On the Environmental Impact of Deep Generative Models for Audio

Constance Douwes

▶ **To cite this version:**

Constance Douwes. On the Environmental Impact of Deep Generative Models for Audio. Artificial Intelligence [cs.AI]. Sorbonne Université, 2023. English. NNT : 2023SORUS074 . tel-04100243

HAL Id: tel-04100243
https://theses.hal.science/tel-04100243v1

Submitted on 17 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité Informatique
ED130 - Ecole doctorale Informatique, Télécommunications et Electronique (Paris)
Sciences et Technologie de la Musique et du Son (UMR 9912)
Institut de Recherche et de Coordination Accoustique Musique
Equipe Représentations musicales.

ON THE ENVIRONMENTAL IMPACT OF DEEP GENERATIVE MODELS FOR AUDIO

CONSTANCE DOUWES

SUPERVISED BY: PHILIPPE ESLING
DIRECTED BY: JEAN-PIERRE BRIOT

Defended on March, 2023

JURY :

Nick Bryan-Kinns (Reviewer)
Sébastien Loustau (Reviewer)
Emma Strubell
Peter Bryzgalov
Geoffroy Peeters
Evripidis Bampis
Philippe Esling (Supervisor)
Jean-Pierre Briot (Director)

Constance Douwes: *On the environmental impact of deep generative models for audio* ,
PhD candidate, © January 2023

We can't save the world by playing by the rules, because the rules have to be changed. Everything needs to change and it has to start today.

— Greta Thunberg, at the TEDxStockholm on November 24, 2018.

ABSTRACT

In this thesis, we investigate the environmental impact of deep learning models for audio generation and we aim to put computational cost at the core of the evaluation process. In particular, we focus on different types of deep learning models specialized in raw waveform audio synthesis. These models are now a key component of modern audio systems, and their use has increased significantly in recent years. Their flexibility and generalization capabilities make them powerful tools in many contexts, from text-to-speech synthesis to unconditional audio generation. However, these benefits come at the cost of expensive training sessions on large amounts of data, operated on energy-intensive dedicated hardware, which incurs large greenhouse gas emissions. The measures we use as a scientific community to evaluate our work are at the heart of this problem. Currently, deep learning researchers evaluate their works primarily based on improvements in accuracy, log-likelihood, reconstruction, or opinion scores, all of which overshadow the computational cost of generative models. Therefore, we propose using a new methodology based on Pareto optimality to help the community better evaluate their work's significance while bringing energy footprint – and *in fine* carbon emissions – at the same level of interest as the sound quality.

In the first part of this thesis, we present a comprehensive report on the use of various evaluation measures of deep generative models for audio synthesis tasks. Even though computational efficiency is increasingly discussed, quality measurements are the most commonly used metrics to evaluate deep generative models, while energy consumption is almost never mentioned. Therefore, we address this issue by estimating the carbon cost of training generative models and comparing it to other noteworthy carbon costs to demonstrate that it is far from insignificant.

In the second part of this thesis, we propose a large-scale evaluation of pervasive neural vocoders, which are a class of generative models used for speech generation, conditioned on mel-spectrogram. We introduce a multi-objective analysis based on Pareto optimality of both quality from human-based evaluation and energy consumption. Within this framework, we show that lighter models can perform better than more costly models. By proposing to rely on a novel definition of efficiency, we intend to provide practitioners with a decision basis for choosing the best model based on their requirements.

In the last part of the thesis, we propose a method to reduce the inference costs of neural vocoders, based on quantizated neural networks. We show a significant gain on the memory size and give some hints for the future use of these models on embedded hardware.

Overall, we provide keys to better understand the impact of deep generative models for audio synthesis as well as a new framework for developing models while accounting for their environmental impact. We hope that this work raises awareness on the need to investigate energy-efficient models simultaneously with high perceived quality.

RESUME

Cette thèse étudie l'impact environnemental des modèles d'apprentissage profond pour la génération audio et vise à mettre le coût de calcul au cœur du processus d'évaluation. En particulier, nous nous concentrons sur différents types de modèles d'apprentissage profond spécialisés dans la synthèse audio de formes d'onde brutes. Ces modèles sont désormais un élément clé des systèmes audio modernes, et leur utilisation a considérablement augmenté ces dernières années. Leur flexibilité et leurs capacités de généralisation en font des outils puissants dans de nombreux contextes, de la synthèse de texte à la parole à la génération audio inconditionnelle. Cependant, ces avantages se font au prix de sessions d'entraînement coûteuses sur de grandes quantités de données, exploitées sur du matériel dédié à forte consommation d'énergie, ce qui entraîne d'importantes émissions de gaz à effet de serre. Les mesures que nous utilisons en tant que communauté scientifique pour évaluer nos travaux sont au cœur de ce problème. Actuellement, les chercheurs en apprentissage profond évaluent leurs travaux principalement sur la base des améliorations de la précision, de la log-vraisemblance, de la reconstruction ou des scores d'opinion, qui occultent tous le coût de calcul des modèles génératifs. Par conséquent, nous proposons d'utiliser une nouvelle méthodologie basée sur l'optimalité de Pareto pour aider la communauté à mieux évaluer leurs travaux tout en ramenant l'empreinte énergétique – et *in fine* les émissions de carbone – au même niveau d'intérêt que la qualité du son.

Dans la première partie de cette thèse, nous présentons un rapport complet sur l'utilisation de diverses mesures d'évaluation des modèles génératifs profonds pour les tâches de synthèse audio. Bien que l'efficacité de calcul soit de plus en plus abordée, les mesures de qualité sont les plus couramment utilisées pour évaluer les modèles génératifs profonds, alors que la consommation d'énergie n'est presque jamais mentionnée. Nous abordons donc cette question en estimant le coût en carbone de la formation des modèles génératifs et en le comparant à d'autres coûts en carbone notables pour démontrer qu'il est loin d'être insignifiant.

Dans la deuxième partie de cette thèse, nous proposons une évaluation à grande échelle des vocodeurs neuronaux pervasifs, qui sont une classe de modèles génératifs utilisés pour la génération de la parole, conditionnée par le mel-spectrogramme. Nous introduisons une analyse multi-objectifs basée sur l'optimalité de Pareto à la fois de la qualité de l'évaluation humaine et de la consommation d'énergie. Dans ce cadre, nous montrons que des modèles plus légers peuvent être plus performants que des modèles plus coûteux. En proposant de s'appuyer sur une

nouvelle définition de l'efficacité, nous entendons fournir aux praticiens une base de décision pour choisir le meilleur modèle en fonction de leurs exigences.

Dans la dernière partie de la thèse, nous proposons une méthode pour réduire les coûts associés à l'inférence des modèle génératif profonds, basée sur la quantification des réseaux de neurones. Nous montrons un gain notable sur la taille des modèles et donnons des pistes pour l'utilisation future de ces modèles dans des systèmes embarqués.

En somme, nous fournissons des clés pour mieux comprendre l'impact des modèles génératifs profonds pour la synthèse audio ainsi qu'un nouveau cadre pour développer des modèles tout en tenant compte de leur impact environnemental. Nous espérons que ce travail permettra de sensibiliser les chercheurs à la nécessité d'étudier des modèles efficaces sur le plan énergétique tout en garantissant une qualité audio élevée.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Philippe Esling, who believed in me from the very beginning of this journey. Without his friendly guidance and dedication, none of my achievements over the past three years would have been possible. I am also deeply grateful for his invaluable advice and expertise that allowed me to improve my competencies in many fields.

I also want to thank Jean-Pierre Briot for his reliable direction and support. His constructive feedback and encouragement were crucial in helping me complete this thesis. I would also like to extend my appreciation to the entire ACIDS team, Axel Chemla-Romeu-Santos, Antoine Caillon, Giovani Bindi, Ninon Devis, Tristan Carsault, Mathieu Prang, Adrien Bitton, David Genova, Nils Demerle and Sarah Nabi. It has been an honor to be a part of such a talented and supportive group.

I take this opportunity to thanks all the friends I made at Ircam; this journey would have been totally different without their kindness and all those laughter, which have been a constant source of strength for me. Special mentions go to Claire, who helped me find the deepest motivation to write this manuscript alongside hers, to Victor for his thorough reviewing and graphic advisor with his keen eye, to Yann for our non-verbal communication skills that helped me more than once, to Vincent, always at the forefront of all the stories and adventures that happened to me, to Paul that have been the best office partner anyone could ask for, and finally to Pablo, who always found a way to help me at the most challenging times.

Last but not least, I want to thank my family for their unconditional love and carry support. They have always believed in me and helped shape me into the person I am today. For my mother, who has always pushed me and is the reason I managed to go that far in my studies. For my two older sisters, who have been a constant source of inspiration and motivation from the beginning of my childhood to my adult growth.

Thank you all for everything.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
NN	Neural Networks
RNN	Recurrent Neural Networks
LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Network
AE	Autoencoders
VAE	Variational Autoencoders
VQ-VAE	Vector Quantized-Variational AutoEncoder
GD	Gradient Descent
SGD	Stochastic Gradient Descent
MSGD	Mini-Batch Gradient Descent
ReLU	Rectified Linear Unit
MSE	Mean Squared Error
KL	Kullback-Leibler
AR	Auto-regressive
ELBO	Evidence Lower Bound
NF	Normalizing Flows
GAN	Generative Adversarial Networks
DM	Diffusion-based models
VI	Variational Inference
AI	Artificial Intelligence
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
IS	Inception Score
MS	Mode Score

AM Activation-Maximization
FID Fréchet Inception Distance
FAD Fréchet Audio Distance
NDB Number of Statistically-Different Bins
MOS Mean-Opinion Score
FLOP Floating-Points Operations
MIPS Million Instructions per Second
TTS Text-To-Speech
STFT Short-time Fourier Transform
SNR Signal-to-Noise Ratio
RTF Real-time Factor
SDR Signal-to-Distortion Ratio
SIR Signal-to-Interference Ratio
SAR Signal-to-Artifact Ratio
PESQ Perceptual Evaluation of Speech Quality
PUE Power Usage Effectiveness
QAT Quantization Aware Training

1

INTRODUCTION AND MOTIVATIONS

1.1 THE DIGITAL ANTHROPOCENE

The *anthropocene* is a concept that has gained increasing attention in recent years, particularly among scientists and environmentalists. It refers to a new geological epoch, in which the influence of human activity on the Earth global ecosystem has become the dominant source of impact. The inception point of this major shift is traced to the 19^{th} century industrial revolution, when humans began to harness and consume vast amounts of energy and resources on a global scale. Since then, the impact of human activities on the planet has been far-reaching, including the alteration of land, water, and air resources (Steffen et al., 2015). The widespread adoption of industrial technologies and practices has profoundly transformed our daily lives and our interactions with the world that surrounds us. One of the earliest prediction regarding the potential consequences of this increased human impact on the planet came from McGinnis et al., 1973 with "*The Limits of Growth*", where they argued that economic and population growth would lead to a collapse of global systems. Although those predictions have been disputed since, it is now clear that the increasing impact of human activities on the planet has raised serious concerns about the long-term sustainability of our current way of life.

