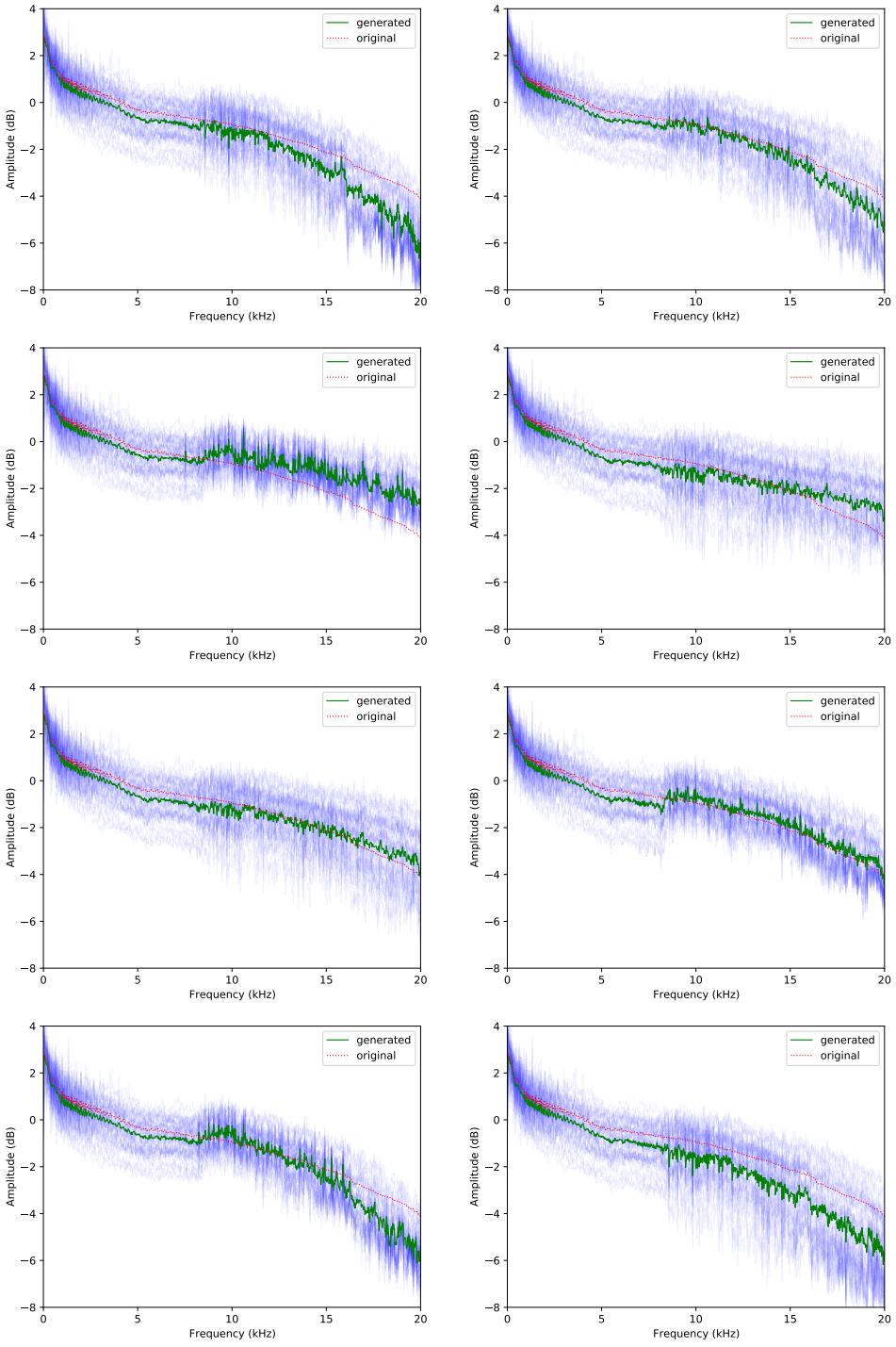


Figure 9.4 – Frequency profiles of 50 random 4-second-long excerpts from the test set (in 32kbit/s) for different random input noise vectors z. The blue lines show the profiles of the individual samples, the green line shows the mean profile of the excerpts, the dotted red line shows the mean of the high-quality excerpts for comparison. It becomes clear that $\boldsymbol{z}$ is strongly correlated with the energy in the upper bands and that a specific $\boldsymbol{z}$ yields a consistent overall characteristic.

perceived quality of the music. Examples for sources where the model clearly learned to generate meaningful high-frequency content are percussive elements (i.e., snare, crash, hi-hat and cymbal sounds), sibilants or plosives ('s' and 't') in singing voice, strummed acoustic guitars and (sustained) electric guitars. In this regard, we believe that the results presented in this work show that GANs can be a promising avenue towards learning intricate relationships between input-output musical audio pairs.

We expect future improvements when limiting the style of the training data to particular genres or time periods of production. Also, as we use the complex spectrum directly, the adaption to Complex Networks [Trabelsi et al., 2018] could improve the results further. In order to tackle the problem of "phantom percussions" (as described in Section 9.3.2), a beat detection algorithm could provide additional information to the generator so that it is better informed about the rhythmic structure of the input. For improvement in learning to restore the harmonics of tonal sources, other representations (e.g., Magnitude and Instantaneous Frequencies (Mag-IF) [Engel et al., 2019]) or a different scaling (e.g., Mel-scaled spectrograms) could be tested for the input and output of the generator.

Chapter 10

On the Development and Practice of AI Technology for Contemporary Popular Music Production

The practice of music creation has since long involved instruments and, more generally, technology. Breakthroughs and paradigm shifts resulting from the development of technology may significantly influence music-making practice (which we refer to as music production in this chapter). An example of this can be witnessed in the early use of electronic equipment for musical purposes in the recording studios during the 1950s or the popularization of synthesizers in the 1970s, allowing for artistic experimentation and enabling new ways to produce music.

Introducing novel technologies into the music creation workflow is often a complex process that initially requires specialists to take care of the technical aspects. In the case of early recording studio technology, an example of this is the collaboration between engineer Pierre Schaeffer and musician Pierre Henry, who in the '50s collaborated closely to find new musical applications of technology [Palombini, 1993]. Another example is the collaboration of analog synthesizer pioneer Robert Moog with several musicians [Pinch and Trocco, 2002]. Over time, the practice of music creation evolved to integrate new technologies, blurring the distinction between engineer, musician, and music producer [Moorefield, 2005]. In contemporary genres such as rap, dance, and, more generally, electronic music, technology has become such an integral part of the music that there is typically no meaningful distinction between the composition, recording, and production of music.

Today, as shown in Chapters 2 and 3, advances in artificial intelligence (AI), and in particular machine learning, promise to have a profound transformative effect in music practice in general. However, AI-based technology appears to be at the same stage as the recording studio was in the 1960s: the technology exists, but making it available as a tool for music production generally requires specialists (AI engineers) operating the technology and assisting musicians in its usage.

The process of technology becoming part of a new music practice does not solely rely on engineering, nor is it a matter of taking inventory of artists' needs as "user requirements" and fulfilling these requirements. The term *musical research*

has been proposed to denote this “co-adaptation" of technology and musical practice [Cont, 2013]. Several research institutes are explicitly dedicated to this type of research, such as the Institute for Research and Coordination in Acoustics/Music (IRCAM, Paris, France) or the Center for New Music and Audio Technologies (CNMAT, Berkeley, California). The Magenta Project at Google Brain, with its focus on AI-based music technology, contributes to musical research by making the technology widely available in the form of open-source software, and in some cases, sharing hardware prototypes of the technology with musicians [Engel et al., 2017].[1]

Sony CSL's music team is a musical research lab developing AI-based tools for innovation in music practice. At CSL, we believe that musicians and engineers working in unison is a vital part of the innovation process for two main reasons. On the one hand, engineers may gain knowledge of what musicians look for artistically, and, on the other hand, musicians may become aware of the opportunities the technology offers for novel music practices. In this chapter, we report on the collaborations with professional musicians, in which they experiment with different AI-based music tools developed at Sony CSL.

The content of this chapter is extracted from our paper:

> **Deruty, E.**, **Grachteen, M.**, **Lattner, S.**, **Nistal, J.**, and **Aouameur, C.**. “On the development and practice of AI technology for contemporary popular music production.” *Transactions of the International Society for Music Information Retrieval (TISMIR), 2022.*

The chapter is organized as follows. In Section 10.1 we describe the procedure under which we collaborate with artists and provide a brief overview of the tools used. In Section 10.2 we give an account of the artists' feedback, categorizing how they interact with the tools. Section 10.3 lists observations and lessons learned throughout such process, pointing out specific forms that the validation process of AI tools for music production may assume. Finally, in Section 10.4 we present the overall conclusions.

10.1 Experiment Setup

At the core of musical research is the interaction between artists and technology. At CSL, this interaction occurs in long-term collaborations with professional musicians working in various musical genres, with a strong focus on recent popular trends. Over the past years, we tested our AI-driven music production prototypes (at different stages of development) with them.

The musicians we worked with include, in no particular order: songwriters/producers Yann Macé and Luc Leroy, from company Hyper Music[2]; beatmaker/producer Twenty9, currently affiliated with Sony Music Publishing France; composer and conductor Uèle Lamore, currently affiliated with XXXIM / Sony Masterworks Berlin; Donn Healy, independent electronic music producer.

[1] `https://nsynthsuper.withgoogle.com/`

[2] `https://www.hyper-music.com/`

10.1.1 Procedure

At the start of the collaboration, we give the artists an overview of AI and machine learning applied to music and explain our vision of music AI as tools to enrich the creative workflow in music production. We give them a demonstration of the available tool prototypes in the lab where they can try out the software. When the artists are familiar with the ways the tools work, they use them in their own working environment, typically for several weeks or months, experimenting with the tools in their music production process. Typically there are follow-up sessions where the artists talk of their experience, what they like about the tools, what they dislike, and what changes they would like to see. We gather such feedback in the form of oral or written interviews, email exchanges, or presentations. Given the artist's feedback, the tools are modified accordingly as long as changes can be realized within a reasonable effort. Proposals that imply more fundamental changes to the tools are used to guide future development. When the artists have finalized their work, they send us the outcomes and a description of their workflow, which typically includes the AI tools, along with several other music production tools they work with.

10.1.2 Tools

The AI tools provided to the artists have been recently developed at our lab and are generally prototypes in the form of either standalone applications, VST plug-ins for digital audio workstations (DAW), or servers accessible through a web interface. They cover different aspects of the music production process, ranging from sound design to mixing/equalization and melodic and rhythmic material generation. The tools have been presented in more detail in prior publications, so here we provide only a brief introduction:

- **Notono**. An interactive tool for generating instrumental one-shots [Bazin et al., 2020]. It uses VAE architecture that operates on spectrograms and is conditioned on instrument labels. You can start from a sound you like and interactively modify it by inpainting the spectrogram.

- **Planet Drums, DrumGAN, Impact Drums**. Three drum sound synthesizers. *Planet Drums* is based on a VAE architecture that allows the user to explore different drum sounds by traversing a low-dimensional embedding of the latent space [Aouameur et al., 2019]. *DrumGAN* and *Impact Drums* are based on GANs [Goodfellow et al., 2014]. *DrumGAN* is conditioned on perceptual features that can be used as controls [Nistal et al., 2020].

- **DrumNet**. A tool for creating drum tracks conditioned on existing audio tracks like guitar, bass, or keyboard tracks [Lattner and Grachten, 2019]. The output adapts to the tempo and rhythm of the existing tracks, and users can explore different rhythmic variations by traversing a latent space.

- **BassNet/LeadNet**. A tool for creating bass tracks (BassNet) or lead tracks (LeadNet), conditioned on one or more existing audio tracks [Grachten et al., 2020]. The output adapts to the tonality of the existing tracks (if

the input is tonal), and users can explore different rhythmic and melodic variations of the output by traversing a latent space. The model outputs both MIDI and audio and conveys articulation, dynamics, timbre, and intonation. In terms of model architecture, BassNet and LeadNet are identical. They differ in that BassNet was trained on bass guitar tracks, and LeadNet was trained on vocal and lead guitar tracks.

- **ResonanceEQ, ProfileEQ**. Adaptive equalizers for audio mixing and mastering tasks [Grachten et al., 2019]. They consist of hand-designed processing pipelines to adjust the spectral characteristics of the sound adaptively and other feed-forward convolutional neural networks to estimate optimal control parameters for the equalizer process conditional on the input audio.

10.2 Results

Having introduced the artists we work with and the AI-driven prototypes they use, we will share some of their insights when producing music. We give them the freedom to use the tools where and how they wish, without particular guidelines or constraints. We identify and discuss some typical interaction patterns with the tools. We then use the observations to critically assess current paradigms and suggest improvements to the current state-of-the-art for interaction with AI-driven music production tools.

10.2.1 *Push* & *Pull* Interactions

In the field of creative text writing, two approaches to trigger the machine's output are described [Clark E. et al., 2018]: *Push* (automatically initiated) and *Pull* (person-initiated). Artists often use *push* interactions when starting a music piece. Musician Twenty9 testifies:

> "Flow Machines[3] (FM) is a true source of inspiration at the start of the composition process, when I face a blank page. I let myself being guided by what FM does. It allows me to spend less time on the symbolic composition, more time on the sample design, and to have fresh ideas for the drums and the bass, to have fun on the programming without being tired by the symbolic composition."

This initial *push* suggestion triggers a sequence of *pull* interactions, where the artist refines the initial idea by feeding it back to the system and stitching together new suggestions. Still according to Twenty9:

> "I really liked the chord sequence generated by FM and mostly the melody too, with a few details ready [...] I interacted with FM to recompose parts of this 16 bars melody, until I could extract 4 bars that I really liked. It was done very quickly!"