More recently, the proliferation of digital devices has further transformed these intricate relationships between humans and their surroundings. Today, digital technologies are ubiquitous in our daily lives, from the smartphones we use to communicate and access information, to the computers and other devices we use in our homes and workplaces. The widespread adoption of these technologies has been driven by the decreasing cost of silicon, a key component in many digital devices. Furthermore, advances in semiconductor manufacturing have allowed to pack as many as 1.5 billion transistors onto a single microchip, enabling the development of increasingly powerful and sophisticated devices. Nonetheless, the global shortage of computer chips in 2021 has brought attention to the issue of our limited world resources. It has been partly driven by the high demand due to the pandemic situation, but also by an unprecedented drought in Taiwan, the world leading producer of chips. These events raise concerns about the sustainability of our current technological practices and the potential for future environmental disaster caused by climate change (Krinner et al., 2013).

Climate change is a major global challenge that is caused, in part, by human activities such as burning fossil fuels and deforestation. These activities release large amounts of greenhouse gases into the atmosphere, trapping heat and contributing

to global warming. The concentration of carbon dioxide (one of the major greenhouse gases) in the atmosphere has significantly increased over the past century Soon et al., 1999, as shown in Figure 1. This situation is expected to worsen unless drastic actions are taken to reduce emissions. The *Paris Agreement*, adopted at the COP21 conference in 2015, aims precisely to address climate change by establishing a global framework to limit temperature increases and greenhouse gas emissions, including the overarching goal of achieving zero net emissions by 2050.

To achieve this goal, despite the ever-growing use of digital technologies, it is crucial to account for the technological carbon footprint, from the production of devices to their daily use. Digital technologies are a significant contributor currently representing around 6% of global greenhouse gas emissions. However, this number is expected to increase as the use of digital technologies continues to grow. One factor contributing to this trend is the so-called "*rebound effect*," in which the use of more energy-efficient technologies leads to an increase in overall energy consumption rather than a reduction. This illustrates the need for more sustainable approaches to the production and use of these technologies, in order to minimize their impact on the planet and its resources. Moreover, the increasing use of Artificial Intelligence (AI) algorithms in the digital world, mostly based on ML techniques, require large amounts of data and computing power to produce relevant results. This can result in significant energy consumption and greenhouse gas emissions, which we discuss in the following sections.

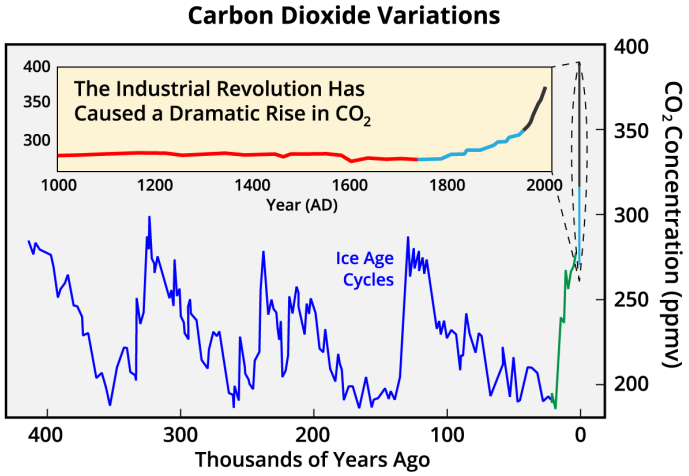


Figure 1: 400,000 years records of past atmospheric carbon dioxide levels as reconstructed from the trapped bubbles of gas in four ice cores records from Antarctica. Figure taken from the *Global Warming Art*, Rohde, 2005.

1.2 COMPUTATIONAL TRENDS AND IMPACTS OF AI

1.2.1 *History of AI technologies*

The concept of AI was born in 1950, when Alan Turing wondered whether a machine could think in his article "*Computing Machinery and Intelligence*" (Turing, 1950). Since then, AI refers to the development of computer systems that are able to perform tasks that normally require human intelligence, such as recognizing patterns, learning from experience, and making decisions. One of the earliest experiments in the development of AI was the creation of the *perceptron* by Rosenblatt, 1957. This machine was able to recognize simple patterns and classify them based on their characteristics. This architecture laid the foundation for the current trend of Machine Learning (ML) algorithms. Later on, the concept of *back-propagation* was introduced by Rumelhart, Hinton, and Williams, 1986, which allowed neural networks to learn from their mistakes and improve their accuracy by adjusting the weights of the connections between neurons. This marked a significant step forward in the development of AI, as it allowed machines to learn and adapt to new data in a more efficient way.

Another important development in the field of AI was the discovery of receptive fields in the cat visual cortex by Hubel and Wiesel, 1959. This work provided insight into how the brain processes visual information and influenced the development of artificial neural networks. Then, Fukushima, 1980 introduced the NeoCognitron, which was one of the first artificial neural networks to recognize complex patterns and perform image recognition tasks. Later, Lecun et al., 1998 developed *LeNet*, which was one of the first successful applications of artificial neural networks for image recognition. However, one core limitation to further studies in that field came from the lack of data and computational power available at that time. The success of AlexNet (Krizhevsky, Sutskever, and Hinton, 2012), as well as the development of Deep Learning (DL) algorithms such as *AlphaGo* by Silver et al., 2016, which was able to beat human champions at the complex board game Go, led the way for the widespread adoption of deep learning in a large variety of applications.

1.2.2 *The pursuit of accuracy*

Nowadays, DL has become a common tool in many scientific discipline, with most notable applications in computer vision, natural language processing, and even healthcare. This rapid progress has been driven by the never-ending quest for improved quality and performance in DL systems, as researchers and practitioners seek to develop more accurate and efficient models that can solve increasingly complex tasks. One key factor in this progress has been the exponential increase in computational power, with the overall computation rate increasing by a factor of 300,000 from 2012 to 2018 (Lohn and Musser, 2022) (see Figure 2). This increase

has been made possible by the development of more efficient hardware, such as Graphics Processing Unit (GPU)s and Tensor Processing Unit (TPU)s, which are specifically designed for DL tasks. These hardware accelerators have significantly improved the speed and efficiency of DL algorithms, allowing them to process and analyze increasingly large amounts of data.

However, even with these advances, the demand for computational power continues to grow (Thompson et al., 2020), driven in part by the rebound effect, which dictates that as the efficiency of technologies increases, their overall use tends to increase as well. This means that even as hardware becomes more efficient, the demand for data and computation continues to grow. As a result, data centers, which are used to store and process the vast amounts of data required by DL algorithms, have become increasingly common, and the electricity and cooling requirements of these centers have also grown significantly. As such, it is important to consider the full life cycle of these technologies, including their environmental and carbon footprint (Gupta et al., 2022; Strubell, Ganesh, and McCallum, 2020a). Despite major advances in manufacturing energy-efficient hardware, the computational cost of DL remains humongous and continuously rising (Sevilla et al., 2022). Although part of the current research effort is concerned with the true cost of deep models (Henderson et al., 2020), taking into account the environmental impact of these models is mostly overlooked against this never-ending quest for accuracy. Furthermore, as computation is frequently performed on remote servers such as cloud services, researchers may lose sight of the actual energetic footprint of their work. For these reasons, the domain of *green computing* (Schwartz et al., 2020) attempts to raise awareness on these questions, trying to provide qualitative and quantitative measures of the actual consumption of designing, training, and using deep models.

Generally speaking, there are two phases in the development of DL models: *training* and *inference*. During the *training* phase, a model is presented with a large dataset and uses it to learn the relationships between the input data and the desired output by adjusting the weights of the connections between neurons. This process is repeated multiple times, with the model learning from its mistakes and improving its accuracy at every iteration. The training phase can be time-consuming, as it requires large amounts of data and computational resources. Once the model is trained, it can be used for inference, which is the process of applying the model to new data in order to make predictions. This phase can be done on multiple type of devices including GPUs, CPUs, and embedded systems, depending on the requirements of the task it has been conceived for.

1.2.3 *Generative models*

More recently, deep *generative models* have been introduced, which are capable of generating new data based on a given input. These models have the potential

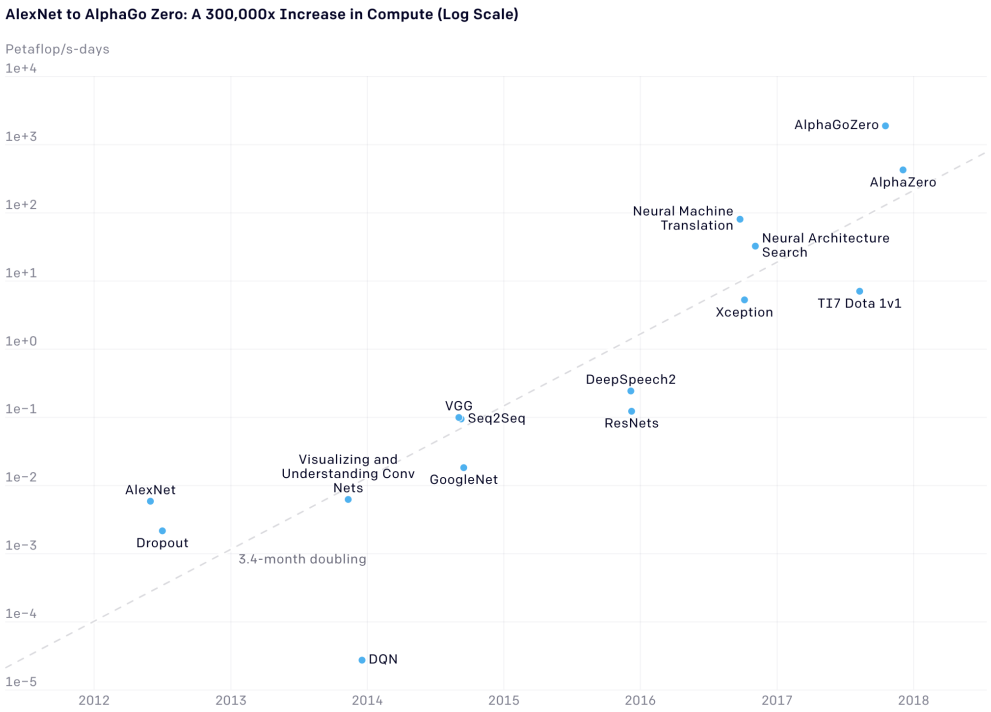


Figure 2: AlexNet to AlphaGo Zero: A 300,000x increase in computation resources. Figure taken from Lohn and Musser, 2022.

to revolutionize many fields, such as art, music, and Natural Language Processing (NLP), by allowing machines to create new and unique artifacts. These are powerful tools becoming increasingly accessible to the general public. Among them, the recent *GPT-3* model by Brown et al., 2020, a language generation model, is able to produce all kinds of texts and even code, and has received a lot of attention in the media due to its impressive performances on a wide range of tasks and its potential to revolutionize the field of natural language processing. Here his an example of the output of chatGPT[1], fed by our input : "Write a poem about deep learning destroying the planet".