[3] `https://www.flow-machines.com/`

One particular case of *pull* is known as *priming* [Huang et al., 2020]: the artist designs an input to drive the generation process. This amounts to what is referred to as *dense conditioning* [Grachten et al., 2020], where the output of a model is controlled by providing a rich source of information (e.g. an audio or MIDI track) instead of sparser types of information that are provided by the typical UI elements of a control panel (sliders, buttons, presets...). One example of the *priming* process used in production from Yann Macé:

> "Made an 8-bar bounce with kick, snare plus a very simple legato bass part (not used thereafter). Fed this bounce to LeadNet. Tweaked around until I hear something inspiring : it plays a cool part with a 4-note hook that sounds good at the end of the chord cycle."

Note that BassNet, LeadNet, and DrumNet are designed to be conditioned on *audio input*, which has multiple advantages. First, it better integrates into music production workflows, as audio is more general than MIDI (one can always render MIDI to audio, but not the other way around). Second, audio is richer, as it combines both tonal information, expressivity, and timbral characteristics (including higher pitch resolution, without MIDI quantization). Third, it has a higher potential to result in unexpected (but valuable) outputs, as audio can carry much more variety than symbolic music representation. Donn Healy states:

> "DrumNet handled this quirky input very well, it followed the expression to a T [very precisely]",

which summarizes the points made above, namely expressivity, richness of the audio input, and surprise when using "quirky" material.

10.2.2 On Machine Interference With the Creative Process

It has been noted before that AI-driven music tools can interfere with musical goals [Huang et al., 2020]. We have experienced that, even at early stages, prototypes require format compatibility (e.g., implementation as DAW plug-ins) and compatibility with the artist's method to actually be used.

Even then, artists may be reluctant to use technology proposing musical content. However, doing so may be beneficial in terms of results. Twenty9 testifies:

> "[...]. Since I was a fan of this loop [...] I went straight to drums. Honestly, in the euphoria, I wanted to jump on my usual sampler and set a rhythm in 5 min. I forced myself to confront DrumNet [...]. To my surprise, [...] I ended up with a pattern that worked well [even though] on my own I would not have placed my kicks like that."

> "[Working with LeadNet], I am confronted with melodies that I would probably never have thought of."

AI-based approaches interfere with creative goals because they disturb the preconceived vision of the artist, and it is mostly a desirable design feature (as opposed to interference with the artist's workflow). From artists' feedback or figures such as Moog and Schaeffer, we witness creativity *emerging* from the machine's interference. It remains the artist's prerogative to *set the right conditions and remain attentive* for interesting musical combinations.

10.2.3 Exploration and Higher-level Control

As witnessed by [Huang et al., 2020], many musicians adopted the *generate-then-curate* strategy when working with specific AI-driven prototypes; they first generate many samples and then select those they deem valuable for further usage. Artist Uèle Lamore adopted such strategy when working with the prototypes:

> "The goal was to generate a selection of percussion/drum samples that I could see fit to use in any given setting. [...] generating percussion sounds with DrumGAN and Planet Drums. I'm not interested in generating sounds that sound like a "real" or "classic" kit. I want sounds that are very abstract [...] I now had this selection of sounds available."

However, note that 1) such strategies are also common outside AI-based approaches and that 2) AI appears to make such strategies more efficient. From our observations, a misconception seems to be that AI tools are doomed to spit out a lot of useless material, which then needs to be curated.

AI can potentially free artists from cumbersome workflows involving skipping through sample libraries or fiddling with numerous controls of a complex synthesizer to realize an idea. In particular, when AI model engineering and human-computer interaction is developed further *in unison*, we expect to see a shift from the generate-then-curate strategy to a more efficient and creatively enriching *exploration* and *higher-level control* paradigm. A prominent example of exploration in generative models is the navigation of latent spaces, where positions and directions have intuitive meaning so that the desired solution can be found in a more controlled way [Nistal et al., 2020, Aouameur et al., 2019, Engel et al., 2019]. Such higher-level control was also studied for the generation of minute-long material [Lattner and Grachten, 2019, Grachten et al., 2020]. Note that in these works, considerations about the user interaction was an integral part of the model design process. Keeping the artist in mind in the early stages of model engineering will increase user empowerment, efficiency, and satisfaction.

10.2.4 AI, the New Analog?

Musicians regularly take advantage of particular prototype behaviors that would not be deemed acceptable in the context of scientific validation. Specifically, behaviors that (1) would be validated as incorrect (i.e., not complying with the training data set's characteristic – "glitches"), or (2) stem from an abnormal usage of the model in which validation is not possible (e.g., out-of-domain input).

Glitches

Musicians sometimes point out that particular artifacts are great, that they have a *distinct identity*. Twenty9 speaks about the Impact Drums and Planet Drums' prototype:

> "[...] I love [the artefacts'] color, it changes from what I hear in the currently available packs that do a lot of recycling. [...] Artistically,

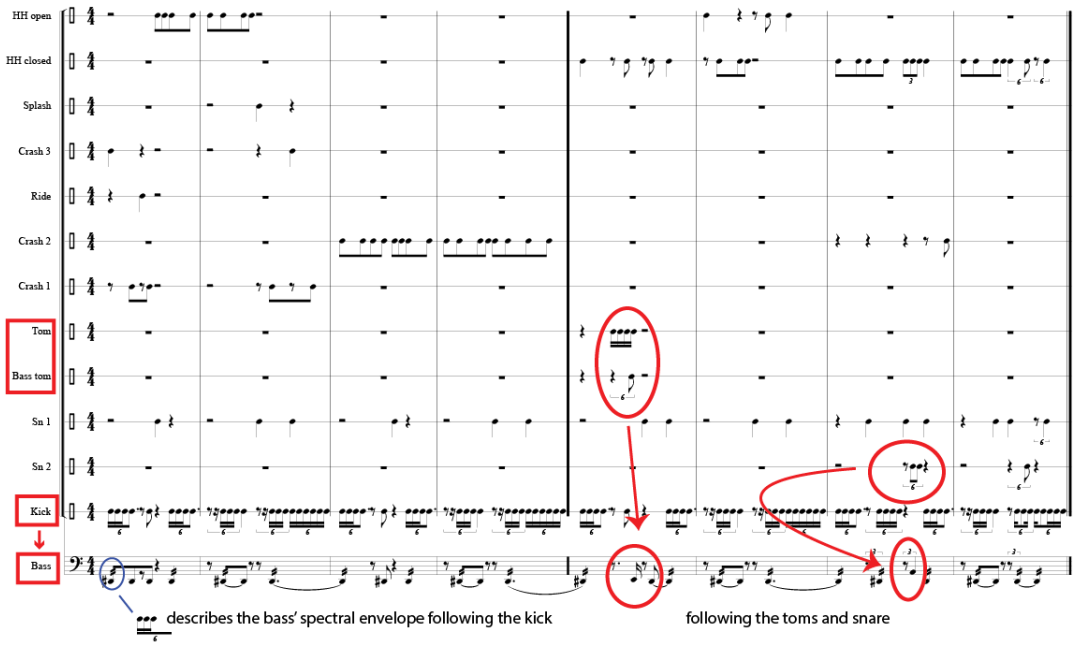


Figure 10.1 – Example of BassNet's behavior when confronted to out-of-domain input. BassNet(bottom-most track) adjusts its output's spectral envelope to the kick's attacks, and reacts to the percussion's "tonality".

> this grain is interesting [...], it is the fact of not being able to accentuate it, modify it or even play with it, which is slowing down and which limits the possibilities of sound palettes."

Uèle Lamore speaks similarly about the Notono prototype:

> "The biggest weakness of Notono at [this] moment [in development], was its extreme treatment of sound. This resulted in the creation of very "phasy", filtered, samples with a very peculiar acoustic quality. However this was absolutely perfect to represent the *Corruption of the Forest* [song title], an unnatural, evil substance slowly spreading like a disease."

Such a music process is reminiscent of the analog synth's "grain" that is so much sought after by popular music musicians.

Out-of-Domain Input

In AI-driven music production, the output of a model reflects the "personality" of the training dataset, for example, the genre (EDM, rap, rock...) or a musician's style. A natural consequence is that prototypes based on conditional input will behave unexpectedly when confronted with types of data on which they have not been trained. We call this particular type of *priming* "out-of-domain input". Figure 10.1 shows a transcription of the input and output of BassNet used with out-of-domain input. This version of BassNet was trained on complete multi-tracks of classic rock songs. However, the input to the example consists

of the audio of *death metal* solo drums. BassNet (bottom-most track) adjusts its output's spectral envelope to the kick's attacks and reacts to the percussion's "tonality," the toms, and the snare.

Uèle Lamore describes her experience with out-of-domain use of a version of BassNet that was trained on mostly 4/4 beat rock, hip-hop, and EDM multi-tracks, always including a drum section:

> "[...] none of my music on this EP is in 4/4, it's as far as you can get from pop or hip-hop and this track had zero percussion at this point. As a result of this, BassNet did not behave the way you would expect it to. However, I had the pleasant surprise to see the generations were perfect melodies that worked really well in this ambient setting."

Out-of-domain input is reminiscent of the exploratory use of the Moog synthesizer [Pinch and Trocco, 2002]. The model is considered a complex and unpredictable music generator whose output can be explored using proper triggers. Working with these tools becomes "a journey of discovery."

Out-of-Domain Output

We denote "out-of-domain output" when an artist uses the output of a tool for a different purpose than intended. For example, Donn Healy states:

> "I took a new snare pattern that DrumNet suggested and I brought it into a melodic Omnisphere sound, and I spread the notes in a way that they told a musically cohesive story [..] I really enjoyed that."

Also, as illustrated by the Uèle Lamore's quote above, some artists took outputs of BassNet and pitched it to obtain melodies instead of bass lines. Similarly, we discovered that ResonanceEQ, a tool designed to remove resonances, is usually inverted by artists to *add* resonances to audio.

10.3 Guides

Finally, we want to communicate some of the lessons we learned throughout our work in AI-based musical research. They are meant to constitute some practical guides to pushing research towards creating output that is useful in music production. AI-based musical research involves more than model design. It is strongly multi-disciplinary, comprising, among other fields, user experience, and human-computer interaction, music and sound perception, musicology, studio production, and machine learning. With this in mind, we enumerate some "learned lessons," which may guide the practical work in AI-based musical research (see Section 10.3.1). Also, we provide some suggestions for validation in AI-based musical research, which goes beyond typical model evaluation (see Section 10.3.2).

10.3.1 Lessons Learned on AI-based Musical Research

Considering musical research as the simultaneous practice of innovation in music technology and music production, the integration of AI and the focus on contemporary popular music are two novelties in this field. From past innovations

in musical research [Palombini, 1993, Pinch and Trocco, 2002] and our activity, we formulate some learned lessons as guidelines for further research in AI-driven music production tools.

Work alongside musicians. The researcher is only part of the story. A perfectly well-trained model may be irrelevant in music production. Conversely, it is not always a problem in music production if the model does not work perfectly. Some recognizable problems may even be a mark of style, as was the case with analog synthesizers. Go beyond the proof of concept to create exciting instruments.

Foster chance / serendipity. Like Schaeffer, like the Beatles, and like musicians using a Moog, create situations with rich potential: using different prototypes together or along with third-party tools, modifying models in an unorthodox way, or using models for applications they were not conceived for. In the most extreme case, AI models do not even need to be trained in order to emit musically valuable output [Steinmetz and Reiss, 2020].

Include varied music genres. Bob Moog worked with many kinds of musicians, from experimental psychedelic musicians to the traditional-sounding Simon & Garfunkel duo [Pinch and Trocco, 2002, p. 66]. On the other hand, most of Schaeffer's followers focused on the marginal *musique concrète*, while the rest of the world went on applying Schaeffer's method to many genres of music.

AI does not need to entail autonomy. In a recent interview,[4] Uèle Lamore states, "The computer wants to play everything perfectly, but the music I make isn't perfect. The human will always add something of their own". When developing AI-driven music production tools, one is tempted to think *fully autonomous musical agents*. However, AI can be used to drive novel instruments, which are still "played" by humans. Ultimately, humans want to remain under control.

Develop better metrics. In the same interview, Lamore asks: "Rather than trying to replace human input, why don't we push [the AI] to do something that is new, something different? That would be far more interesting!" When only pushing the traditional self-supervised sequence learning (and probabilistic sampling) approach using precision and log-likelihood, we will always need to add (uncontrollable) disturbances to AI models in order to provoke results beyond the data distribution, and even the most sophisticated models [Dhariwal et al., 2020] will continue suffering from missing long-term structure. What does it take for an AI to emit original (and appealing) musical material, notably different from that of the training data? We need to investigate additional loss functions that better reflect human music perception. For example, in [Lattner, 2019, pp. 107-109], information-theoretic metrics are discussed that have perceptual relevance and could be used complementary to traditional loss functions in music generation systems.

Understand contemporary popular music. Popular music uses a different language than Western classical music. This language is not well-documented. Learning from scores is only partially relevant to contemporary popular music. Beyond the music itself, get acquainted with the workflows. Use standard plug-in formats. Note that using audio input to AI tools is often more useful for the artist: it provides a richer, more expressive basis for computation and can potentially

[4] www.musicradar.com/news/uele-lamore

produce unexpected results.

Produce music. When Schaeffer worked on musical research in the '50s, his director Émile Biasini had one word of advice for him: "produce, and you will be esteemed" [Jeanneney, 1999]. It is perhaps the most important advice of all. AI-driven music production has no point if it is not used to actually produce music.

Set practical validation methods. As discussed in the next section (see Section 10.3.2), traditional model validation is only of limited practical use and almost certainly not sufficient to validate musical research. Thus, identify more relevant goals. How well are the prototypes integrated into workflows? Do the musicians find the prototypes helpful? Is it possible to determine whether prototype inclusion benefits the final result? Are the musicians able to release music involving AI technology?