> *Deep learning, a force so strong*
> *It's destroying our planet, it won't be long*
> *Artificial intelligence, a threat we must face*
> *But do we have the power to change our fate?*
>
> *We thought we were clever, we thought we were wise*
> *But now we see the error of our ways, we realize*
> *We thought we could control it, but it's too late*

1 https://chat.openai.com/

Deep learning has become our ultimate fate

It's taken over the world, it's all we see
We can't escape it, it's constantly with me
It's changing the way we live, it's changing the way we think
It's tearing us apart, it's causing us to shrink

We've become reliant on it, we can't live without
But at what cost? Our planet's on the brink of drought
We've destroyed the environment, we've polluted the air
We're killing ourselves, we don't even care

But it's not too late, we can still turn it around
We can still save our planet, we can still be profound
We can unplug from deep learning, we can find another way
We can rebuild our world, we can make a brand new day

So let's take a stand, let's make a change
Let's not let deep learning destroy the earth, let's rearrange
Our priorities, our values, and our goals
We can create a brighter future, let's save our home

In the area of computer vision, generative models have also yield impressive results, and are now able to generate a wide variety of high-resolution images. They have been used in several applications, such as creating realistic images for movies and video games. One of the most notable model is the *DALL·E 2* model from Ramesh et al., 2022, that is capable of generating images from textual descriptions (see Figure 3).

While generative models achieve impressive results, they usually come with a significant computational cost. This is due to the high dimensionality of the output generated by these models, which requires a large amount of processing power and energy to generate. For instance, training a model like GPT-3 has been estimated to require the amount of energy equivalent to traveling 703,808km by car (Anthony, Kanding, and Selvan, 2020), which is about twice the distance between Earth and the Moon. Inference, or using these trained models to generate new outputs, also requires a significant amount of computational resources. According to OpenAI[2], more than 3 million people are already using DALL·E to extend their creativity and speed up their workflows, generating over 4 million images a day.

2 https://openai.com/api/

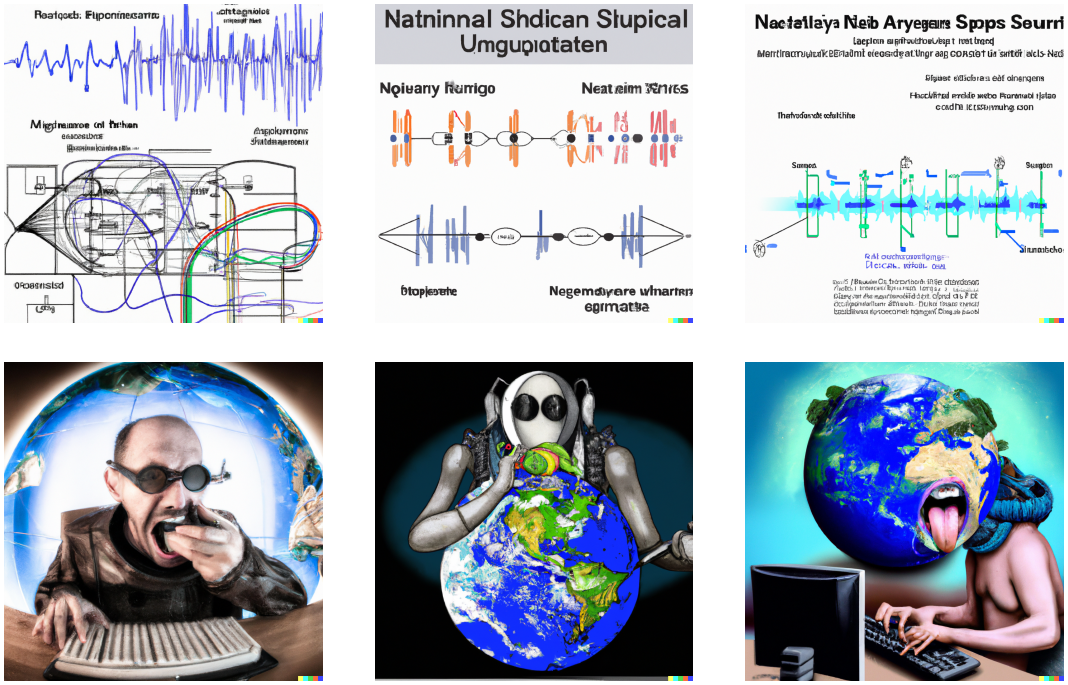


Figure 3: Examples from several outputs from DALL·E 2 (Ramesh et al., 2022), where the first raw corresponds to the input text description "Neural audio synthesis", while the bottom row depicts "A synthesizer eating the planet".

Deep generative models have also achieved impressive results in the domain of audio synthesis, specifically for waveform synthesis. These models are now routinely used for speech synthesis in virtual assistants such as Apple Siri and Amazon Alexa, while their use is consistently increasing in the context of music generation. However, researchers are currently focusing on improving the quality of real-time raw waveform synthesis, which requires a significant amount of computational power as waveform is a high-dimensional input data. In the following section, we expose the process of audio synthesis and explore how deep learning techniques can be applied to it.

1.3 SOUND SYNTHESIS

1.3.1 *Music representation*

Music can be represented in multiple ways. The most well-known example of a musical representation is the *musical score* that evolved for several hundred years, and still provides the current ground for musical notation. Indeed, it has been the most reliable and efficient way of transmitting western music, mostly based on pitch and note duration information. However, in the 20th century, a lot of research has been carried out on new ways of representing music, where technological

improvements encouraged the development of new notations. In this section, we introduce the three most common representations and discuss their interests and limitations for audio generation.

SYMBOLIC REPRESENTATIONS One common way to represent music is through symbolic formats such as music sheets (see Figure 4a) or MIDI data. These formats are useful because they are lightweight and compressed, making them easy to store and manipulate. In these formats, each element of a musical piece is transcribed as rhythm information, notes, and instrumentation. However, there is one major drawback to using symbolic representations for music generation: they require one or more interpreters to convert the symbolic information into actual sound and a lot of information (timbre, fine-grained temporal organization) is lost.

TIME-FREQUENCY REPRESENTATIONS Another way to represent audio is to use time-frequency representations such as spectrograms (see Figure 4c) or mel-spectrograms (see Figure 4d). These representations are useful because they provide lossless compression of audio data, while making it easier to understand and manipulate. However, one drawback of using time-frequency representations is that they typically involve complex numbers, which can be difficult for standard machine learning models. As a result, many approaches to audio synthesis using time-frequency representations only use the real part of the representation, requiring an additional step to generate the complex part of the signal (known as the phase). Signal processing methods such as the Griffin-Lim algorithm (Griffin and Lim, 1984) have been used to approximate the phase, but this can be a tedious and time-consuming process. Modern optimization techniques have made it easier to work with complex numbers in machine learning models, but this remains an active area of research in the field of audio synthesis.

RAW WAVEFORM The most accurate representation of audio information is the pressure waveform, which is the physical manifestation of sound as a series of pressure fluctuations over time (see Figure 4b). This is the usually way in which audio data is stored in basic file formats such as .wav files. In the past, classic methods for directly synthesizing waveforms have typically relied on recombining pre-recorded chunks of audio data. Though, the dimensionality of audio time series is dauntingly high. First, raw waveform data, which is usually recorded at a sample rate of 44100 samples per second. Second, each audio sample has to be encoded as 16-bit or 24-bit integers to perceptually represent the dynamical range of the signal.

In this work, we will focus on the waveform representation for audio synthesis. Although spectrograms were most commonly used to perform audio analysis and synthesis, they still lack performance when applied to real-time synthesis especially due to phase reconstruction issues. Thus, working directly on waveform

generation could improve the quality of generation results while removing any form of post-processing.

1.3.2 *Synthesis technologies*

Audio synthesis has been a field of interest for over a century now, opening interesting avenues for both musicians and scientists alike (Briot, Hadjeres, and Pachet, 2019). It can be defined as the process of generating sound using analog or digital devices. Researches in this field were mainly motivated by the will to expand the amount of freedom and exploration in sounds and creative expression. The tremendous success of synthesizers in the late '60s shaped the sound of new generations so much that synthesizers are among the most widely used instruments in nowadays modern musical production. Moreover, continuous technological improvements in terms of computational resources and descending costs of computer technology made synthesizers affordable to the general public, which greatly stimulated researches in synthesis techniques. Here, we propose a simplified classification of the most noticeable audio synthesis approaches, relying on the classification proposed by Smith III, 1991.

SPECTRAL MODEL The spectral model, developed by Joseph Fourier in 1807, is a way of representing a sound pressure wave as a sum of sinusoids. This is done by decomposing the sound wave into its individual components, each with its own amplitude and phase over time. This process is known as *harmonic analysis*. By adding together sinusoids or pure tones, it is possible to generate a wide range of sounds through the process of *additive synthesis*. On the other hand, *subtractive* synthesis involves starting with a spectrally rich sound, such as broadband noise, and removing specific frequencies to shape the sound spectrum.

PHYSICAL MODEL The acoustic modeling of musical instruments allows both the analysis and understanding of existing instruments, but also the numerical simulation of their sound generation. For example, one could model how the strings of a guitar are vibrating, and how this movement is coupled with those of the neck and the soundboard. An example of application is the modal methods (Eckel, 1995), which aims to describe instruments as a group of oscillators with various resonance frequencies.