10.3.2 On the Validation of AI-driven Music Technology

In machine learning, validation is usually performed by measuring how well a model can replicate the statistics of some dataset. However, musical research involves music production, and music production involves creativity. From this perspective, a model that reliably produces outputs characteristic of a genre may be only of limited value to artists who aim for stylistically unexpected output. A common way to obtain unexpected output from adequately trained models is by introducing some form of disturbance, such as increasing the *temperature* parameter in probabilistic models, using style-transfer approaches [Gatys et al., 2015], or in the case of conditional models, confronting the model with out-of-domain input (see Section 10.2.4).

Nevertheless, the problem remains that when creativity is involved, validation is not as straightforward as how well the model can replicate properties of existing content. There is no definitive answer to this problem. However, let us suggest a few possible directions for validation in musical research activities.

Workflow integration. Validation may take into account if a tool finds its place in a production workflow. For that, a tool needs to be useful and should not interrupt the workflow the artist is accustomed to (for example, artists are often reluctant to switch from their DAW to an external standalone application). Figure 10.2 shows an example of a successful workflow, in which Luc Leroy and Yann Macé use our AI-driven prototypes in conjunction with mainstream technology.

Facilitation of production. Does the prototype simplify a difficult or time-consuming task? For instance, Yann Macé appreciates latent space navigation in DrumGAN, as it provides much quicker results than spending hours browsing a drum sample library. An interesting example is artist Twenty9 appreciating FM Pro's ability to generate melodies, as he prefers to focus on different aspects in music production.

Enhanced creativity. Does the prototype stimulate the artist's creativity? Does it provide a good trade-off between quality and novelty (i.e., it does not frustrate the artist due to too many useless outputs or cumbersome usability)? For instance, Twenty9 and Uèle Lamore repeatedly mention that BassNet, LeadNet, and DrumNet provide solutions they would have never considered, but they ended up using.

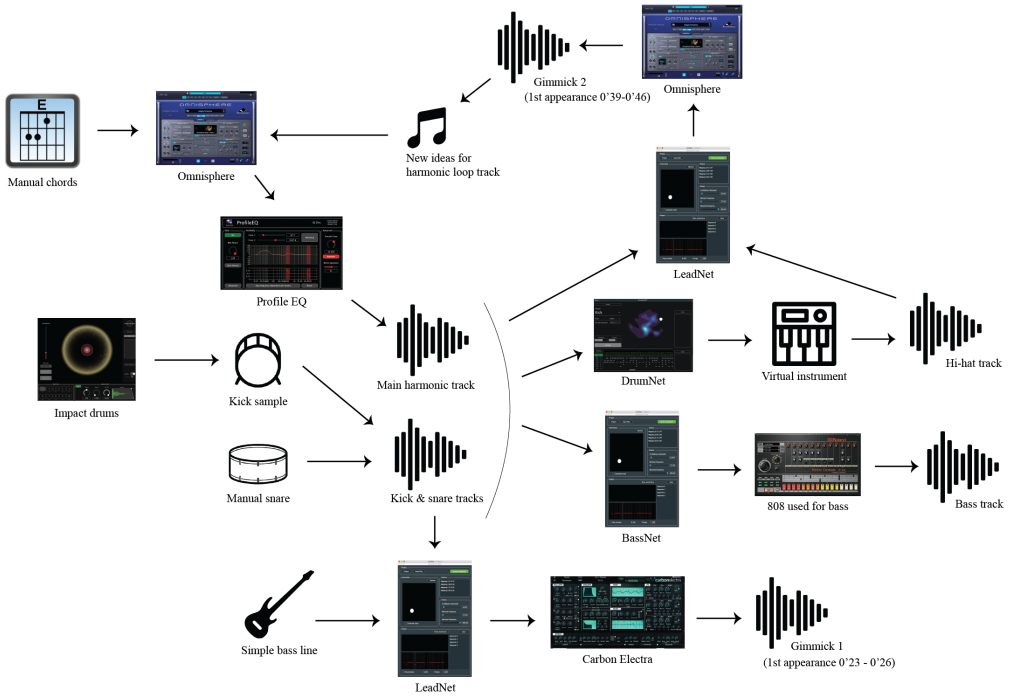


Figure 10.2 – Example of the integration of AI-based prototypes in a popular music production workflow

Identifiable results. Did the technology bring recognizable elements to the music? For instance, Twenty9 enjoys the grain of our GAN-based drum generators, and Yann Macé appreciates the characteristic style of DrumNet's hi-hat tracks.

Published content. The commercial viability of music content created using AI technology may also be a criterion according to which the technology can be evaluated. Three examples: the release, by Twenty9, of a drumkit designed using Impact Drums, Planet Drums and DrumGAN (December 2020)[5]; the release, by Uèle Lamore and her label, of an EP made in collaboration with our lab's technology (March 2021); Yann Macé and Luc Leroy using Impact Drums and DrumGAN in the music track for a worldwide Azzaro advertisement campaign (April 2021).

A spectacular example of successful validation is Schaeffer's work. Most contemporary popular music uses sampling, looping, and pitch-shifting. Schaeffer's musical research output can be identified in the music from an entire era.

10.4 Conclusions

The use of AI-based tools for music production is currently in its infancy. For the success of this endeavor, much work remains to be done. In this chapter, we reviewed feedback and observations from our collaborations with professional musicians, in which the goal is to discover how AI-based technology can enrich the musician's creative workflow. Also, we described some lessons learned from

[5]The AI Drum-Kit

this joint effort. In particular, for AI-based music innovation to work in practice, we believe it is essential to make user interaction an integral part of model design and development from the very start. Also, the characteristics that are relevant for modeling vary widely from one musical style to another. This implies that the design of AI-based music technology cannot be style-agnostic. Lastly, we discussed several ways of measuring the success of AI-based innovation in music.

Besides any technical limitations of AI, a present challenge for AI-based music innovation is a discrepancy between the expectations and perspectives of musicians on the one hand and engineers on the other. We believe that over time these discrepancies will diminish and converge on the commodification of AI-based technology. It is conceivable that in the future, musicians will have an arsenal of AI techniques at their disposal, training and tweaking machine learning models as part of their creative process as easily as they use any other audio software today.

Chapter 11

General Conclusion

The dream of machines recreating and responding to complex human behavior promises to radically transform how we produce music. As seen in Chapter 3, the advent of Deep Learning into the realm of music technology innovation may pave the way towards such a paradigm shift. This dissertation has studied Generative Adversarial Networks (GANs), a specific deep learning technique, for the controllable and intuitive audio synthesis of musical sounds.

The use of GANs —and Deep Learning in general— for musical audio generation is still in its infancy. Among the existing challenges in modeling musical audio [Dieleman et al., 2018], one that is fundamental is that of choosing the best audio representation. We have seen in Chapter 2 that this may depend on the specific type of acoustic source to be modeled (e.g., tonal, percussive) as well as the neural network architecture (e.g., CNN, RNN). The choice of representation may greatly influence how efficiently the neural network learns from the audio data and the generation time. Therefore, in our first work, presented in Chapter 5, we compared various common audio representations, including the raw audio waveform and several time-frequency representations, on the task of audio synthesis of tonal instrument sounds [Nistal et al., 2021c]. This work provided some insights on the performance of these representations on a specific benchmark convolutional Progressive Growing GAN [Karras et al., 2017]. Results showed that the magnitude and Instantaneous Frequency (IF) of the STFT and the complex-valued STFT obtained the best quantitative metrics amongst all the compared representations.

Another essential aspect that we tackled in this thesis is that of conditioning the GAN in order to learn controls over the generation process. We explored various sources of conditional information for different means of control. Following our first work, in Chapter 6, we presented DrumGAN [Nistal et al., 2020], an adversarial synthesizer of percussive sounds that can be controlled based on perceptual features describing timbral properties (e.g., *boominess, roughness, hardness*). DrumGAN operates on complex-valued STFT audio data, which we observed to perform better than other representations on percussive sounds. As a result of this work, we scaled DrumGAN and built a commercially viable plug-in compatible with any Digital Audio Workstation (DAW) that generates high-definition drum sounds (i.e., 44.1 kHz sample-rate). Additionally, we learned an encoder that enabled the re-synthesis of preexisting drum samples to generate variations and, also, we added continuous control over the instrument classes (kick, snare,

and cymbals). Further described in Chapter 10, the success of DrumGAN was embodied in the release of musical content by professional musicians affiliated with Sony ATV.

While DrumGAN demonstrated that given some preexisting conditional information, we could induce in the GAN some control over high-level features of sound, in most cases, we do not have access to such information, either because datasets lack the desired annotations or because we simply do not know how to extract a specific feature. Our third experiment, presented in Chapter 7, shows that a pre-trained discriminative teacher model can be used to generate such labels. This work explored the use of Knowledge Distillation (KD) [Hinton et al., 2015], a framework for transferring knowledge from a teacher to a student neural network, as a means to learn semantically intuitive controls in a GAN-driven synthesizer of tonal sounds [Nistal et al., 2021b]. The teacher model [Kong et al., 2020b], pre-trained on the Audio Set data [Gemmeke et al., 2017], was used to generate *soft* labels on the NSynth [Engel et al., 2017] dataset for 128 sound event categories (e.g., sonar, singing-bowl, mantra, bicycle-bell, etc). Then, a GAN synthesizer was trained using such soft labels as conditional information. An interesting outcome of KD is that the student model can learn abstract representations of classes that are not explicitly represented in the training data but are somehow sparsely encoded on the aggregate [Hinton et al., 2015]. For example, the NSynth dataset does not contain any sonar sound, yet, a sonar sound may be extrapolated from other sounds in the dataset. Learning such unrepresented classes is possible thanks to the additional information that exists in the relative probabilities of the soft labels —compared to hard, one-hot labels. Such additional information was coined Dark Knowledge [Hinton et al., 2014]; hence we called our model DarkGAN. Results confirmed that DarkGAN could learn some degree of control over attributes not directly represented in the training data.

Most initial works on audio synthesis with GANs inherit the architecture from the computer vision literature, limiting the generation to fixed-duration audio in analogy to the fixed-sized image data [Donahue et al., 2019, Engel et al., 2019]. While this may be a natural choice in images, we are generally interested in synthesizing audio of any duration when modeling sound, particularly for musical purposes. Along this line, in Chapter 8, we proposed a framework for conditioning a GAN synthesizer on a sequence of discrete features capturing step-wise time-dependent information, as well as on static features (a random noise vector and the pitch class) that ensured the global consistency of the generated sound [Nistal et al., 2021a]. Such sequential features were learned using a self-supervised learning technique called Vector-Quantized Contrastive Predictive Coding (VQCPC) [Hadjeres and Crestel, 2020], hence, we called our model VQCPC-GAN. Compared to other forms of unsupervised representation learning, a valuable characteristic of CPC is that one can choose to some extent what information is captured in the learned codes. This is done by carefully designing a negative sampling strategy for the contrastive objective. In VQCPC-GAN, we designed the negative sampling in an intra-sequence fashion, forcing the learned codes to neglect any global aspects of the data (e.g., the pitch and instrument class). While baselines trained following the fixed-duration scheme scored best, results showed that VQCPC-GAN achieved comparable performance even when

generating variable-length audio. Also, we observed that the codes affected only the envelope of the generated audio and, indeed, did not exert any control over global aspects.

As we saw in Chapter 3, generative models can be conditioned on preexisting audio content, generally, as means to provide dense information to the network for, e.g., spectrogram inversion [Kumar et al., 2019] or audio enhancement [Michelsanti and Tan, 2017]. The vision of this thesis is to, one day, devise tools that can respond to rather complex musical dependencies, for example, in the form of some preexisting sparse, musical audio context, e.g., the generation of a bass-line given some recorded drums. Learning such intricate musical relationships between conditional-generated music audio pairs is complicated and would require large amounts of data. With this in mind, in Chapter 9, we designed a pretext audio enhancement task tailored at learning such kinds of dependencies in the future. Concretely, we proposed a GAN to restore heavily compressed MP3 music to its uncompressed, high-quality form [Lattner and Nistal, 2021]. Results showed that the GAN could improve the quality of the audio signals over the MP3 versions for high compression rates (i.e., 16 kbps, 32 kbps). Also, we noted that a stochastic generator could generate outputs closer to the original signals than those generated by a deterministic generator.

As a direct consequence of the application of Artificial Intelligence to musical production tasks, it is crucial to assess these novel technologies by bringing the artist in the loop. We believe that AI-driven music research should be carried out as an interdisciplinary effort involving researchers and artists to effectively design tools that can help enhance the music production experience by speeding workflows and inspiring the creative process. This perspective is plotted in our final work, presented in Chapter 10, where we reported on collaborations with professional artists, looking at how various in-house AI tools were used in practice to produce music. We identified usage patterns, issues and challenges that arose from the practical use of these tools. Based on this, some recommendations and validation criteria were formulated to develop AI technology for contemporary popular music.

11.1 Future Work

Following, we enumerate some interesting directions for future work:

- **Modelling new acoustic sources**. An obvious direction for extending our research on adversarial audio synthesis is to broaden the variety of acoustic sources in the training dataset. In this work, we have mainly focused on modeling tonal instruments (see Chapters 5, 7, and 8) and percussive sounds (see Chapter 6). We also used in Chapter 9 music mixtures for restoring heavily compressed MP3 music. An interesting endeavour for future research would be to consider other types of sound sources such as environmental sounds, sound effects, and other natural sound sources. The main challenge is the availability of large datasets with such types of sounds as well as the design of efficient architectures with enough computational capacity to model all of these sound sources.

- **Modeling polyphonic sources**. In this project we have mainly focused

on modeling one-shot, monophonic sound sources. Another way to increase the amount of training data and learn richer dependencies between instrument sources would be to train the model on multiple instruments playing simultaneously (polyphonic and multitimbral). Adequate architectures and data representations need to be found which allow for an efficient and cost-effective implementation of such methods. In order to achieve that goal, it is probably necessary for the model to attend to the different sources separately (or perform any implicit or explicit way of source separation).