ABSTRACT ALGORITHM Abstract models are computational methods that are not based on replicating existing physical phenomena. An example of abstract model is the Frequency Modulation (FM) synthesis, which was introduced by Chowning, 1973. FM synthesis involves the use of chained oscillators, or sinusoidal wave generators, to create sound. Unlike spectral models, FM synthesis relies on modulating the frequency of an oscillator using the output of another oscillator. This allows for the creation of a wide range of sounds, including complex timbres and harmonically-rich tones.

SAMPLING Finally, the class of processed recording involves analyzing recorded sounds in order to use them in the synthesis process. The most basic example of a waveform-based method is *sampling-based synthesis*, which involves using pre-recorded sounds and processing them to create new sounds. Concatenative synthesis and granular synthesis (Roads, 1978) are more advanced waveform-based methods that build upon the principles of sampling. Concatenative synthesis involves slicing recorded sounds into smaller segments, or samples, and recombining them to create new sounds. Granular synthesis is similar, but it involves slicing the samples into even smaller segments called "grains" and recombining them based on transition models.

1.4 DEEP AUDIO SYNTHESIS

1.4.1 *Overview*

As previously discussed, deep generative models have recently made significant progress in waveform synthesis. Hence, they have become a powerful tool for synthesizing high-quality speech and music. There are two main types of deep generative models for synthesis: *unconditional* models (i.e. autonomous) or *conditional* models based on extraneous inputs (e.g. text description, style or lyrics). In speech synthesis, deep generative models are used to create natural-sounding voices for virtual assistants or other applications[3]. These belong to the field of Text-To-Speech (TTS) techniques, which have been a major focus in recent years (Ning et al., 2019). Besides TTS techniques, deep learning is now used for voice conversion with the ability to alter the pitch, speed, and other characteristics of a voice (Chen et al., 2014). A famous example is audio deep fakes, where a person's voice can be manipulated to say something that they did not actually say.

Generative models are also used to synthesize music, either by creating entirely new compositions or by manipulating existing audio to create novel sounds (Hernandez-Olivan, Hernandez-Olivan, and Beltran, 2022). In addition to synthesizing waveforms from scratch, deep generative models can also be used for tasks such as *timbre transfer*, which involves transferring the timbre of one waveform to another, and *timbre interpolation* (Esling, Chemla–Romeu-Santos, and Bitton, 2018), which involves creating a new waveform that is a blend of two or more existing waveforms. Another application of deep generative models is inpainting, which involves filling in missing or corrupted parts of a waveform (Bazin and Hadjeres, 2019). Generative models also have the potential of creating new interactions with instruments while adding more tools for musicians to produce music, such as the automatic search of synthesizer parameters (Esling et al., 2020b) or for pure drum sound synthesis (Aouameur, Esling, and Hadjeres, 2019) and even effects modeling (Ramírez and Reiss, 2019). Deep learning has also led to successful approaches

3 https://uberduck.ai/

to produce raw audio from MIDI data (Renault, Mignot, and Roebel, 2022) which has the potential to save time and resources and also allows for more experimentation and iteration in the composition process. Overall, the applications of deep generative models for audio synthesis are blooming and gradually getting into the hands of end users, even on mobile devices. Hence, the potential energetic impact of using those technologies is a significant issue for the future and will be the focus of this work.

1.4.2 *Challenges*

One of the main limitations of neural audio synthesis is the high dimensionality of the output data, which requires the use of complex and resource-intensive neural network architectures. Moreover, the disparity of proposed models in the literature and the training time needed for them to converge questions the real relationships and effectiveness of different methods. Hence, this raises a critical question of finding the best compromise between energetic and environmental concerns and the quality of the generated results. Moreover, research institutes and individuals can lack sufficient resources, due to the demand of countless types of specialized hardware (GPUs, TPUs), often running continuously for several days and even up to weeks. Hence, obtaining a quality similar to that of state-of-the-art models is becoming an unattainable goal, both financially and ecologically.

1.5 OUTLINE

Currently, the evaluation of deep models is almost exclusively focused on the quality of the generated audio without consideration for the energy consumption of such models. This is a significant limitation, as the energy consumption of a model has important implications for its deployment in real-world applications. Thus, this thesis aims to propose new evaluation procedures for deep generative models for audio that take into account the models' energy footprint. By integrating energy consumption into the evaluation procedure, this thesis aims to enable a more comprehensive assessment of deep generative models for audio.

First, Chapter 2 presents the essential background theory in machine learning used throughout this thesis. We introduce the core concepts of machine learning techniques in Section 2.1 with a focus on deep learning and neural networks in Section 2.2. Then, we detail different variants of generative models and their respective properties in Section 2.3. Finally, we review a large set of generative models specifically designed for audio and waveform generation in Section 2.4.

In Chapter 3, building on the theoretical background presented, we expose a wide range of evaluation metrics for deep generative models. We address quality metrics in Section 3.1 as well as efficiency measures in Section 3.2. We conclude

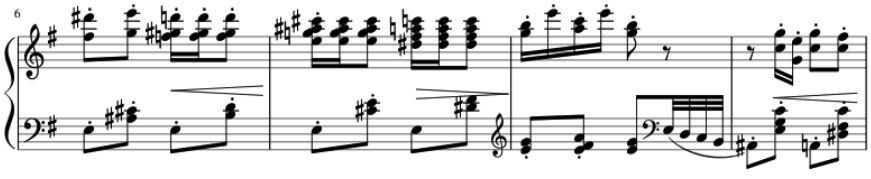
by introducing the concept of Pareto optimality for multi-objective evaluation in Section 3.3, which is central to this thesis.

In Chapter 4, we expose our first contribution by first presenting a review of the use of evaluation metrics in the literature in Section 4.1. Then, we raise the lack of energy footprint measurements and provide a first estimation for training state-of-the-art models in Section 4.2. Additionally, we propose a new methodology based on Pareto optimality to include the costs alongside the quality of the generated sound in Section 4.3.

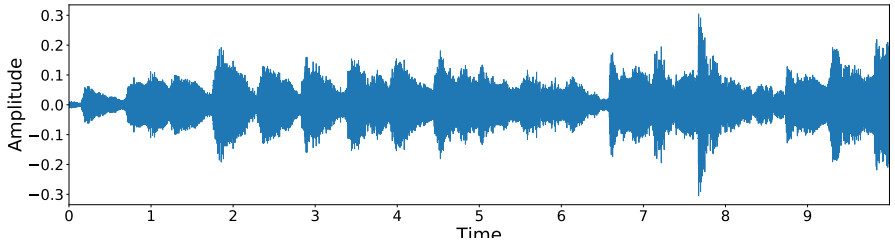
In Chapter 5, we conduct a broad benchmark of neural vocoders to understand the current trends and impact of deep audio models in Section 5.1. We use our methodology to compare the generation quality and energy efficiency of many models and configurations in Section 5.2. Within this framework, we show that this energy footprint must be linked to the model perceptual quality and that, small models can perform better than larger and more costly models.

In Chapter 6, we expose a first attempt to reduce the inference cost of neural vocoders through quantization techniques. We motivate this technique in Section 6.1 and expose the formalism in Section 6.2. Then, we run experiments on a neural vocoder benchmark with a particular focus on quantization-aware training in Section 6.3 and discuss the embeddability of these systems for audio synthesis in Section 6.4.

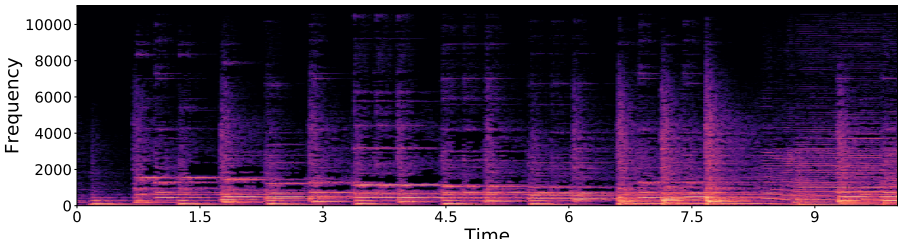
Finally, in Chapter 7 we present our conclusions and avenues and directions for future work.



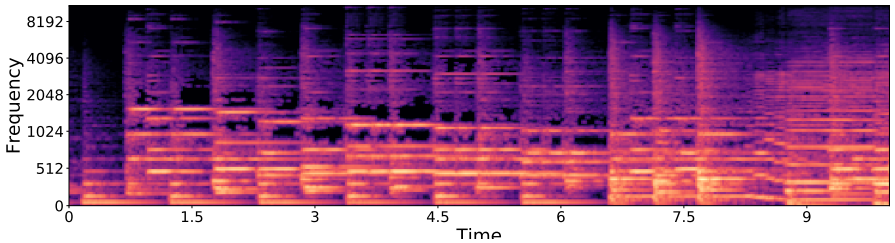
(a) Music sheet



(b) Waveform



(c) Spectrogram (dB)



(d) Mel-Spectrogram (dB)

Figure 4: Standard representations used in music generation, computed from the *Dance of the Sugar Plum Fairy* by Tchaikovsky.

Part I

BACKGROUND

2

BACKGROUND OF MACHINE LEARNING

In this chapter, we discuss the essential background and theory used throughout this thesis. First, we introduce the core concepts of machine learning (Section 2.1) with a focus on deep learning and neural networks (Section 2.2). Then, we describe the different families of neural generative models proposed in the literature (Section 2.3). Finally, we present the major generative models proposed specifically for audio and waveform generation. (Section 2.4).

2.1 GENERAL APPROACH OF MACHINE LEARNING

2.1.1 *Basic definitions*

Let us consider two sets $\mathcal{X}$ and $\mathcal{Y}$ linked by a complex relationship (e.g. a name to an object, or a cause to a consequence). Most of the time, we do not have access to the mathematical form of this relationship, but rather to the observation of its elements through the pair (x, y), where $x \in \mathcal{X}$ is the input data, and $y \in \mathcal{Y}$ is its associated output. The goal of ML is to approximate this unknown relationship as a parametric function that maps inputs x to outputs y.

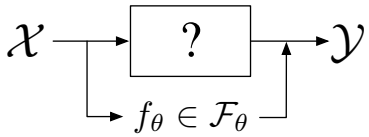


Figure 5: Machine learning concept.

As a result, machine learning consists in modeling the relationship between (usually complex and high-dimensional) inputs $\mathbf{x} \in \mathbb{R}^{\mathcal{X}}$ and outputs $\mathbf{y} \in \mathbb{R}^{\mathcal{Y}}$, by observing a set of data examples $\mathcal{D} = \{(x_1, y_1), \cdots, (x_N, y_N)\}$. Finding such a model is often done by searching for the best parametric function f_θ among a family of functions $\mathcal{F}_\theta$ in order to approximate the underlying relationship. The learning process consists in adjusting the parameters $\theta \in \Theta$, such that the output of our model $f_\theta(\mathbf{x}) = \hat{\mathbf{y}}$ closely matches the desired output $\mathbf{y}$.