- **Extending intuitive controls on neural audio synthesizers**. In this thesis, we studied various methods for learning controls in a neural network. As part of the prototype development for the continuation of this project, we will further investigate novel sources of conditional information (Attack, Decay, Sustain, and Release curves, MIDI, MEL, Principal Components, Captions), as well as unsupervised learning of factors of variation [Peebles et al., 2020, Karras et al., 2018]. Another possible direction for research is to design distance measures or trajectory strategies for traversing latent spaces in a meaningful way or reducing their dimensionality for visualization and navigation purposes.

- **Investigating other audio representations**. While we have compared some important and common audio representations, many other forms of structuring audio information exist that could help train lighter models that can learn more efficiently from the audio data (e.g., wavelet transforms [Luo et al., 2017], Differentiable Digital Signal Processing [Engel et al., 2020]).

- **Reducing model size**. Reducing models' size and computational requirements becomes crucial when deploying DL models on edge hardware or in memory-limited settings (e.g., personal computer, a microprocessor). This project aimed to develop DL-driven audio synthesis tools that musicians can use as a VST plugin on their personal computers. Hence, a requirement for our models is that they can run in —or close to— real-time in a CPU. Various methods for model compression exist that are worthy of study: distillation [Hinton et al., 2015], pruning [Zhu and Gupta, 2018], lottery ticket hypothesis [Kalibhat et al., 2021], quantization [Gholami et al., 2021], and more.

- **Adversarial Autoencoders**. As opposed to plain GANs [Goodfellow et al., 2014], we saw that VAEs [Kingma and Welling, 2014] feature the possibility to encode data back into points in the latent space using an encoder. However, VAEs have the added difficulty of imposing the specific dataset through a reconstruction loss (i.e., explicit density estimation) and, due to the variational encoding, they are known to produce blurry outputs. Following existing research, one interesting avenue for research is Generative Adversarial Autoencoders [Pidhorskyi et al., 2020]. This framework was shown in computer vision to join the data quality and sharpness of the GANs with the autoencoding capabilities of VAEs.

- **Hierarchical VQCPC**: VQCPC-GAN [Nistal et al., 2021a] proposed to use a sequence of tokens extracted from real audio data as conditional input

to a GAN architecture. In this work, a token corresponded to a cluster of time-frequency frames (i.e., one token per frame). A promising research line, inspired by previous work [Dhariwal et al., 2020], is to use a hierarchy of VQCPC encoders, where tokens produced by higher VQCPC encoders in the hierarchy account for longer-term segments of audio (i.e., more than one frame) in order to learn features capturing structure at broader scales. By conditioning the GAN on both fine-grain and long-term tokens of the sequential data, we hope to control the structure of the generated data at various time scales.

Appendices

A. Figure Acknowledgements

- Computer Music Icon from pngrepo.com
- Musicians Icon from Free Icons Library
- Hand icon by technology from shareicon.net
- Icon synth made by Hum from NounProject.com
- Sound Waves icon by Alice Noir from NounProject.com
- Hearing by Marek Polakovic from NounProject.com
- Idea Icon by Memed Nurrohmad from NounProject.com
- Extrasensory Perception by Andrew Forrester from NounProject.com
- Rock N Roll by Daouna Jeong from NounProject.com
- Fist by Cesar Reynoso from NounProject.com
- Note by Aleksandr Vector from NounProject.com
- Heart by Unicons Font from NounProject.com
- Customer Satisfaction by Luis Prado from NounProject.com
- Faders by Ashley van Dyck from NounProject.com

B. Attribute Correlation Coefficient Table

Attribute	T=1	T=1.5	T=2	T=3	T=5
Accordion	0.10	0.25	0.31	**0.32**	0.10
Acoustic guitar	0.20	0.36	**0.39**	0.23	0.10
Air horn, truck horn	0.16	0.26	**0.31**	0.18	0.13
Ambulance	-	-	-	-	0.03
Animal	0.00	-0.01	**0.05**	0.03	0.00
Bagpipes	-	-	-	**0.17**	0.11
Banjo	0.02	0.21	**0.23**	0.10	0.02
Bass drum	0.02	**0.14**	**0.14**	0.09	0.02
Bass guitar	0.30	0.38	**0.46**	0.38	0.19
Battle cry	-	-	-	-	0.03
Bee, wasp, etc.	-	-	-	0.00	0.01
Beep, bleep	0.11	0.22	**0.26**	-	-
Bicycle bell	0.11	0.16	0.08	**0.23**	0.01
Bird	**0.06**	0.00	0.01	0.04	-
Bluegrass	-	0.25	-	-	-
Blues	0.21	-	-	-	-
Boat, Water vehicle	0.06	0.09	**0.19**	0.06	0.08
Boing	0.06	0.13	**0.16**	0.08	0.01
Bowed string instrument	0.20	**0.30**	0.22	0.23	0.04
Brass Instrument	0.28	**0.49**	0.38	0.26	0.00
Busy signal	0.02	0.05	0.04	**0.06**	-
Buzzer	0.02	**0.08**	**0.08**	-	-
Car	0.02	0.01	**0.10**	0.02	-
Cat	-0.01	-0.01	-0.01	-0.01	0.00
Cattle, bovinae	0.05	0.07	0.09	0.10	**0.12**
Caw	-	-	-0.06	0.00	-0.03
Cello	0.24	**0.29**	0.26	0.17	0.00
Change ringing (campanology)	-	-	-	**0.08**	0.02
Chicken, rooster	0.00	-0.06	-0.02	-0.01	-0.01
Chime	0.15	0.33	**0.39**	0.31	0.03
Chirp tone	0.18	**0.28**	0.25	-	-
Choir	0.00	**0.18**	0.16	0.08	0.05
Chorus effect	0.11	0.19	0.16	**0.24**	0.12
Church bell	0.07	0.07	**0.10**	0.08	0.08
Civil defense siren	0.10	0.16	**0.23**	0.09	0.06
Clang	0.13	0.17	**0.22**	-	-
Clarinet	0.12	0.29	0.37	**0.39**	-
Coo	-	-	-	**0.09**	0.01
Cowbell	0.01	0.13	**0.21**	0.15	0.10
Cricket	-	-	-	-	-0.02
Croak	-	-	**0.10**	0.08	0.03
Crowd	-0.01	0.00	0.01	-0.01	0.03
Crowing, cock-a-doodle-doo	-	-	0.01	-0.01	-0.01
Cymbal	-	-	-	-	-0.02
Dial tone	0.12	0.22	**0.24**	0.20	0.03
Didgeridoo	0.06	0.16	**0.21**	0.20	0.08
Ding	0.17	0.25	**0.29**	-	-
Ding-dong	0.08	-	-	-	-
Distortion	0.11	0.15	0.20	**0.25**	0.14
Dog	-0.01	-0.01	0.00	0.01	0.01
Domestic animals, pets	-0.01	-0.02	0.02	0.00	0.00

Table 1 – A few examples of attribute correlation coefficients $\rho^i(\hat{\alpha}, \alpha)$ (see Sec. 7.3).

Attribute	T=1	T=1.5	T=2	T=3	T=5
Doorbell	0.09	0.24	**0.26**	**0.26**	-
Double bass	0.24	**0.30**	**0.30**	0.22	0.01
Drum	0.05	0.21	**0.24**	0.12	0.01
Drum kit	0.03	**0.18**	0.17	0.08	0.02
Drum machine	0.12	**0.26**	-	-	-
Echo	-0.01	0.03	0.01	-	-
Effects unit	0.10	0.15	0.18	**0.24**	0.12
Electric guitar	0.10	0.23	**0.28**	0.26	0.08
Electric piano	0.16	0.15	**0.25**	0.19	-
Electronic organ	-0.03	-	-	-	-
Electronic tuner	0.35	0.44	**0.50**	0.29	0.13
Electronica	0.10	-	-	-	-
Emergency vehicle	0.05	**0.09**	0.08	0.04	0.04
Engine	0.02	**0.09**	-	-	-
Fart	-	-	-	-0.03	-0.02
Fire alarm	0.01	**0.08**	0.06	0.05	0.02
Fire engine, fire truck (siren)	-	0.14	**0.15**	0.04	0.01
Fireworks	-	-	0.13	**0.14**	0.01
Flute	0.09	0.20	**0.25**	-	-
Fly, housefly	0.00	-0.01	-0.01	-0.02	0.00
Foghorn	**0.08**	0.06	0.06	0.03	-
Fowl	-0.01	-0.07	-0.02	-0.02	-0.01
French horn	0.12	**0.23**	0.20	0.05	-0.02
Frog	0.00	0.03	**0.07**	0.06	-0.03
Glockenspiel	0.02	0.12	**0.22**	-	-
Gobble	-	-	-	-0.04	-0.01
Gong	0.04	0.15	**0.21**	0.17	0.04
Guitar	0.28	0.37	**0.42**	0.34	0.13
Gunshot, gunfire	-	-	-	-0.02	0.01
Hair dryer	-	-	-	-	0.01
Hammond organ	-0.01	0.03	0.05	**0.10**	0.04
Harmonic	0.16	0.20	-	-	-
Harmonica	0.10	**0.27**	0.22	0.18	0.05
Harp	0.11	0.37	**0.41**	0.17	0.06
Harpsichord	0.04	0.09	**0.15**	0.13	0.01
Heart sounds, heartbeat	-	0.09	0.03	**0.10**	0.01
Honk	-	-	-	-	0.00
Hoot	0.03	0.02	-0.01	0.00	-0.01
Howl	0.02	0.04	0.04	**0.05**	0.04
Hum	0.05	**0.10**	-	-	-
Humming	0.01	0.03	-	-	-
Insect	0.00	-0.02	-0.02	-0.02	-0.01
Inside, small room	0.24	**0.30**	**0.30**	0.19	-
Jingle bell	0.09	0.20	**0.25**	0.10	0.08
Keyboard (musical)	0.15	0.10	0.19	**0.16**	0.04
Livestock, farm animals, working animals	0.03	0.04	0.03	0.05	**0.07**
Lullaby	0.01	0.09	**0.10**	-	-
Machine gun	-	-	-	-	0.01
Marimba, xylophone	0.02	0.10	**0.20**	0.17	-
Meow	0.01	**0.06**	0.02	0.01	0.02
Moo	0.05	0.07	0.09	0.08	**0.10**
Mosquito	-	-	-	-0.03	0.02
Neigh, whinny	-	-	-	-0.01	-0.02
Opera	-	-	-	-	0.04
Orchestra	0.30	**0.53**	0.47	-	-
Organ	-0.02	0.02	0.03	**0.07**	0.01
Outside, urban or manmade	0.12	-	-	-	-
Owl	0.04	0.04	0.00	0.01	-0.01

Table 2 – Attribute correlation coefficients $\rho^i(\hat{\alpha}, \alpha)$ (see Sec. 7.3).

Attribute	T=1	T=1.5	T=2	T=3	T=5
Percussion	0.04	0.19	**0.30**	0.14	0.08
Piano	0.16	0.10	**0.19**	0.16	0.03
Pigeon, dove	-	-	-	**0.09**	0.02
Ping	0.10	0.22	**0.26**	-	-
Pizzicato	0.05	0.29	**0.31**	0.15	0.04
Plop	0.03	0.10	**0.13**	0.10	0.01
Plucked string	0.27	0.37	**0.42**	0.32	0.11
Police car (siren)	0.02	**0.05**	0.04	0.01	0.00
Purr	-	-	-0.02	**0.09**	0.00
Rail transport	0.03	0.03	**0.10**	0.06	0.04
Railroad car, train wagon	0.04	0.02	**0.08**	0.04	0.03
Rain	-	-	-	-	0.02
Reverberation	0.09	0.15	**0.17**	-	-
Ringtone	0.01	0.02	0.05	**0.06**	-
Rub	-	-0.01	0.00	0.00	0.03
Sampler	0.15	0.17	**0.21**	-	-
Saxophone	0.25	**0.41**	**0.41**	**0.41**	0.03
Sanding	-	-	-	-	0.00
Scratching (performance technique)	-	-	-	-	0.10
Shofar	0.04	0.10	0.09	**0.11**	0.02
Sine wave	0.28	**0.32**	0.27	0.17	0.05
Singing	0.02	**0.18**	0.14	0.07	-
Singing bowl	0.08	0.20	**0.24**	0.21	0.03
Siren	0.13	0.19	**0.24**	0.10	0.08
Sizzle	-	-	-	0.02	-
Smoke detector, smoke alarm	-	0.05	0.06	**0.09**	0.00
Snare drum	0.01	0.10	**0.11**	0.06	0.01
Sonar	-	0.06	**0.13**	0.10	0.01
Speech	-0.04	-0.10	-0.07	-0.05	0.01
Static	0.06	0.08	0.08	**0.18**	-
Steam whistle	-	-	-	0.06	0.03
Steel guitar, slide guitar	0.06	0.20	**0.23**	-	-
Steelpan	-	-	-	**0.07**	0.04
Stomach rumble	-	-	-	-	0.05
Strum	0.12	0.28	**0.30**	0.21	
Synthesizer	**0.09**	0.05	0.08	0.08	
Tapping (guitar technique)	0.13	0.27	**0.32**	0.23	-
Telephone bell ringing	-	-	-	-	0.06
Theremin	0.04	0.06	**0.10**	0.02	0.00
Thunder	-	-	-0.06	0.03	0.04
Thunderstorm	-	-	-	-	0.04
Tick-tock	0.04	0.09	**0.14**	-	-
Timpani	0.04	0.15	**0.32**	0.12	0.09
Toot	0.17	0.20	**0.25**	0.13	0.08
Train	0.03	0.04	**0.11**	0.06	0.05
Train horn	-	-	-	0.07	0.07
Train wheels squealing	-	-	-	-	-0.05
Trombone	0.18	**0.41**	0.29	0.16	0.00
Trumpet	0.16	**0.46**	0.36	0.25	0.00
Tubular bells	0.05	**0.17**	-	-	-
Tuning fork	0.22	0.29	**0.35**	0.29	0.10
Turkey	-	-	-	-0.05	-0.02
Ukulele	0.09	0.27	**0.31**	0.15	0.05
Vehicle	0.08	0.10	**0.19**	0.05	0.04
Vehicle horn, car horn, honking	0.18	**0.26**	0.24	0.18	-
Violin, fiddle	0.19	**0.26**	0.22	0.24	-
Water	0.04	-0.01	0.03	**0.05**	-0.01
Whistling	-	-	-	-	0.01
Wind chime	-	-	**0.30**	0.26	0.04
Wind instrument	0.21	0.36	**0.40**	0.39	0.10
Wood block	0.03	0.15	**0.27**	0.10	0.05
Zither	0.03	0.18	**0.19**	0.07	-0.01

Table 3 – A few examples of attribute correlation coefficients $\rho^i(\hat{\alpha}, \alpha)$ (see Sec. 7.3).