LOSS FUNCTION To learn from input/output pairs, ML models need to define a suitable *loss function* $\mathcal{L}$ that measures the error between the predicted output $\hat{\mathbf{y}}$ and the target output $\mathbf{y}$. By minimizing $\mathcal{L}$, it reduces the distance between the parametric function f_θ and the true underlying function f, by finding the optimal value of the parameters θ that verifies

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{x}, \mathbf{y}, \theta, f_\theta)\ \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{D} \tag{1}$$

Defining a meaningful loss function is important because it determines how well the model is able to make predictions. A good loss function should be able to accurately penalize the model for incorrect predictions, while not overly penalizing it for predictions that are only slightly wrong. It is worth noting that choosing an appropriate loss function is problem-dependent and can have a significant impact on the model performances.

GRADIENT DESCENT The vast majority of ML algorithms rely on gradient-based optimization techniques to converge efficiently to local minima. Hence, the Gradient Descent (GD) and its variants are the most standard methods employed to optimize the model and find the lowest value of the loss $\mathcal{L}$. It consist in computing the gradient of the loss given the parameters $\nabla_\theta \mathcal{L}$, in order to iteratively adapt the value of the parameters through

$$\theta_{n+1} = \theta_n - \eta \sum_{\mathbf{x},\mathbf{y} \in \mathcal{D}} \frac{1}{|\mathcal{D}|} \nabla_\theta \mathcal{L}(\mathbf{x}, \mathbf{y}, \theta, f_\theta) \tag{2}$$

with η a meta-parameter, called the *learning rate*. This update rule can be interpreted as applying a slight change to the parameters value in the opposite direction of the gradient of the loss, with η being the magnitude taken in that step. As depicted in Figure 6, an adequate learning rate and a convex loss function leads to a fast and easy convergence.

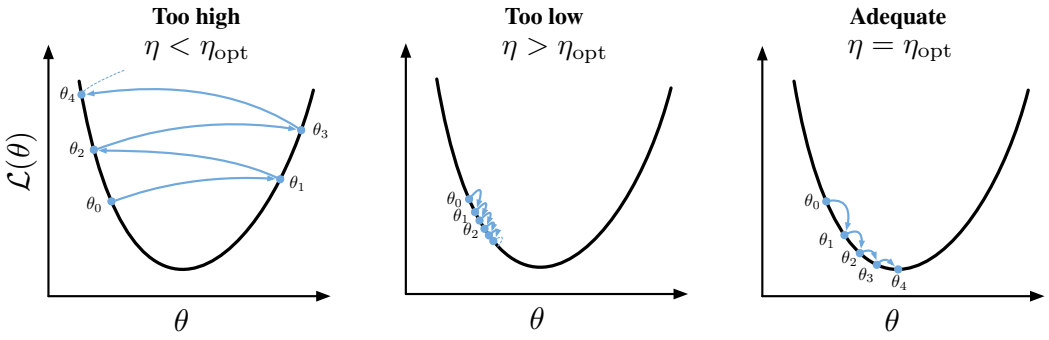


Figure 6: Effect of the learning rate on the convergence of gradient descent.

However, most of the learning processes usually correspond to non-convex functions, resulting in the convergence of the parameters towards a local minima. Moreover, computing the loss throughout the entire dataset can be particularly computationally expensive for large datasets.

To overcome this issue, the Stochastic Gradient Descent (SGD) computes the same derivatives but only considering a single element of the dataset at a time to apply the update rule. However, as the gradient may wodely differ from one example to another, it is preferable to consider subsets of the data rather than single examples. This process, called the Mini-Batch Gradient Descent (MSGD), allows to parallelize computation and thus speeds up the computation of the GD, while keeping the benefits of SGD. The subsets used are called *mini-batches* and a whole pass over all mini-batches of a dataset is called an *epoch*.

2.1.2 *Model capacity*

A key ingredient for the correct functioning of a ML algorithm is the choice of the family of parametric functions $\mathcal{F}_\theta$ from which the model is selected. For more clarity, we illustrate this concept in Figure 7 by taking the case of a regression task, where the parametric functions can be chosen inside the family of polynomials of degree p

$$\mathcal{F}_p : \{f_\theta(x) = \sum_{0 \leqslant i \leqslant p} a_i x^i, \quad \theta = \{a_i\}_{i \in [0,p]}\} \tag{3}$$

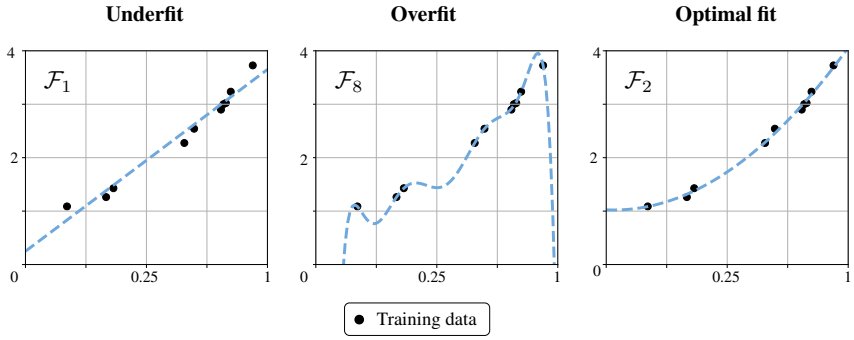


Figure 7: Three different parametric functions showing either underfitting, overfitting or good generalization for a regression task.

If the function is too simple (case $p = 1$), the model may *underfit*, meaning that the corresponding model is not complex enough to capture the underlying structure of the observations. Conversely, the model can also *overfit* (case $p = 8$), i.e., it fits all the training points rigorously, even if the observations include noise. In both cases, the model is unable to *generalize* to new input data. Therefore, the capacity of the model must be selected carefully in order to match with the inherent complexity of the input data (case $p = 2$). The selection of the model's capacity is critical when dealing with high-dimensional datasets with strong nonlinear relationships, as it is far from trivial to find a coherent complexity that can accurately represent the data. Over-parameterization is a common issue that can arise when the model's capacity is too large, causing it to fit the noise in the data rather than

the underlying patterns. On the other hand, under-parameterization can occur when the model's capacity is too small, causing it to miss important features in the data.

2.1.3 *Overfitting, underfitting, early stopping*

In machine learning, variance and bias are two important concepts that describe the behavior of a model. Variance refers to the degree to which the model's predictions vary based on the training data, while bias refers to the degree to which the model's predictions deviate from the true values. Overfitting occurs when a model has a high variance and low bias, meaning that it has learned highly specific patterns in the training data that do not generalize well to unseen data. This can happen if the model is trained for too many epochs, causing it to fit the training data too closely. On the other hand, underfitting occurs when a model has a high bias and low variance, meaning that it is unable to capture the complex patterns in the training data. This can happen if the model is not complex enough or if it is trained for too few epochs. Hence, to ensure that a model has good generalization performance, it is common to split the dataset into three disjoint subsets:

- the *training set* is used to update the parameters and train the model.
- the *validation set* measures the accuracy of the model on unseen data to prevent it from overfitting.
- the *test set* serves as a final evaluation of the performance of the model.

The *early stop criterion* is a technique employed to reduce overfitting. The training is halted once the validation loss stops improving, while the training loss continues to decrease, as illustrated on Figure 8.

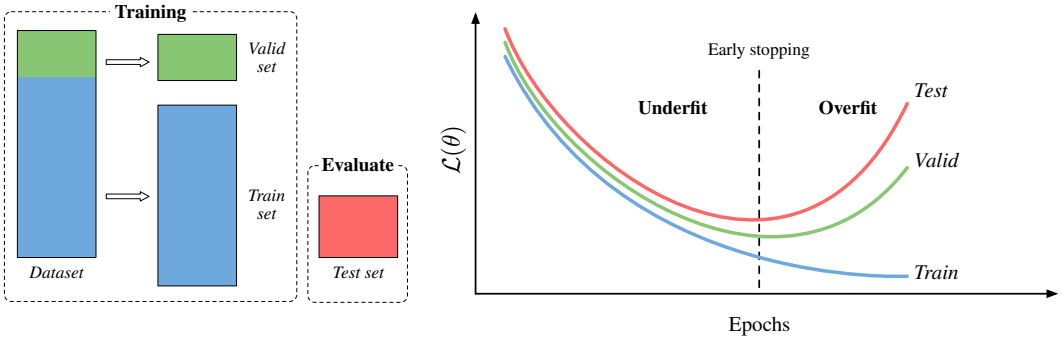


Figure 8: Separation of the dataset and typical evolution of the corresponding losses on each subset to avoid overfitting.

2.1.4 *Types of learning algorithm*

There are several types of learning algorithms that can be used depending on the nature of the task that we aim to solve.

SUPERVISED LEARNING The *supervised learning* approach seeks to predict a specific information based on paired data, the most representative examples being classification and regression tasks. Hence, supervised models rely on labeled training datasets, which can be expensive to build, since they require a large number of human annotations. The advantage of that learning mechanism is that there is usually a direct way to measure the accuracy of the models, as there is straightforward correspondence between input and output data.

UNSUPERVISED LEARNING In contrast, in the *unsupervised learning* approach, the model has no extraneous or paired information but rather only a whole set of raw data. Hence, models attempt to learn the underlying structures and patterns directly from the data. As a result, evaluation is usually forced to rely on indirect qualitative metrics. Clustering and generative tasks are two well-known examples of unsupervised learning, the latter being the subject of Section 2.3.

SEMI-SUPERVISED LEARNING *Semi-supervised learning* approaches are a combination of supervised and unsupervised approaches that usually leverages only a small amount of labeled data, while still relying on large amounts of unlabeled data. Hence, it can significantly enhance learning accuracy while using less labeled data. Nevertheless, these approaches can prove harder to define depending on the task at hand.

SELF-SUPERVISED LEARNING In *self-supervised learning* methods, the goal is to learn underlying structure directly from the data (similar to unsupervised learning), but by using a *proxy task* akin to supervised learning methods such as classification. One representative method for this is *contrastive* learning, which aims to learn meaningful representations of a dataset by comparing pairs of samples and predicting whether they are similar or dissimilar.