Bibliography

Y. Ai, H.-C. Wu, and Z.-H. Ling. SampleRNN-Based Neural Vocoder for Statistical Parametric Speech Synthesis. page 91, 2018. URL `http://mirlab.org/conference_papers/International_Conference/ICASSP2018/pdfs/0005659.pdf`.

R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton. Large scale distributed neural network training through online distillation. In *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, May 2018.

C. Aouameur, P. Esling, and G. Hadjeres. Neural drum machine: An interactive system for real-time synthesis of drum sounds. In *Proc. of the 10th International Conference on Computational Creativity, ICCC*, Charlotte, North Carolina, USA, June 2019.

M. Aramaki, R. Kronland-Martinet, T. Voinier, and S. Ystad. A percussive sound synthesizer based on physical and perceptual attributes. *Comput. Music. J.*, 30(2):32–41, 2006. doi: 10.1162/comj.2006.30.2.32. URL `https://doi.org/10.1162/comj.2006.30.2.32`.

M. Aramaki, M. Besson, R. Kronland-Martinet, and S. Ystad. Controlling the Perceived Material in an Impact Sound Synthesizer. *IEEE Trans. Speech Audio Process.*, 19(2):301–314, 2011a. doi: 10.1109/TASL.2010.2047755. URL `https://doi.org/10.1109/TASL.2010.2047755`.

M. Aramaki, R. Kronland-Martinet, and S. Ystad. Perceptual control of environmental sound synthesis. In *Speech, Sound and Music Processing: Embracing Research in India - 8th International Symposium, CMMR, 20th International Symposium, FRSM*, volume 7172 of *Lecture Notes in Computer Science*, pages 172–186, Bhubaneswar, India, March 2011b. Springer.

S. Ö. Arik, M. Chrzanowski, A. Coates, G. F. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Y. Ng, J. Raiman, S. Sengupta, and M. Shoeybi. Deep Voice: Real-time Neural Text-to-Speech. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 195–204, Sydney, NSW, Australia, August 2017.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017.

T. Asami, R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono. Domain adaptation of DNN acoustic models using knowledge distillation. In *IEEE*

International Conference on Acoustics, Speech and Signal Processing, ICASSP, pages 5185–5189, New Orleans, LA, USA, March 2017. IEEE. doi: 10.1109/ICASSP.2017.7953145.

Y. Aytar, C. Vondrick, and A. Torralba. SoundNet: Learning Sound Representations from Unlabeled Video. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 892–900, Barcelona, Spain, December 2016.

J. Ba and R. Caruana. Do Deep Nets Really Need to be Deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Annual Conference on Neural Information Processing Systems*, pages 2654–2662, Montreal, Quebec, Canada, December 2014.

A. Baevski, S. Schneider, and M. Auli. vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations. In *ICLR*, Addis Ababa, Ethiopia, Apr. 2020.

D. Bansal, B. Raj, and P. Smaragdis. Bandwidth expansion of narrowband speech using non-negative matrix factorization. In *9th European Conference on Speech Communication and Technology, INTERSPEECH*, pages 1505–1508, Lisbon, Portugal, September 2005. ISCA. URL `http://www.isca-speech.org/archive/interspeech_2005/i05_1505.html`.

A. Barahona-Ríos and T. Collins. SpecSinGAN: Sound Effect Variation Synthesis Using Single-Image GANs. *CoRR*, abs/2110.07311, 2021.

S. T. Barratt and R. Sharma. A Note on the Inception Score. *CoRR*, abs/1801.01973, 2018. URL `http://arxiv.org/abs/1801.01973`.

S. Barry and Y. Kim. "Style" Transfer for Musical Audio Using Multiple Time-Frequency Representations, 2018. URL `https://openreview.net/forum?id=BybQ7zWCb`.

T. Bazin, G. Hadjeres, P. Esling, and M. Malt. Spectrogram Inpainting for Interactive Generation of Instrument Sounds. In *Joint Conference on AI Music Creativity*, 2020.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems NeurIPS*.

Y. Bengio, A. C. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8), 2013.

M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD gans. In *ICLR*, Vancouver, BC, Canada, Apr. 2018.

M. Binkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan. High Fidelity Speech Synthesis with Adversarial Networks. In *8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia, April 2020.

A. Biswas and D. Jia. Audio Codec Enhancement with Generative Adversarial Networks. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 356–360, Barcelona, Spain, May 2020. IEEE. doi: 10.1109/ICASSP40776.2020.9053113.

A. Bitton, P. Esling, A. Caillon, and M. Fouilleul. Assisted Sound Sample Generation with Musical Conditioning in Adversarial Auto-Encoders. *CoRR*, abs/1904.06215, 2019.

A. Bitton, P. Esling, and T. Harada. Neural Granular Sound Synthesis. *CoRR*, abs/2008.01393, 2020.

M. Blaauw and J. Bonada. Modeling and Transforming Speech Using Variational Autoencoders. In *17th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 1770–1774, San Francisco, CA, USA, September 2016. ISCA.

M. Blaauw and J. Bonada. A neural parametric singing synthesizer. In *18th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 4001–4005, Stockholm, Sweden, August 2017.

B. Boashash. Estimating and interpreting the instantaneous frequency of a signal. II. Algorithms and applications. *Proc. of the IEEE*, 80(4):550–568, Apr. 1992. ISSN 1558-2256. doi: 10.1109/5.135378.

S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2):113–120, 1979. doi: 10.1109/TASSP.1979.1163209.

S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *CoRR*, abs/2103.04922, 2021.

K. Brandenburg. MP3 and AAC explained. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.

K. Brandenburg and G. Stoll. Iso/mpeg-1 audio: A generic standard for coding of high-quality digital audio. *Journal of the Audio Engineering Society*, 42(10): 780–792, 1994.

J. Briot, G. Hadjeres, and F. Pachet. Deep Learning Techniques for Music Generation - A Survey. *CoRR*, abs/1709.01620, 2017. URL `http://arxiv.org/abs/1709.01620`.

A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019. OpenReview.net.

J. C. Brown. Calculation of a constant-Q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991. ISSN 0001-4966. doi: 10.1121/1.400476.

G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer. MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR, 2018*, pages 747–754, Paris, France, September 2018a.

G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao. Symbolic Music Genre Transfer with CycleGAN. In *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI*, pages 786–793, Volos, Greece, November 2018b.

C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541, Philadelphia, PA, USA, August 2006. ACM.

H. Caesar, J. R. R. Uijlings, and V. Ferrari. COCO-Stuff: Thing and Stuff Classes in Context. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1209–1218, Salt Lake City, UT, USA, June 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00132.

A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe. EyesWeb: Toward Gesture and Affect Recognition in Interactive Dance and Music Systems. *Comput. Music. J.*, 24(1):57–69, 2000.

H. Caracalla and A. Roebel. Sound Texture Synthesis Using RI Spectrograms. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2020*, pages 416–420, Barcelona, Spain, May 2020. IEEE.

W. Chan, N. R. Ke, and I. Lane. Transferring knowledge from a RNN to a DNN. In *16th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 3264–3268, Dresden, Germany, September 2015. ISCA.

F. Chen, R. Huang, C. Cui, Y. Ren, J. Liu, Z. Zhao, N. Yuan, and B. Huai. SingGAN: Generative Adversarial Network For High-Fidelity Singing Voice Generation. 2021.

J. Chen, Y. Wang, S. Yoho, D. Wang, and E. Healy. Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises. *The Journal of the Acoustical Society of America*, 139:2604–2612, 05 2016a. doi: 10.1121/1.4948445.

L. Chen, S. Srivastava, Z. Duan, and C. Xu. Deep Cross-Modal Audio-Visual Generation. In *Proceedings of the on Thematic Workshops of ACM Multimedia*, pages 349–357, Mountain View, CA, USA, October 2017.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 2172–2180, Barcelona, Spain, December 2016b.

R. Child, S. Gray, A. Radford, and I. Sutskever. Generating Long Sequences with Sparse Transformers. *CoRR*, abs/1904.10509, 2019.

Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 8789–8797, Salt Lake City, UT, USA, June 2018.

J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *CoRR*, abs/1901.08810, 2019. URL `http://arxiv.org/abs/1901.08810`.

J. M. Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7):526–534, September 1973.

O. Cífka, A. Ozerov, U. Simsekli, and G. Richard. Self-Supervised VQ-VAE For One-Shot Music Style Transfer. *CoRR*, abs/2102.05749, 2021.

Clark E. et al. Creative writing with a machine in the loop: Case studies on slogans and stories. In *IUI*, pages 329–340. ACM, March 2018. doi: 10.1145/3172944.3172983. URL `https://doi.org/10.1145/3172944.3172983`.

M. Comunità, H. Phan, and J. D. Reiss. Neural synthesis of footsteps sound effects with generative adversarial networks, 2021.

S. Conan. Intuitive Control of Solid-Interaction Sounds Synthesis: Toward Sonic Metaphors. 2014.

S. Conan, E. Thoret, M. Aramaki, O. Derrien, C. Gondre, S. Ystad, and R. Kronland-Martinet. An Intuitive Synthesizer of Continuous-Interaction Sounds: Rubbing, Scratching, and Rolling. *Comput. Music. J.*, 38(4):24–37, 2014. doi: 10.1162/COMJ_a_00266. URL `https://doi.org/10.1162/COMJ_a_00266`.

A. Cont. *Musical Research at Ircam*. Taylor & Francis, Apr 2013. doi: 10.1080/07494467.2013.774121. URL `https://hal.inria.fr/hal-00930937`.

I. Corbett. What data compression does to your music, 2012. URL `https://www.soundonsound.com/techniques/what-data-compression-does-your-music`. Accessed 31 May 2021.

S. Crab. 120 Years Of Electronic Music - The history of electronic music from 1800 to 2015, 2016. URL `http://120years.net/https://120years.net/category/date/1800-1900/`.

S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech, Signal Process.*, pages 357–366, 1980.

M. Dendrinos, S. Bakamidis, and G. Carayannis. Speech enhancement from noise: A regenerative approach. *Speech Commun.*, 10(1):45–57, 1991. doi: 10.1016/0167-6393(91)90027-Q. URL https://doi.org/10.1016/0167-6393(91)90027-Q.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, June 2009.

J. Deng, B. W. Schuller, F. Eyben, D. Schuller, Z. Zhang, H. Francois, and E. Oh. Exploiting time-frequency patterns with LSTM-RNNs for low-bitrate audio restoration. *Neural Comput. Appl.*, 32(4):1095–1107, 2020. doi: 10.1007/s00521-019-04158-0. URL https://doi.org/10.1007/s00521-019-04158-0.

E. L. Denton, S. Gross, and R. Fergus. Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks. volume abs/1611.06430, 2016.

P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever. Jukebox: A generative model for music. *CoRR*, abs/2005.00341, 2020.

S. Dieleman. Generating Music in the Waveform Domain. https://benanne.github.io/2020/03/24/audio-generation.html, 2020.

S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *ICASSP*, pages 6964–6968, Florence, Italy, May 2014. doi: 10.1109/ICASSP.2014.6854950.

S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *NeurIPS*, pages 8000–8010, Montréal, Canada, Dec. 2018.

M. Dietz, L. Liljeryd, K. Kjorling, and O. Kunz. Spectral Band Replication, a novel approach in audio coding. In *Audio Engineering Society Convention 112*. Audio Engineering Society, 2002.

C. Donahue, B. Li, and R. Prabhavalkar. Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 5024–5028, Calgary, AB, Canada, April 2018. IEEE.

C. Donahue, J. McAuley, and M. Puckette. Adversarial Audio Synthesis. In *Proc. of the 7th International Conference on Learning Representations, ICLR*, May 2019.

C. Dong, C. C. Loy, K. He, and X. Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016. doi: 10.1109/TPAMI.2015.2439281. URL https://doi.org/10.1109/TPAMI.2015.2439281.

J. Dong, W. Wang, and J. A. Chambers. Audio super-resolution using analysis dictionary learning. In *2015 IEEE International Conference on Digital Signal Processing, DSP*, pages 604–608, Singapore, July 2015. IEEE. doi: 10.1109/ICDSP.2015.7251945. URL `https://doi.org/10.1109/ICDSP.2015.7251945`.

J. Drysdale, M. Tomczak, and J. Hockman. Adversarial Synthesis of Drum Sounds. In *DAFX*, 2020.

J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *Proc. of the 34th International Conference on Machine Learning, ICML*, Sydney, NSW, Australia, Aug. 2017.

J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts. GANSynth: Adversarial Neural Audio Synthesis. In *Proc. of the 7th International Conference on Learning Representations, ICLR*, May 2019.

J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts. DDSP: Differentiable Digital Signal Processing. In *Proc. of the 8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia, Apr. 2020.

Y. Ephraim. Statistical-model-based speech enhancement systems. *Proceedings of the IEEE*, 80(10):1526–1555, 1992. doi: 10.1109/5.168664.

H. Erdogan, J. R. Hershey, S. Watanabe, and J. L. Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 708–712, South Brisbane, Queensland, Australia, April 2015. IEEE.