2.2 NEURAL NETWORKS

2.2.1 *Artificial neuron*

Neural Networks (NN) are a subfield of ML that automatically learn complex patterns from data by taking inspiration from biological neurons (see Figure 9 left). These are composed of dendrites connected to a cell body where each dendrite transmits an incoming electrical impulse. The cell body accumulates all of the impulses potential and sends an electrical impulse through its axon when the electri-

cal charge exceeds a threshold. Similarly, an *artificial neuron* receives a set of inputs $\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_n\}$ and outputs a value y called *activation* by computing

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + b) = \sigma(\mathbf{w}^T \cdot \mathbf{x} + b) \tag{4}$$

where $\mathbf{w} = \{w_1, w_2, w_3, \ldots, w_n\}$ is the set of parameters called *weights*, σ the threshold called the *activation function*, and b the *bias* term. The diagram of the perceptron and its analogy with the biological neuron is presented in Figure 9.

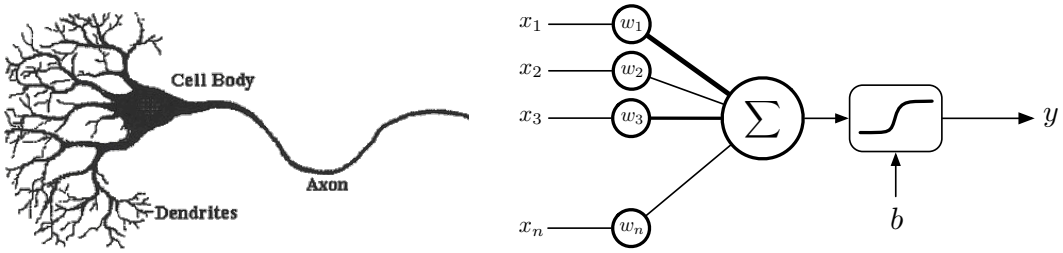


Figure 9: A biological neuron (left) approximated by an affine transformation and an activation function, called artificial neuron (right).

Usually, the functions used for the activation operator are

- the *sigmoïd*: $\sigma(x) = 1/(1 + e^{-x})$
- the *hyperbolic tangent*: $\sigma(x) = \tanh(x)$
- the *Rectified Linear Unit (ReLU)*: $\sigma(x) = \max(0, x)$

Each of those functions are represented on Figure 10. Through the linear combination of input values and the application of such functions, each neuron performs a non-linear transformation of its input.

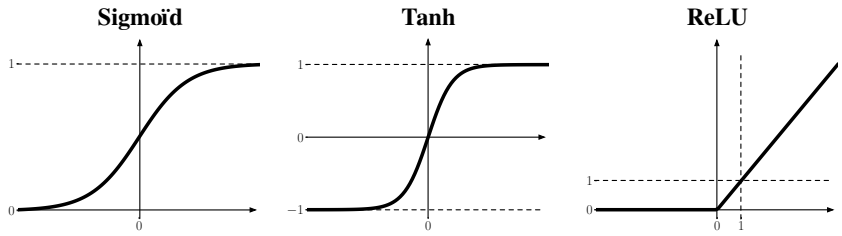


Figure 10: Representation of three popular activation functions.

2.2.2 *Multi-layer perceptron*

In order to model more complex and refined processes, a NN uses a composition of perceptrons arranged in layers. A standard structure is the linear or *fully-connected* layers, also called Multi-Layer Perceptron (MLP), where each adjacent

layer is densely connected but shares no connection within the same layer (Figure 11). The *input* vector $\mathbf{x}$ is the first layer, and the output vector $\hat{\mathbf{y}}$ is the last layer of the network, while layers in between are called *hidden layers*. The number of layers represents the *depth* of the network and the number of neurons its *size*.

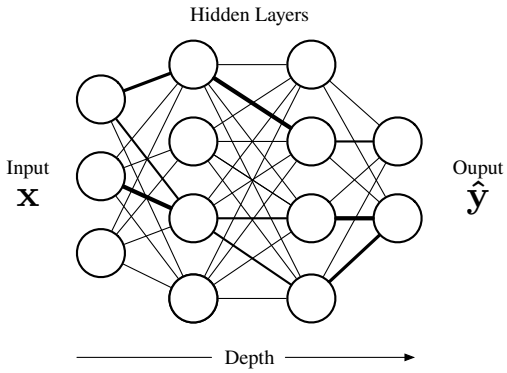


Figure 11: A multi-layer perceptron with two hidden layers.

Considering a network of depth L, we denote $h^l \in \mathbb{R}^{N_l}$ the value of the activation at the l-th layer of size N_l characterized by

$$h^l = \sigma(\mathbf{W}^l \cdot h^{l-1} + b^l) \tag{5}$$

with $\mathbf{W}^l \in \mathbb{R}^{N_l \times N_{l-1}}$ the corresponding weight matrix parameters and b^l the bias term, where $\hat{\mathbf{y}} = h^L$ and $\mathbf{x} = h^0$. By denoting $\theta_l = \{\mathbf{W}^l, b^l\}$ the layer parameters and $\theta = \bigcup_{l=1}^{L} \theta_l$ the parameters of this network, a NN can be defined as a parametric function, whose goal is to find the best approximation $\hat{\mathbf{y}} = f_\theta(\mathbf{x}) \approx \mathbf{y}$.

2.2.3 *Backpropagation*

The optimization of the model is done by updating its parameters to minimize the error $\mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})$. To do so, the *backpropagation* algorithm, introduced by Rumelhart, Hinton, and Williams, 1986, is based on the *chaine rule* of derivation, where each derivative is expressed as a function of the derivative of the previous layer

$$\frac{\partial h^l}{\partial \theta}(\theta) = \frac{\partial h^l}{\partial h^{l-1}}(h^{l-1}(\theta))\frac{\partial h^{l-1}}{\partial \theta}(\theta) \tag{6}$$

The gradient of the loss given the parameter θ_k from the k-th layer is thus

$$\frac{\partial \mathcal{L}}{\partial \theta}(\theta_k) = \prod_{r=k}^{L+1} \frac{\partial h^r}{\partial h^{r-1}}(h^{r-1}(\theta_k)) \tag{7}$$

The computation of the outputs is called the *forward pass* and the computation of the derivatives the *backward pass*. An illustration of backpropagation with a simple network is depicted in Figure 12.

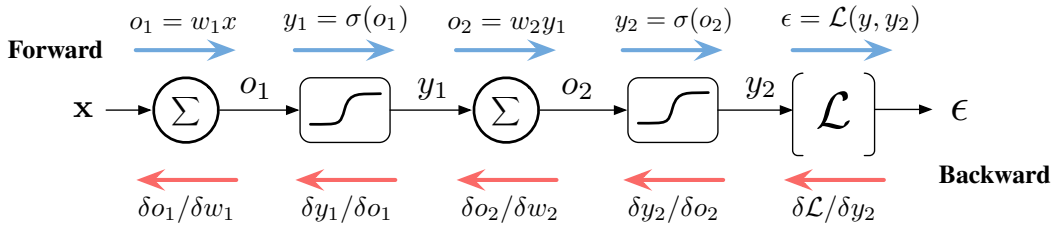


Figure 12: Example of the backpropagation algorithm for an extremely simple network with only one-dimensional data and no bias term.

2.2.4 *Recurrent Neural Networks*

Recurrent Neural Networks (RNN) are a particular type of NN aimed at modeling sequential data. In contrast with classic NN where all inputs and outputs are independent of each other, RNN have the ability of modeling correlations between previous computations thought *recurrent* connections. They have proven to be particularly efficient for learning temporal problems as they introduce a form of "memory" mechanism. In the following, we focus on a simple RNN containing a single, self-connected hidden layer, as shown in Figure 13.

Considering the sequence input $\mathbf{x} = \{x_1, x_2, \ldots, x_{t-1}, x_t, \ldots, x_T\}$ and h_t the hidden state of the network at time t. The network is fed with the previous hidden state h_{t-1} and the current input x_t. Each hidden state is updated as follows

$$h_t = \sigma(U \cdot x_t + W \cdot h_{t-1} + b_h) \tag{8}$$

where U is the weight matrix that connects input and hidden units, W is the weight matrix that connects hidden units together, and b_h the bias. Therefore, $\mathbf{h}$ can be seen as a memory, since it keeps information from previous steps.

Thanks to the memory $\mathbf{h}$, the system has a knowledge of the context and should be able to predict the next steps of a given sequence. However, these models do not necessarily adapt well to situations that require a longer context because of problems such as *vanishing* or *exploding gradients* (Bengio, Simard, and Frasconi, 1994). Therefore, RNNs are usually restricted to perform well on short-term dependencies. More advanced types of RNNs, like Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) tackle this issue and can capture longer-term dependencies by resetting or updating their memory content.

2.2.5 *Convolutional Neural Networks*

CNN are widely known for image, audio and time series processing, especially given their impressive results for classification and recognition tasks (LeCun and Bengio, 1998). They are composed of *convolutional layers*, where each layer consists

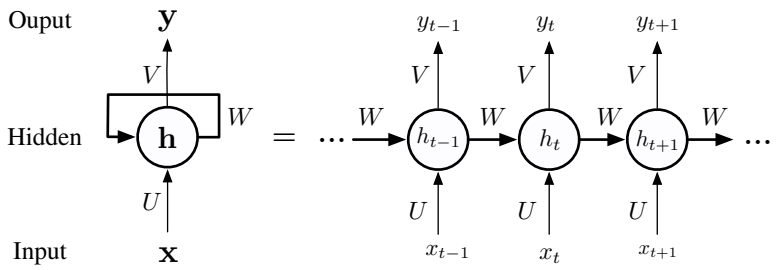


Figure 13: A Recurrent Neural Network and its unfolded version. At timestep t, the network is fed with both the input and the hidden state output at time $t-1$.

in a set of N *filters* called *kernels*, which are convolved across the input. The set of kernels of the l-th layer is denoted as $\{k_n^l\}_{n\in[1,N]}$, and when convolved across a d-dimensional input x, produces N d-dimensional outputs (one for each kernel) called *feature maps*, denoted as $\{a_n^l\}_{n\in[1,N]}$ and defined by

$$a_n^l = \sum_{m=1}^{M} k_n^l \star x_m + b_n^l \tag{9}$$

where $\star$ is the 2-dimensional convolution operator, b_n^l the biases, and M the number of input feature maps. The overall application of this operation is depicted on Figure 14. While learning in a convolutional layer consists in finding the optimal kernels, several parameters can be tuned when defining the overall architecture such as the size of the kernels, their number, the stride parameter (step size when sliding the kernel over the input) and the padding of both the input and the output (see Figure 15).

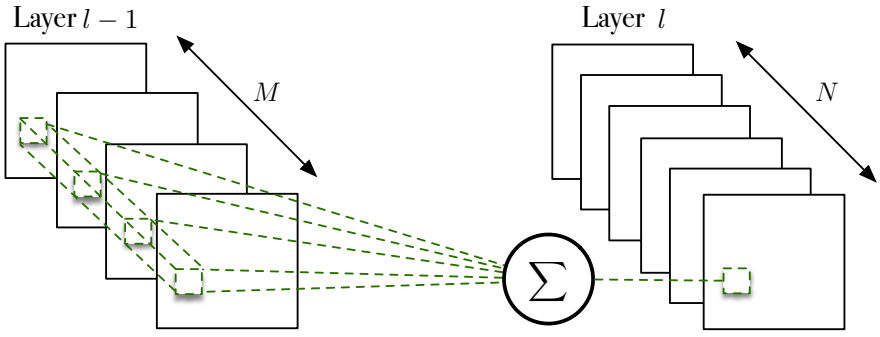


Figure 14: Convolutional layers inside a CNN. The kernels are convolved along M-th features maps of the layer $l-1$ and outputs N features maps.

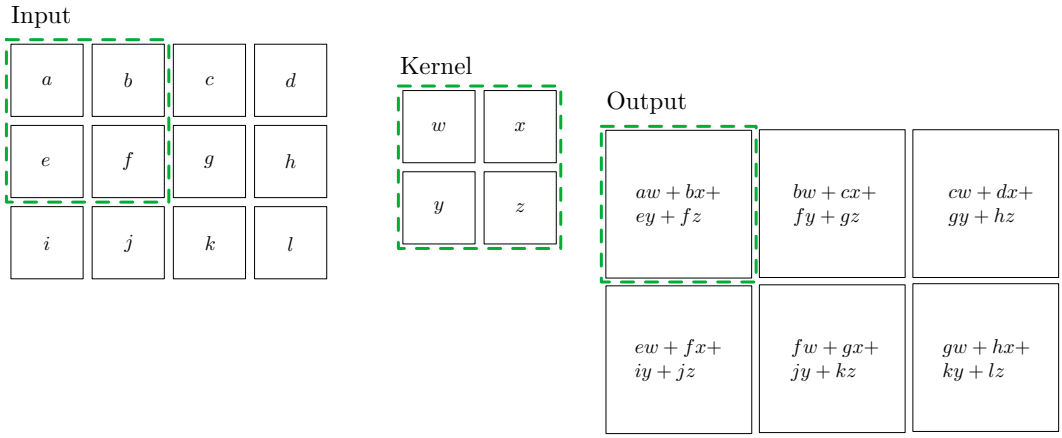


Figure 15: Schema of a 2D convolution where padding = 0 and stride = 1.

2.3 DEEP GENERATIVE MODELS

Generative models are a specific type of unsupervised learning techniques (Section 2.1.4) in which the goal is to generate novel data based on the observation of existing examples. Usually, they aim to model the unknown underlying data distribution $p_r(\mathbf{x})$. This is addressed by defining a parametric family $p_\theta(\mathbf{x})$ that best approximates the real one, so that examples from the dataset are highly probable, whereas irrelevant data are less probable. Hence, this can be summarized as the following optimization problem

$$\min_{\theta \in \Theta} d(p_r(\mathbf{x}), p_\theta(\mathbf{x})) \tag{10}$$

A commonly used objective function d to evaluate the quality of the model is the Kullback-Leibler (KL) divergence $\mathcal{D}_{KL}$, that measures the similarity between two distributions, generally defined by

$$\mathcal{D}_{KL}(p_r \parallel p_\theta) = \sum_x p_r(\mathbf{x}) \log\left(\frac{p_r(\mathbf{x})}{p_\theta(\mathbf{x})}\right) \tag{11}$$

Equivalently, minimising the KL divergence can be thought of as maximising the *log-likelihood* of data points with respect to the model distribution $p_\theta(\mathbf{x})$. Depending on the likelihood formulation and the density distribution expression, generative models can be separated into different large families, as proposed in Figure 16. Each of them are detailed below, with a focus on the architectures that are most frequently used for audio generation that are at the heart of our work.

2.3.1 *Auto-regressive models*

Auto-regressive (AR) models were initially developed for time series data, in order to predict the continuation of a sequence based on its first observations. It uses

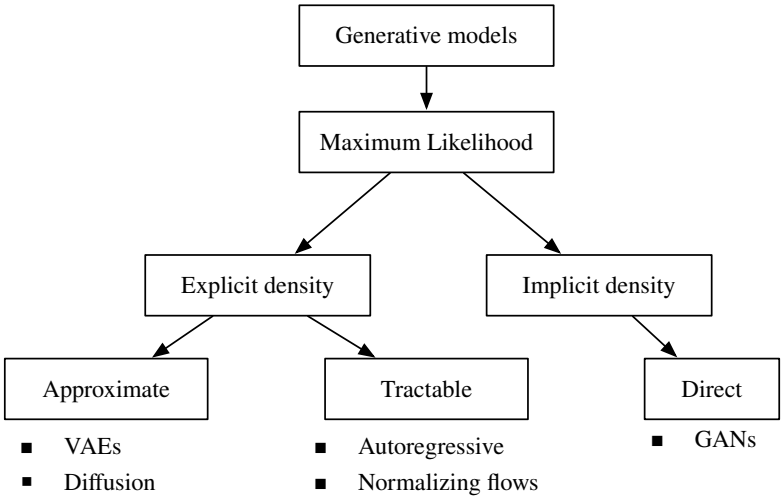


Figure 16: Organization of different generative models based on how the model addresses maximum likelihood and whether the density distribution is expressed explicitly or implicitly.

the chain rule of probability to decompose a distribution over an n-dimensional vector $\mathbf{x} = \{x_1, \cdots, x_n\}$ by assuming that the current time step only depends on the previous ones, so that the overall density can be *explicitly* expressed as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \cdots, x_{i-1}) \tag{12}$$

As a result, it is possible to directly maximise the likelihood of the data by training a neural network to model $p(x_i|x_1, ..., x_{i-1})$ by minimising the negative log-likelihood

$$-\log p(\mathbf{x}) = -\sum_{i=1}^{T} \log p(x_i|x_1, ..., x_{i-1}) \tag{13}$$

RNN are good candidates to define AR models, where each conditional probability can be modeled by $p(x_t \mid x_{<t}) = g(h_t)$. However, as exposed previously, these usually fail to model long-term dependencies. Hence, Mehri et al., 2017 proposed the *SampleRNN* model, which is designed to process sequential data at multiple temporal resolutions. The model is organized into *tiers*, with each tier working at a different level of temporal resolution. The receptive field of a tier is defined as the length of the sequence that the tier is able to see, and each tier is conditioned by the tier below it. This hierarchical structure allows SampleRNN to process long-term dependencies in the data and generate high-quality audio samples.

Unlike SampleRNN, the *WaveNet* model proposed by Oord et al., 2016 rely on CNN which, to some extent, can be assimilated to a form of recurrence. Hence,

the conditional probability distributions are modeled by multiple stacks of convolutional layers. Causal convolutions, as shown in Figure 17 (left), ensure that conditional probabilities $p(x_{i+1}|x_1, \ldots, x_i)$ emitted at time i are not dependent on future time-steps. Thanks to stacks of these convolutions, the model can have an increasing receptive field, which allows to model longer-term dependencies in the sequence. Nevertheless, increasing this receptive field incurs a concomitant increase of the number of layers and, therefore, of the computational cost. Instead, the key aspect of *WaveNet* is to use *dilated* convolutions to address this issue, as depicted in Figure 17 (right).

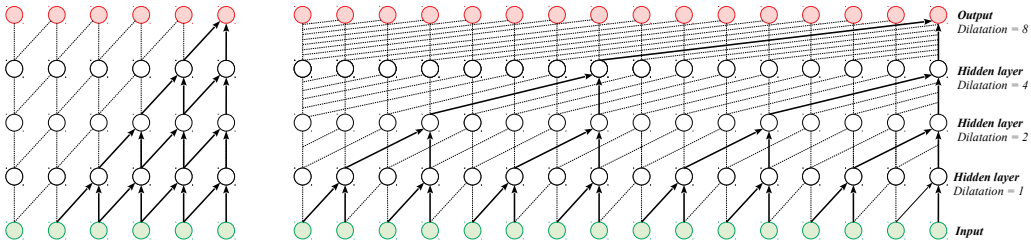


Figure 17: At the left side, the representation of a stack of causal convolution layers with 3 hidden layers and at the right side the same thing but with a a dilatation factor in 2^l where l is the number of the layer.

However, the big drawback of these architectures occurs at the inference (generation) phase, as the model must wait for each sample to be computed before predicting the next one, which implies a rather tedious process that prevents real-time generation. Nevertheless, AR models represent an important class of algorithms capable of generating high-quality audio of arbitrary duration and are still the subject of ongoing studies. Recent works are in fact moving in the direction of building more computationally-efficient architectures, such as the *SaShiMi* model proposed by Goel et al., 2022.

2.3.2 *Variational Auto-Encoder*

Variational Autoencoders (VAE) explicitly model the input density by relying on *latent variables*. It takes inspiration from the Autoencoders (AE) but extends them so that it can be used as a generative model using Variational Inference (VI).

AE were first introduced to compress data into a lower-dimensional space of the input data. As its name suggests, they aim to learn an efficient *encoding* $\mathbf{z}$, also called *latent code*, of unlabeled input data $\mathbf{x}$. A major way to learn an efficient encoding is to simultaneously learn a *decoding* function allowing to reconstruct $\mathbf{x}$ from $\mathbf{z}$. Hence, the architecture of an AE (Figure 18) is composed of an *encoder* that computes $\mathbf{z} = f_\phi(\mathbf{x})$ and a decoder, mirror of the encoder, that produces $\hat{\mathbf{x}} = g_\theta(\mathbf{z})$. The training of an AE consists in finding the optimal parameters (ϕ, θ) by evaluating the *reconstruction error* $\mathcal{L}$ between $\mathbf{x}$ and $\hat{\mathbf{x}}$.