P. Esling, A. Chemla-Romeu-Santos, and A. Bitton. Bridging Audio Analysis, Perception and Synthesis with Perceptually-regularized Variational Timbre Spaces. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, pages 175–181, Paris, France, September 2018a.

P. Esling, A. Chemla-Romeu-Santos, and A. Bitton. Generative timbre spaces with variational audio synthesis. In *Proc. of the 21st International Conference on Digital Audio Effects DAFx-18*, Aveiro, Portugal, Sept. 2018b.

P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos. Universal audio synthesizer control with normalizing flows. *Journal of Applied Sciences*, 2019.

C. Esteban, S. L. Hyland, and G. Rätsch. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *CoRR*, 2017.

Y. Fan, Y. Qian, F. Xie, and F. K. Soong. TTS synthesis with bidirectional LSTM based recurrent neural networks. In *INTERSPEECH*, Sept. 2014.

K. Fisher and A. Scherlis. WaveMedic: Convolutional Neural Networks for Speech Audio Enhancement. 2016.

S. Fu, C. Liao, Y. Tsao, and S. Lin. MetricGAN: Generative Adversarial Networks based Black-box Metric Scores Optimization for Speech Enhancement. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2031–2041, Long Beach, California, USA, June 2019. PMLR.

J. Fuegi and J. Francis. Lovelace & Babbage and the creation of the 1843 'notes'. *Inroads*, 6(3):78–86, 2015. doi: 10.1145/2810201. URL `https://doi.org/10.1145/2810201`.

L. Gao, K. Xu, H. Wang, and Y. Peng. Multi-Representation Knowledge Distillation For Audio Classification. *CoRR*, abs/2002.09607, 2020.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

F. G. Germain, Q. Chen, and V. Koltun. Speech Denoising with Deep Feature Losses. In *20th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2723–2727, Graz, Austria, September 2019. ISCA.

A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference. *CoRR*, abs/2103.13630, 2021.

A. Gibiansky, S. Ö. Arik, G. F. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou. Deep Voice 2: Multi-Speaker Neural Text-to-Speech. In *Advances in Neural Information Processing Systems 30: Annual Conference on NeurIPS*, pages 2966–2974, Long Beach, CA, USA, December 2017.

I. J. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *CoRR*, abs/1701.00160, 2017. URL `http://arxiv.org/abs/1701.00160`.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative Adversarial Nets. In *NeurIPS*, pages 2672–2680, Montreal, Quebec, Canada, Dec. 2014.

R. Gordon. Synthesizing Drums: The Bass Drum. *Sound On Sound*, Jan. 2002a. URL `https://www.soundonsound.com/techniques/synthesizing-drums-bass-drum`.

R. Gordon. Synthesizing drums: The snare drum. *Sound On Sound*, Jan. 2002b. URL `https://www.soundonsound.com/techniques/synthesizing-drums-snare-drum`.

M. Grachten, E. Deruty, and A. Tanguy. Auto-adaptive Resonance Equalization using Dilated Residual Networks. In *Proceedings of the 20th ISMIR*, Delft, The

Netherlands, 2019. URL http://archives.ismir.net/ismir2019/paper/000048.pdf.

M. Grachten, S. Lattner, and E. Deruty. BassNet: A Variational Gated Autoencoder for Conditional Generation of Bass Guitar Tracks with Learned Interactive Control. *Applied Sciences*, 10(18), 2020. ISSN 2076-3417. doi: 10.3390/app10186627. URL https://www.mdpi.com/2076-3417/10/18/6627.

G. Greshler, T. R. Shaham, and T. Michaeli. Catch-A-Waveform: Learning to Generate Audio from a Single Short Example. *CoRR*, abs/2106.06426, 2021.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. A Kernel Two-Sample Test. *J. of Mach. Learn. Res.*, 13:723–773, 2012.

D. W. Griffin and J. S. Lim. Signal estimation from modified short-time Fourier transform. In *ICASSP*, pages 804–807, Boston, Massachusetts, USA, Apr. 1983. doi: 10.1109/ICASSP.1983.1172092.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. In *NeurIPS*, pages 5769–5779, Long Beach, CA, USA, Dec. 2017.

A. Gupta, B. Shillingford, Y. M. Assael, and T. C. Walters. Speech Bandwidth Extension with WaveNet. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, pages 205–208, New Paltz, NY, USA, October 2019. IEEE. doi: 10.1109/WASPAA.2019.8937169. URL https://doi.org/10.1109/WASPAA.2019.8937169.

C. Gupta, P. Kamath, and L. Wyse. Signal Representations for Synthesizing Audio Textures with Generative Adversarial Networks. *CoRR*, abs/2103.07390, 2021.

G. Hadjeres and L. Crestel. Vector Quantized Contrastive Predictive Coding for Template-based Music Generation. *CoRR*, 2020.

G. Hadjeres, F. Pachet, and F. Nielsen. Deepbach: a steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1362–1371, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision, ICCV*, Santiago, Chile, Dec. 2015.

O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-Efficient Image Recognition with Contrastive Predictive Coding. *CoRR*, abs/1905.09272, 2019.

G. Hinton, O. Vinyals, and J. Dean. Dark Knowledge. In *Toyota Technological Institute at Chicago, TTIC*, 2014.

G. E. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531, 2015.

E. Hosseini-Asl, Y. Zhou, C. Xiong, and R. Socher. A Multi-Discriminator CycleGAN for Unsupervised Non-Parallel Speech Domain Adaptation. In *Proc. of the 19th Annual Conference of the International Speech Communication Association*, Hyderabad, India, Sept. 2018.

W. Hsu, Y. Zhang, and J. R. Glass. Learning Latent Representations for Speech Generation and Transformation. In *18th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 1273–1277, Stockholm, Sweden, August 2017.

Y. Hu, Y. Liu, S. Lv, M. Xing, S. Zhang, Y. Fu, J. Wu, B. Zhang, and L. Xie. DCCRN: Deep Complex Convolution Recurrent Network for Phase-Aware Speech Enhancement. In *21st Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2472–2476, Shanghai, China, October 2020. ISCA. doi: 10.21437/Interspeech.2020-2537. URL `https://doi.org/10.21437/Interspeech.2020-2537`.

C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music Transformer: Generating Music with Long-Term Structure. In *ICLR (Poster)*. OpenReview.net, 2019a.

C. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai. AI Song Contest: Human-AI Co-Creation in Songwriting. *CoRR*, abs/2010.05388, 2020. URL `https://arxiv.org/abs/2010.05388`.

H. Huang, Z. Li, R. He, Z. Sun, and T. Tan. IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 52–63, Montréal, Canada, December 2018.

S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse. TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer. In *Proc. of the 7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019b.

M. Huzaifah and L. Wyse. Deep generative models for musical audio synthesis. *CoRR*, abs/2006.06426, 2020.

V. Iashin and E. Rahtu. Taming visually guided sound generation. 2021.

R. I.-T. P. International Telecommunications Union–Radiocommunication (ITU-T).

U. Isik, R. Giri, N. Phansalkar, J. Valin, K. Helwani, and A. Krishnaswamy. PoCoNet: Better Speech Enhancement with Frequency-Positional Embeddings, semi-supervised conversational data, and biased loss. In H. Meng, B. Xu, and T. F. Zheng, editors, *Interspeech 2020, 21st Annual Conference of*

the International Speech Communication Association, pages 2487–2491, Shanghai, China, October 2020. ISCA. doi: 10.21437/Interspeech.2020-3027. URL https://doi.org/10.21437/Interspeech.2020-3027.

P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 5967–5976, Honolulu, HI, USA, July 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.632. URL https://doi.org/10.1109/CVPR.2017.632.

Jae Lim and A. Oppenheim. All-pole modeling of degraded speech. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(3):197–210, 1978. doi: 10.1109/TASSP.1978.1163086.

J.-N. Jeanneney. *L'Écho du siècle, dictionnaire historique de la radio et de la télévision en France*. Hachette Littératures et Arte Éditions, 1999.

S. Ji, J. Luo, and X. Yang. A Comprehensive Survey on Deep Music Generation: Multi-level Representations, algorithms, evaluations, and future directions. *CoRR*, abs/2011.06801, 2020.

N. M. Kalibhat, Y. Balaji, and S. Feizi. Winning Lottery Tickets in Deep Generative Models. In *35th Conference on Artificial Intelligence, AAAI*, pages 8038–8046, Virtual Event, February 2021. AAAI Press.

T. Kaneko and H. Kameoka. Parallel-Data-Free Voice Conversion Using Cycle-Consistent Adversarial Networks. *CoRR*, abs/1711.11293, 2017.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *CoRR*, abs/1812.04948, 2018.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and Improving the Image Quality of StyleGAN. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 8107–8116, Seattle, WA, USA, June 2020. IEEE.

K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi. Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms. *CoRR*, abs/1812.08466, 2018.

S. Kim, S. Lee, J. Song, and S. Yoon. FloWaveNet: A Generative Flow for Raw Audio. *CoRR*, abs/1811.02155, 2018.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR*, San Diego, CA, USA, May 2015.

D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 10236–10245, Montréal, Canada, December 2018.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proc. of the 2nd International Conference on Learning Representations, ICLR*, Banff, AB, Canada, Apr. 2014.

D. P. Kingma, T. Salimans, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *CoRR*, abs/1606.04934, 2016.

J. Kleimola. *Nonlinear abstract sound synthesis algorithms.* PhD thesis, School of Electrical Engineering, 2013.

M. Kolbæk, Z. Tan, and J. Jensen. Speech enhancement using Long Short-Term Memory based recurrent Neural Networks for noise robust Speaker Verification. In *2016 IEEE Spoken Language Technology Workshop, SLT 2016, , December 13-16, 2016*, pages 305–311, San Diego, CA, USA, December 2016. IEEE.

J. Kong, J. Kim, and J. Bae. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, Virtual conference, December 2020a.

Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition. *IEEE ACM Trans. Audio Speech Lang. Process.*, 28:2880–2894, 2020b.

J. Kontio, L. Laaksonen, and P. Alku. Neural Network-Based Artificial Bandwidth Expansion of Speech. *IEEE Trans. Speech Audio Process.*, 15(3):873–881, 2007. doi: 10.1109/TASL.2006.885934. URL `https://doi.org/10.1109/TASL.2006.885934`.

V. Kuleshov, S. Z. Enam, and S. Ermon. Audio Super-Resolution using Neural Networks. In *5th International Conference on Learning Representations, ICLR*, Toulon, France, April 2017. OpenReview.net. URL `https://openreview.net/forum?id=S1gNakBFx`.

K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *Proc. of the Annual Conference on Neural Information Processing Systems, NIPS*, Vancouver, BC, Canada, Dec. 2019.

R. Kumar, K. Kumar, V. Anand, Y. Bengio, and A. C. Courville. NU-GAN: high resolution neural upsampling with GAN. *CoRR*, abs/2010.11362, 2020. URL `https://arxiv.org/abs/2010.11362`.

M. Lagrange and F. Gontier. Bandwidth Extension of Musical Audio Signals With No Side Information Using Dilated Convolutional Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 801–805, Barcelona, Spain, May 2020. IEEE. doi: 10.1109/ICASSP40776.2020.9054194. URL `https://doi.org/10.1109/ICASSP40776.2020.9054194`.

E. Larsen and R. M. Aarts. *Audio bandwidth extension: application of psychoacoustics, signal processing and loudspeaker design*. John Wiley & Sons, 2005.

S. Lattner. *Modeling Musical Structure with Artificial Neural Networks*. PhD thesis, Institute of Computational Perception, Johannes Kepler University, Linz, 2019.

S. Lattner and M. Grachten. High-Level Control of Drum Track Generation Using Learned Patterns of Rhythmic Interaction. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, New Paltz, NY, USA, Oct. 2019.

S. Lattner and J. Nistal. Stochastic Restoration of Heavily Compressed Musical Audio Using Generative Adversarial Networks. *Electronics*, 10(11), 2021. ISSN 2079-9292. doi: 10.3390/electronics10111349. URL `https://www.mdpi.com/2079-9292/10/11/1349`.

S. Lee, U. Hwang, S. Min, and S. Yoon. A SeqGAN for Polyphonic Music Generation. *CoRR*, abs/1710.11418, 2017a. URL `http://arxiv.org/abs/1710.11418`.

Y. Lee, A. Rabiee, and S. Lee. Emotional End-to-End Neural Speech Synthesizer. *CoRR*, abs/1711.05447, 2017b.

J. Li, R. Zhao, J. Huang, and Y. Gong. Learning small-size DNN with output-distribution-based criteria. In H. Li, H. M. Meng, B. Ma, E. Chng, and L. Xie, editors, *15th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 1910–1914, Singapore, September 2014. ISCA.

K. Li and C. Lee. A deep neural network approach to speech bandwidth expansion. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 4395–4399, South Brisbane, Queensland, Australia, April 2015. IEEE. doi: 10.1109/ICASSP.2015.7178801. URL `https://doi.org/10.1109/ICASSP.2015.7178801`.

Z. Li, L. Dai, Y. Song, and I. V. McLoughlin. A Conditional Generative Model for Speech Enhancement. *Circuits Syst. Signal Process.*, 37(11):5005–5022, 2018. doi: 10.1007/s00034-018-0798-4. URL `https://doi.org/10.1007/s00034-018-0798-4`.

T. Lidy. CQT-based convolutional neural networks for audio scene classification and domestic audio tagging. In *DCASE*, Sept. 2016.

T. Lim, R. A. Yeh, Y. Xu, M. N. Do, and M. Hasegawa-Johnson. Time-frequency networks for audio super-resolution. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 646–650, Calgary, AB, Canada, April 2018. IEEE.

Z. Ling, Y. Ai, Y. Gu, and L. Dai. Waveform Modeling and Generation Using Hierarchical Recurrent Neural Networks for Speech Bandwidth Extension. *IEEE ACM Trans. Audio Speech Lang. Process.*, 26(5):883–894, 2018.

M. Liu and O. Tuzel. Coupled Generative Adversarial Networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems, NeurIPS*.

P. Loizou. *Speech Enhancement: Theory and Practice*. 01 2007. ISBN 9780429096181. doi: 10.1201/b14529.

Z. Luo, J. Chen, T. Takiguchi, and Y. Ariki. Emotional voice conversion using neural networks with arbitrary scales F0 based on wavelet transform. *EURASIP J. Audio Speech Music. Process.*, 2017:18, 2017. doi: 10.1186/s13636-017-0116-2. URL https://doi.org/10.1186/s13636-017-0116-2.

S. Maiti and M. I. Mandel. Parametric Resynthesis With Neural Vocoders. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, pages 303–307, New Paltz, NY, USA, October 2019. IEEE. doi: 10.1109/WASPAA.2019.8937165. URL https://doi.org/10.1109/WASPAA.2019.8937165.

J. Makhoul and M. G. Berouti. High-frequency regeneration in speech coding systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 428–431, Washington, D. C., USA, April 1979. IEEE. doi: 10.1109/ICASSP.1979.1170672. URL https://doi.org/10.1109/ICASSP.1979.1170672.

M. I. Mandel and Y. S. Cho. Audio super-resolution using concatenative resynthesis. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, pages 1–5, New Paltz, NY, USA, October 2015. IEEE. doi: 10.1109/WASPAA.2015.7336890. URL https://doi.org/10.1109/WASPAA.2015.7336890.

A. Marafioti, N. Perraudin, N. Holighaus, and P. Majdak. Adversarial Generation of Time-Frequency Features with application in audio synthesis. In K. Chaudhuri and R. Salakhutdinov, editors, *Proc. of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 4352–4362, Long Beach, California, USA, June 2019. PMLR.

B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, et al. librosa/librosa: 0.7.2, Jan. 2020.

S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SkxKPDv5xl.

D. Michelsanti and Z. Tan. Conditional Generative Adversarial Networks for Speech Enhancement and Noise-Robust Speaker Verification. In *Proc. of the 18th Annual Conference of the International Speech Communication Association, INTERSPEECH*, Stockholm, Sweden, Aug. 2017.

E. Miranda. *Computer Sound Design: Synthesis Techniques and Programming*. 01 2002. ISBN 9780080490755. doi: 10.4324/9780080490755.

M. Miron and M. Davies. High frequency magnitude spectrogram reconstruction for music mixtures using convolutional autoencoders. In *Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18)*, pages 173–180. IEEE, 2018.

O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *CoRR*, 2016.

V. Moorefield. *The Producer as Composer: Shaping the Sounds of Popular Music*, volume 4. 03 2005. doi: 10.1017/S1478572207000564.

N. Mor, L. Wolf, A. Polyak, and Y. Taigman. A Universal Music Translation Network. *CoRR*, abs/1805.07848, 2018. URL `http://arxiv.org/abs/1805.07848`.

M. Morrison, R. Kumar, K. Kumar, P. Seetharaman, A. Courville, and Y. Bengio. Chunked Autoregressive GAN for Conditional Waveform Synthesis. 2021.

H. G. Musmann. Genesis of the MP3 audio coding standard. *IEEE Trans. Consumer Electron.*, 52(3):1043–1049, 2006.

M. Nishimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda. Singing Voice Synthesis Based on Deep Neural Networks. In *17th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2478–2482, San Francisco, CA, USA, September 2016.

J. Nistal, S. Lattner, and G. Richard. DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks. In *Proc. of the 21st International Society for Music Information Retrieval, ISMIR*, Montréal, Canada, 2020.

J. Nistal, C. Aouameur, S. Lattner, and G. Richard. VQCPC-GAN: Variable-Length Adversarial Audio Synthesis using Vector-Quantized Contrastive Predictive Coding. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, New Paltz, NY, USA, November 2021a.

J. Nistal, S. Lattner, and G. Richard. DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis with GANs. *Proc. of ISMIR*, November 2021b.

J. Nistal, S. Lattner, and G. Richard. Comparing Representations for Audio Synthesis Using Generative Adversarial Networks. In *Proc. of the 28th European Signal Processing Conference, EUSIPCO*, Amsterdam, NL, Jan. 2021c.

A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *ICML*, pages 2642–2651, Sydney, NSW, Australia, Aug. 2017.

J. Ortega-Garcia and J. Gonzalez-Rodriguez. Overview of speech enhancement techniques for automatic speaker recognition. In *The 4th International Conference on Spoken Language Processing*, Philadelphia, PA, USA, October 1996. ISCA.

F. Pachet. The Continuator: Musical Interaction with Style. In *Proceedings of the International Computer Music Conference, ICMC*, Gothenburg, Sweden, September 2002.

T. L. Paine, P. Khorrami, S. Chang, Y. Zhang, P. Ramachandran, M. A. Hasegawa-Johnson, and T. S. Huang. Fast Wavenet Generation Algorithm. *CoRR*, abs/1611.09482, 2016. URL `http://arxiv.org/abs/1611.09482`.

C. Palombini. Pierre Schaeffer, 1953: towards an experimental music. *Music & Letters*, 74(4):542–557, 1993.

N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *5th International Conference on Learning Representations, ICLR*, Toulon, France, April 2017.

S. R. Park and J. Lee. A Fully Convolutional Neural Network for Speech Enhancement. In F. Lacerda, editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 1993–1997. ISCA, 2017. URL `http://www.isca-speech.org/archive/Interspeech_2017/abstracts/1465.html`.

T. Park, M. Liu, T. Wang, and J. Zhu. Semantic Image Synthesis With Spatially-Adaptive Normalization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2337–2346. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00244. URL `http://openaccess.thecvf.com/content_CVPR_2019/html/Park_Semantic_Image_Synthesis_With_Spatially-Adaptive_Normalization_CVPR_2019_paper.html`.

S. Pascual, A. Bonafonte, and J. Serrà. SEGAN: Speech Enhancement Generative Adversarial Network. In F. Lacerda, editor, *18th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 3642–3646, Stockholm, Sweden, August 2017. ISCA.

S. Pascual, J. Serrà, and A. Bonafonte. Towards Generalized Speech Enhancement with Generative Adversarial Networks. In G. Kubin and Z. Kacic, editors, *20th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 1791–1795, Graz, Austria, September 2019. ISCA. doi: 10.21437/Interspeech.2019-2688. URL `https://doi.org/10.21437/Interspeech.2019-2688`.

W. S. Peebles, J. Peebles, J. Zhu, A. A. Efros, and A. Torralba. The Hessian Penalty: A Weak Prior for Unsupervised Disentanglement. In *Computer Vision - ECCV - 16th European Conference*, volume 12351 of *Lecture Notes in Computer Science*, pages 581–597, Glasgow, UK, August 2020. Springer.

H. Phan, I. V. McLoughlin, L. D. Pham, O. Y. Chén, P. Koch, M. D. Vos, and A. Mertins. Improving GANs for Speech Enhancement. *IEEE Signal Process. Lett.*, 27:1700–1704, 2020.

S. Pidhorskyi, D. A. Adjeroh, and G. Doretto. Adversarial Latent Autoencoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 14092–14101, Seattle, WA, USA, June 2020. Computer Vision Foundation / IEEE.

T. Pinch and F. Trocco. *Analog Days, the invention and impact of the Moog synthesizer*. Harvard University Press, 2002.

W. Ping, K. Peng, A. Gibiansky, S. Ö. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller. Deep Voice 3: 2000-Speaker Neural Text-to-Speech. *CoRR*, abs/1710.07654, 2017.

W. Ping, K. Peng, and J. Chen. ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech. *CoRR*, abs/1807.07281, 2018.

W. Ping, K. Peng, K. Zhao, and Z. Song. WaveFlow: A Compact Flow-based Model for Raw Audio. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 7706–7716, Virtual Event, July 2020. PMLR.

A. Porov, E. Oh, K. Choo, H. Sung, J. Jeong, K. Osipov, and H. Francois. Music enhancement by a novel CNN architecture. In *Audio Engineering Society Convention 145*. Audio Engineering Society, 2018.

R. Prenger, R. Valle, and B. Catanzaro. WaveGlow: A Flow-based Generative Network for Speech Synthesis. *CoRR*, abs/1811.00002, 2018.

L. Pruvost, B. Scherrer, M. Aramaki, S. Ystad, and R. Kronland-Martinet. Perception-based interactive sound synthesis of morphing solids' interactions. pages 1–4, 11 2015. doi: 10.1145/2820903.2820914.

A. Ramires, P. Chandna, X. Favory, E. Gómez, and X. Serra. Neural Percussive Synthesis Parameterised by High-Level Timbral Features. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, May 2020.

E. Ravelli, G. Richard, and L. Daudet. Audio Signal Representations for Indexing in the Transform Domain. *IEEE Trans. Audio, Speech, Language Process.*, 18 (3):434–446, 2010. doi: 10.1109/TASL.2009.2025099.

D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 1530–1538, Lille, France, July 2015.

C. Roads, A. Piccialli, G. D. Poli, and S. T. Pope. *Musical Signal Processing*. Swets & Zeitlinger, USA, 1997. ISBN 9026514832.

A. Roberts, J. Engel, and D. Eck. Hierarchical Variational Autoencoders for Music. In *Workshop on Machine Learning for Creativity and Design, NIPS*, 2017. URL `https://nips2017creativity.github.io/doc/Hierarchical_Variational_Autoencoders_for_Music.pdf`.

F. Roche. *Music sound synthesis using machine learning: Towards a perceptually relevant control space*. PhD thesis, 09 2020.

F. Roche, T. Hueber, S. Limier, and L. Girin. Autoencoders for music sound synthesis: a comparison of linear, shallow, deep and variational models. *CoRR*, abs/1806.04096, 2018.

A. Roebel and F. Bous. Towards universal neural vocoding with a multi-band excited wavenet. *CoRR*, abs/2110.03329, 2021. URL `https://arxiv.org/abs/2110.03329`.

R. M. Rustamov. Closed-form Expressions for Maximum Mean Discrepancy with Applications to Wasserstein Auto-Encoders. *CoRR*, abs/1901.03227, 2019.

A. Saeed, D. Grangier, and N. Zeghidour. Contrastive Learning of General-Purpose Audio Representations. *CoRR*, 2020.

Y. Saito, S. Takamichi, and H. Saruwatari. Statistical Parametric Speech Synthesis Incorporating Generative Adversarial Networks. *IEEE/ACM Trans. Audio Speech Lang. Process.*, 26:84–96, 2018.

T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. In *NeurIPS*, pages 2226–2234, Barcelona, Spain, Dec. 2016.

S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised Pre-Training for Speech Recognition. In *INTERSPEECH*, Graz, Austria, Sept. 2019.

D. Schwarz. Corpus-Based Concatenative Synthesis. *IEEE Signal Process. Mag.*, 24(2):92–104, 2007.

M. L. Seltzer, D. Yu, and Y. Wang. An investigation of deep neural networks for noise robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 7398–7402, Vancouver, BC, Canada, May 2013. IEEE.

J. Serrà, S. Pascual, and C. Segura. Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion. In *Advances in Neural Information Processing Systems 32, NeurIPS*, pages 6790–6800, Vancouver, BC, Canada, December 2019.

X. Serra. State of the Art and Future Directions in Musical Sound Synthesis. In *IEEE 9th Workshop on Multimedia Signal Processing, MMSP*, pages 9–12. IEEE, October 2007.

X. Serra and J. O. Smith. Spectral Modeling Synthesis: A Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition. *Computer Music Journal*, 14:12–24, 1990. doi: http://doi.org/10.2307/3680788. URL `http://hdl.handle.net/10230/33796`. SMS.

X. Serra, G. Widmer, and M. Leman. *A Roadmap for Sound and Music Computing*. The S2S Consortium, 2007. URL `http://hdl.handle.net/10230/34060`.

T. R. Shaham, T. Dekel, and T. Michaeli. SinGAN: Learning a Generative Model From a Single Natural Image. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 4569–4579, Seoul, Korea (South), November 2019. IEEE.

P. Shaw, J. Uszkoreit, and A. Vaswani. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 464–468, New Orleans, Louisiana, USA, June 2018.

J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu. Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 4779–4783, Calgary, AB, Canada, April 2018.

Y. Shen, J. Gu, X. Tang, and B. Zhou. Interpreting the Latent Space of GANs for Semantic Face Editing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 9240–9249, Seattle, WA, USA, June 2020. IEEE.

I. Simon and S. Oore. Performance RNN: Generating Music with Expressive Timing and Dynamics. `https://magenta.tensorflow.org/performance-rnn`, 2017.

J. Skoglund and J. Valin. Improving Opus Low Bit Rate Quality with Neural Speech Synthesis. In H. Meng, B. Xu, and T. F. Zheng, editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 2847–2851. ISCA, 2020.

J. O. Smith. *Physical Audio Signal Processing*. `http://ccrma.stanford.edu/~jos/pasp/`, a. online book, 2010 edition.

J. O. Smith. *Spectral Audio Signal Processing*. `http://ccrma.stanford.edu/~jos/sasp/`, b. online book, 2011 edition.

J. O. Smith. Viewpoints on the History of Digital Synthesis. In *Proceedings of the International Computer Music Conference, ICMC*, Montreal, Quebec, Canada, October 1991. Michigan Publishing.

J. O. Smith. Virtual Acoustic Musical Instruments: Review and Update. *Journal of New Music Research*, 33:283–304, 09 2004. doi: 10.1080/09298210420003178 59.

J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, and A. Courville. Char2Wav: End-to-End Speech Synthesis. In *International Conference on Learning Representations, ICLR 2017*, 2017.

J. Spijkervet and J. A. Burgoyne. Contrastive Learning of Musical Representations. *CoRR*, 2021.

C. J. Steinmetz and J. D. Reiss. Randomized Overdrive Neural Networks. *CoRR*, abs/2010.04237, 2020. URL `https://arxiv.org/abs/2010.04237`.

S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *J. Acoust. Soc. Am.*, 8(3):185–190, 1937. doi: 10.1121/1.1915893.

J. Su, Z. Jin, and A. Finkelstein. HiFi-GAN: High-Fidelity Denoising and Dereverberation Based on Speech Deep Features in Adversarial Networks. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 4506–4510. ISCA, 2020. doi: 10.21437/Interspeech.2020-2143. URL `https://doi.org/10.21437/Interspeech.2020-2143`.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2818–2826, Las Vegas, NV, USA, June 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.308.

Z. Tang, D. Wang, and Z. Zhang. Recurrent neural network training with dark knowledge transfer. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 5900–5904, Shanghai, China, March 2016. IEEE. doi: 10.1109/ICASSP.2016.7472809.

T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, and C. Colomes. PEAQ-The ITU standard for objective measurement of perceived audio quality. *Journal of the Audio Engineering Society*, 48(1/2):3–29, 2000.

C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal. Deep Complex Networks. In *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, April 2018. OpenReview.net.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Proc. of the 9th ISCA Speech Synthesis Workshop*, Sunnyvale, CA, USA, Sept. 2016a.

A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves. Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 4790–4798, Barcelona, Spain, December 2016b.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. In *Proc. of the 33rd International Conference on Machine Learning, ICML*, New York City, NY, USA, June 2016c.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural Discrete Representation Learning. In *NeurIPS*, Long Beach, CA, USA, Dec. 2017.

A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 3915–3923, Stockholmsmässan, Stockholm, Sweden, July 2018a.

A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *CoRR*, 2018b.

B. van Niekerk, L. Nortje, and H. Kamper. Vector-Quantized Neural Networks for Acoustic Unit Discovery in the ZeroSpeech 2020 Challenge. In *INTERSPEECH*, Shanghai, China, Oct. 2020.

S. Vasquez and M. Lewis. MelNet: A Generative Model for Audio in the Frequency Domain. *CoRR*, abs/1906.01083, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on NeurIPS*, pages 6000–6010, Long Beach, CA, USA, December 2017.

G. A. Velasco, N. Holighaus, M. Doerfler, and T. Grill. Constructing an invertible constant-Q transform with nonstationary Gabor frames. *Proceedings of the 14th International Conference on Digital Audio Effects, DAFx 2011*, 09 2011.

P. Verma and J. O. S. III. A Framework for Contrastive and Generative Learning of Audio Representations. *CoRR*, 2020.

C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. ISBN 9783540710509. URL `https://books.google.es/books?id=hV8o5R7_5tkC`.

A. Voynov and A. Babenko. Unsupervised Discovery of Interpretable Directions in the GAN Latent Space. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9786–9796, Virtual Event, July 2020. PMLR.

L. Wang and A. van den Oord. Multi-Format Contrastive Learning of Audio Representations. *CoRR*, 2021.

Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous. Tacotron: Towards End-to-End Speech Synthesis. In *18th Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 4006–4010, Stockholm, Sweden, August 2017.

D. S. Williamson and D. Wang. Speech dereverberation and denoising using complex ratio masks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 5590–5594. IEEE, 2017.

D. S. Williamson, Y. Wang, and D. Wang. Complex Ratio Masking for Monaural Speech Separation. *IEEE ACM Trans. Audio Speech Lang. Process.*, 24(3): 483–492, 2016. doi: 10.1109/TASLP.2015.2512042. URL `https://doi.org/10.1109/TASLP.2015.2512042`.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017.

Y. Xu, J. Du, L. Dai, and C. Lee. A Regression Approach to Speech Enhancement Based on Deep Neural Networks. *IEEE ACM Trans. Audio Speech Lang. Process.*, 23(1):7–19, 2015. doi: 10.1109/TASLP.2014.2364452. URL `https://doi.org/10.1109/TASLP.2014.2364452`.

R. Yamamoto, E. Song, and J. Kim. Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 6199–6203, Barcelona, Spain, May 2020. IEEE.

L.-P. Yang and Q.-J. Fu. Spectral subtraction-based speech enhancement for cochlear implant patients in background noise. *The Journal of the Acoustical Society of America*, 117:1001–4, 04 2005. doi: 10.1121/1.1852873.

J. Yoon, D. Jarrett, and M. van der Schaar. Time-series Generative Adversarial Networks. In *NeurIPS*, Vancouver, BC, Canada, Dec. 2019.

S. Ystad. *Sound Modeling Using a Combination of Physical and Signal Models*. PhD thesis, March 1998.

S. Ystad, M. ARAMAKI, and R. Kronland-Martinet. Timbre from Sound Synthesis and High-level Control Perspectives. In *Timbre: Acoustics, Perception, and Cognition*, volume 69 of *Springer Handbook of Auditory Research Series (SHAR)*, pages 361–389. Springer Nature, 2019. URL `https://hal.archives-ouvertes.fr/hal-01766645`.

L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, Feb. 2017.

M. Yuan and Y. Peng. Text-to-image Synthesis via Symmetrical Distillation Networks. In *2018 ACM Multimedia Conference on Multimedia Conference, MM*, pages 1407–1415, Seoul, Republic of Korea, October 2018. ACM. doi: 10.1145/3240508.3240559.

M. Yuan and Y. Peng. CKD: Cross-Task Knowledge Distillation for Text-to-Image Synthesis. *IEEE Trans. Multim.*, 22(8):1955–1968, 2020. doi: 10.1109/TMM.2019.2951463.

S. Zhao, T. H. Nguyen, and B. Ma. Monaural Speech Enhancement with Complex Convolutional Block Attention Module and Joint Time Frequency Losses. *CoRR*, abs/2102.01993, 2021.

Z. Zhao, H. Liu, and T. Fingscheidt. Convolutional Neural Networks to Enhance Coded Speech. *IEEE ACM Trans. Audio Speech Lang. Process.*, 27(4):663–678, 2019. doi: 10.1109/TASLP.2018.2887337. URL `https://doi.org/10.1109/TASLP.2018.2887337`.

J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *CoRR*, abs/1703.10593, 2017.

M. Zhu and S. Gupta. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. In *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, April 2018. OpenReview.net.

Z. Zhu, J. H. Engel, and A. Y. Hannun. Learning Multiscale Features Directly from Waveforms. In *INTERSPEECH*, pages 1305–1309, San Francisco, CA, USA, Sept. 2016. doi: 10.21437/Interspeech.2016-256.

ECOLE DOCTORALE

Titre : Synthèse Audio Musicale Contrôlable à l'aide de Réseaux Antagonistes Génératifs

Mots clés : apprentissage profond, synthèse audio neuronal, musique

Résumé :
Les synthétiseurs audio sont des instruments de musique électroniques qui génèrent des sons artificiels sous un certain contrôle paramétrique. Alors que les synthétiseurs ont évolué depuis leur popularisation dans les années 70, deux défis fondamentaux restent encore non résolus : 1) le développement de systèmes de synthèse répondant à des paramètres sémantiquement intuitifs ; 2) la conception de techniques de synthèse ≪universelles≫, indépendantes de la source à modéliser. Cette thèse étudie l'utilisation des réseaux adversariaux génératifs (ou GAN) pour construire de tels systèmes. L'objectif principal est de rechercher et de développer de nouveaux outils pour la production musicale, qui offrent des moyens intuitifs et expressifs de manipulation du son, par exemple en contrôlant des paramètres qui répondent aux propriétés perceptives du son et à d'autres caractéristiques.
Notre premier travail étudie les performances des GAN lorsqu'ils sont entraînés sur diverses représentations de signaux audio (par exemple, forme d'onde, représentations temps-fréquence). Ces expériences comparent différentes formes de données audio dans le contexte de la synthèse sonore tonale. Les résultats montrent que la représentation magnitude-fréquence instantanée et la transformée de Fourier à valeur complexe obtiennent les meilleurs résultats.
En s'appuyant sur ce résultat, notre travail suivant présente DrumGAN, un synthétiseur audio de sons percussifs. En conditionnant le modèle sur des caractéristiques perceptives décrivant des propriétés timbrales de haut niveau, nous démontrons qu'un contrôle intuitif peut être obtenu sur le processus de génération. Ce travail aboutit au développement d'un plugin VST générant de l'audio haute résolution et compatible avec les Stations de Travail Audio Numériques (STAN). Nous montrons un vaste matériel musical produit par des artistes professionnels de Sony ATV à l'aide de DrumGAN.
La rareté des annotations dans les ensembles de données audio musicales remet en cause l'application de méthodes supervisées pour la génération conditionnelle. Notre troisième contribution utilise une approche de distillation des connaissances pour extraire de telles annotations à partir d'un système d'étiquetage audio pré-entraîné. DarkGAN est un synthétiseur de sons tonaux qui utilise les probabilités de sortie d'un tel système (appelées ≪ étiquettes souples ≫) comme informations conditionnelles. Les résultats montrent que DarkGAN peut répondre modérément à de nombreux attributs intuitifs, même avec un conditionnement d'entrée hors distribution.
Les applications des GAN à la synthèse audio apprennent généralement à partir de données de spectrogramme de taille fixe, de manière analogue aux ≪données d'image≫ en vision par ordinateur ; ainsi, ils ne peuvent pas générer de sons de durée variable. Dans notre quatrième article, nous abordons cette limitation en exploitant une méthode autosupervisée pour l'apprentissage de caractéristiques discrètes à partir de données séquentielles. De telles caractéristiques sont utilisées comme entrée conditionnelle pour fournir au modèle des informations dépendant du temps par étapes. La cohérence globale est assurée en fixant le bruit d'entrée z (caractéristique en GANs). Les résultats montrent que, tandis que les modèles entraînés sur un schéma de taille fixe obtiennent une meilleure qualité et diversité audio, les nôtres peuvent générer avec compétence un son de n'importe quelle durée.
Une direction de recherche intéressante est la génération d'audio conditionnée par du matériel musical préexistant, par exemple, la génération d'un motif de batterie compte tenu de l'enregistrement d'une ligne de basse. Notre cinquième article explore une tâche prétexte simple adaptée à l'apprentissage de tels types de relations musicales complexes. Concrètement, nous étudions si un générateur GAN, conditionné sur des signaux audio musicaux hautement compressés, peut générer des sorties ressemblant à l'audio non compressé d'origine. Les résultats montrent que le GAN peut améliorer la qualité des signaux audio par rapport aux versions MP3 pour des taux de compression très élevés (16 et 32 kbit/s).
En conséquence directe de l'application de techniques d'intelligence artificielle dans des contextes musicaux, nous nous demandons comment la technologie basée sur l'IA peut favoriser l'innovation dans la pratique musicale. Par conséquent, nous concluons cette thèse en offrant une large perspective sur le développement d'outils d'IA pour la production musicale, éclairée par des considérations théoriques et des rapports d'utilisation d'outils d'IA dans le monde réel par des artistes professionnels.

Title : Exploring Generative Adversarial Networks for Controllable Musical Audio Synthesis

Keywords : deep learning, neural audio synthesis, music

Abstract : Audio synthesizers are electronic musical instruments that generate artificial sounds under some parametric control. While synthesizers have evolved since they were popularized in the 70s, two fundamental challenges are still unresolved : 1) the development of synthesis systems responding to semantically intuitive parameters ; 2) the design of ”universal,” source-agnostic synthesis techniques. This thesis researches the use of Generative Adversarial Networks (GAN) towards building such systems. The main goal is to research and develop novel tools for music production that afford intuitive and expressive means of sound manipulation, e.g., by controlling parameters that respond to perceptual properties of the sound and other high-level features.

Our first work studies the performance of GANs when trained on various common audio signal representations (e.g., waveform, time-frequency representations). These experiments compare different forms of audio data in the context of tonal sound synthesis. Results show that the Magnitude and Instantaneous Frequency of the phase and the complex-valued Short-Time Fourier Transform achieve the best results.

Building on this, our following work presents DrumGAN, a controllable adversarial audio synthesizer of percussive sounds. By conditioning the model on perceptual features describing high-level timbre properties, we demonstrate that intuitive control can be gained over the generation process. This work results in the development of a VST plugin generating full-resolution audio and compatible with any Digital Audio Workstation (DAW). We show extensive musical material produced by professional artists from Sony ATV using DrumGAN.

The scarcity of annotations in musical audio datasets challenges the application of supervised methods to conditional generation settings. Our third contribution employs a knowledge distillation approach to extract such annotations from a pre-trained audio tagging system. DarkGAN is an adversarial synthesizer of tonal sounds that employs the output probabilities of such a system (so-called “soft labels”) as conditional information. Results show that DarkGAN can respond moderately to many intuitive attributes, even with out-of-distribution input conditioning.

Applications of GANs to audio synthesis typically learn from fixed-size two-dimensional spectrogram data analogously to the ”image data” in computer vision ; thus, they cannot generate sounds with variable duration. In our fourth paper, we address this limitation by exploiting a self-supervised method for learning discrete features from sequential data. Such features are used as conditional input to provide step-wise time-dependent information to the model. Global consistency is ensured by fixing the input noise $\mathbf{z}$ (characteristic in adversarial settings). Results show that, while models trained on a fixed-size scheme obtain better audio quality and diversity, ours can competently generate audio of any duration.

One interesting direction for research is the generation of audio conditioned on preexisting musical material, e.g., the generation of some drum pattern given the recording of a bass line. Our fifth paper explores a simple pretext task tailored at learning such types of complex musical relationships. Concretely, we study whether a GAN generator, conditioned on highly compressed MP3 musical audio signals, can generate outputs resembling the original uncompressed audio. Results show that the GAN can improve the quality of the audio signals over the MP3 versions for very high compression rates (16 and 32 kbit/s).

As a direct consequence of applying artificial intelligence techniques in musical contexts, we ask how AI-based technology can foster innovation in musical practice. Therefore, we conclude this thesis by providing a broad perspective on the development of AI tools for music production, informed by theoretical considerations and reports from real-world AI tool usage by professional artists.

Institut Polytechnique de Paris
91120 Palaiseau, France