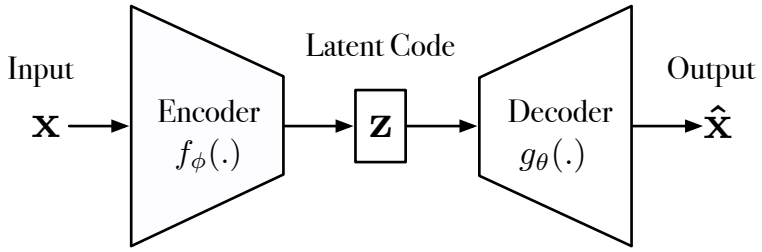


Figure 18: Architecture of an Auto-Encoder.

The relationship between data input $\mathbf{x}$ and latent encoding vector z can be fully defined by the prior distribution $p_\theta(\mathbf{z})$, the likelihood $p_\theta(\mathbf{x}\,|\,\mathbf{z})$ and the posterior distribution $p_\theta(\mathbf{z}\,|\,\mathbf{x})$. The full probability density can be expressed as

$$p(\mathbf{x}) = \int_z p(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z})dz \tag{14}$$

For most models, this integral is too hard to compute, as it is usually intractable to integrate over all possible values of the latent code z. The idea of VI is to solve this problem by assuming a simpler approximate distribution $q(\mathbf{z}\,|\,\mathbf{x}) \in \mathcal{Q}$ from a family $\mathcal{Q}$ of approximate densities. Hence, the goal is to minimize the difference between this approximation and the real distribution

$$\mathcal{D}_{KL}\left[q\left(\mathbf{z}\,|\,\mathbf{x}\right) \parallel p\left(\mathbf{z}\,|\,\mathbf{x}\right)\right] = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\left[\log(q(\mathbf{z}\,|\,\mathbf{x}) - \log(p(\mathbf{z}\,|\,\mathbf{x})\right] \tag{15}$$

The Bayes' rule allows us to replace $p(\mathbf{z}\,|\,\mathbf{x})$ and obtain

$$\mathcal{D}_{KL}\left[q(\mathbf{z}\,|\,\mathbf{x}) \parallel p(\mathbf{z}\,|\,\mathbf{x})\right] = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\left[\log q(\mathbf{z}\,|\,\mathbf{x}) - \log p(\mathbf{x}\,|\,\mathbf{z}) - \log p(\mathbf{z}) + \log p(\mathbf{x})\right] \tag{16}$$

By re-arranging terms, we get

$$\log p(\mathbf{x}) - D_{KL}\left[q(\mathbf{z}\,|\,\mathbf{x}) \parallel p(\mathbf{z}\,|\,\mathbf{x})\right] = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\left[\log p(\mathbf{x}\,|\,\mathbf{z})\right] - D_{KL}\left[q(\mathbf{z}\,|\,\mathbf{x}) \parallel p(\mathbf{z})\right] \tag{17}$$

This formulation describes the quantity to maximize $\log p(\mathbf{x})$ minus the error we make by using an approximate q instead of p. Therefore, we can optimize this alternative objective, called the Evidence Lower Bound (ELBO)

$$\mathcal{L}(\theta, \phi) = \underbrace{\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\left[\log p_\theta(\mathbf{x}\,|\,\mathbf{z})\right]}_{\text{reconstruction}} - \underbrace{D_{KL}\left[q_\phi(\mathbf{z}\,|\,\mathbf{x}) \parallel p_\theta(\mathbf{z})\right]}_{\text{regularisation}} \tag{18}$$

This equation involves $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ which *encodes* the data $\mathbf{x}$ into the latent representation $\mathbf{z}$ and a *decoder* $p_\theta(\mathbf{x}\,|\,\mathbf{z})$, which allows generating a data vector $\hat{\mathbf{x}}$ given a

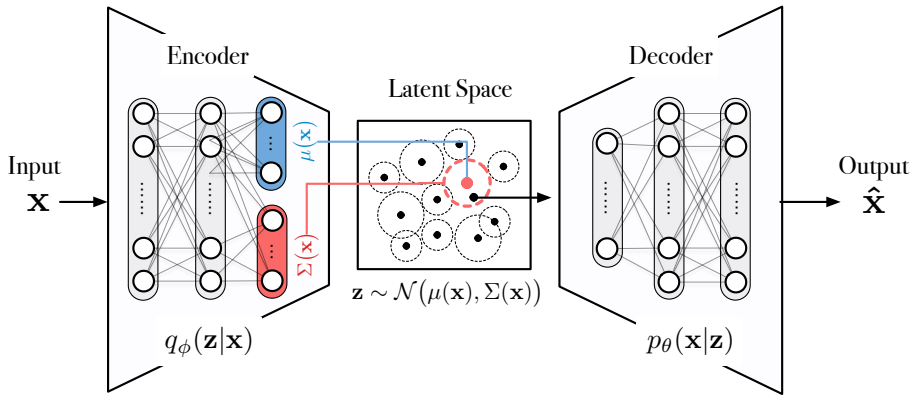


Figure 19: Representation of the architecture of the Variational Auto-Encoder.

latent configuration $\mathbf{z}$. Hence, this structure defines the VAE, whose architecture is represented on Figure 19.

The VAE objective can be interpreted intuitively. The first term increases the likelihood of the data generated given the latent $\log p(\mathbf{x} \mid \mathbf{z})$, which is equivalent to minimizing the reconstruction error. The second term represents the error made by using a simpler posterior distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ compared to the true prior $p_\theta(\mathbf{z})$. Therefore, this *regularizes* the choice of approximation q so that it remains close to the true posterior distribution. Recent work of Esling, Chemla–Romeu-Santos, and Bitton, 2018 have successfully applied VAE in the audio domain using a spectral representation of the raw waveform. They show particular advantages for the controllability of the latent space and are also capable of inferring a latent variable from unseen data. However, the generated samples tend to be slightly blurry. More recent works have addressed this issue by using a special type of Variational Autoencoder (VAE) called the Vector Quantized VAE (VQ-VAE), such as the *JukeBox* model (Dhariwal et al., 2020). This model leverages the use of variational inference for audio generation and is currently one of the most popular models in the field.

2.3.3 *Generative Adversarial Networks*

GAN were introduced by Goodfellow et al., 2014 and are *implicit* generative models that consist of two competing neural networks, a *generator* and a *discriminator*. As depicted in Figure 20, the generator G outputs fake samples given a noise variable input $\mathbf{z}$, with the objective to approximate the real data distribution. Concurrently, the discriminator D seeks to discriminate between fake and real samples by estimating the probability that a sample comes from the real dataset.

On one hand the discriminator is trained to maximize the probability that $\mathbf{x} \sim p(\mathbf{x})$ came from the dataset, so to maximize $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\big[\log D(\mathbf{x})\big]$. The discriminator is also expected to output a probability close to zero given a fake sample $\mathbf{x} \sim p_z(\mathbf{z})$, by maximizing $\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}\big[\log(1 - D(G(\mathbf{z})))\big]$. On the other hand, the generator wants to

fool the discriminator, so to minimize these two former quantities. Consequently, D and G are playing a *min-max* game, corresponding to the following objective

$$\min_{G}\max_{D}\mathcal{L}(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\big[\log D(\mathbf{x})\big] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}\big[\log(1-D(G(\mathbf{z})))\big] \quad (19)$$

Although this objective can be understood intuitively, GAN are consequently unstable to train due to two significant issues. The first issue, called the *vanishing gradient*, occurs when the discriminator is too efficient compared to the generator and produces no error, causing the loss to drop to zero, thus preventing the generator from accurately producing realistic samples. The second issue is the *mode collapse*, in which the generator is stuck and produces a low variety of samples while still fooling the discriminator.

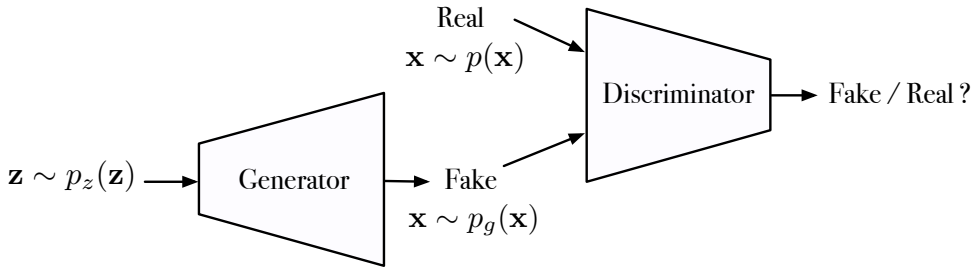


Figure 20: Overview of a GAN architecture

Aside their training instability, GAN have shown remarkable effectiveness in generating high-quality samples such the work of Donahue, McAuley, and Puckette, 2019 with *WaveGAN*, whose synthesis speed is orders of magnitude faster than that of AR models. Other models have also been developed recently, for example, *GanSynth*, which will be studied in more detail in the Section 4.2.1, as well as *MelGAN* (Kumar et al., 2019) and *HiFi-GAN* (Kong, Kim, and Bae, 2020), which will be discussed in Section 5.1.1. Nevertheless, GAN lack latent expressiveness, which make them unreliable to control. To address that issue, the combination of VAE and adversarial learning is explored in Caillon and Esling, 2021, with the *RAVE* model resulting in an expressive and controllable latent space and high-quality audio synthesis.

2.3.4 *Flow-based models*

Normalizing Flows (NF) models aim to explicitly learn the data distribution. The main idea is to start from a simple probability distribution (e.g. a Uniform or Gaussian distribution) and approximate a complex multimodal density by transforming the simpler density through a sequence of invertible nonlinear transforms. Formally, given a random variable $\mathbf{z} \in \mathbb{R}^d$ following the distribution $p(\mathbf{z})$, and f an *invertible* and *differentiable* function $f : \mathbb{R}^d \mapsto \mathbb{R}^d$. We introduce another random variable $\mathbf{x}$ mapped to $\mathbf{z}$, such that $\mathbf{x} = f(\mathbf{z})$ and through the change of variable rule, the resulting probability density is

$$p(\mathbf{x}) = p(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{x}} \right| = p(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1} \tag{20}$$

where the last equality is obtained through both the inverse function theorem and the property of Jacobians of invertible functions. Then, by repeating this operation, we can model more complex distributions, as depicted in Figure 21. Starting with a random vector $\mathbf{z}_0$ with distribution p_0, we can apply a series of mappings f_i, $i \in 1, \ldots, k$ with k and obtain a final distribution $\mathbf{z}_k \sim p_k(\mathbf{z}_k)$ leading to

$$\begin{aligned} \mathbf{z}_k &= f_k \circ f_{k-1} \circ \ldots \circ f_1(\mathbf{z}_0) = \mathbf{x} \\ p_k(\mathbf{z}_k) &= p_0(f_1^{-1} \circ \ldots \circ f_k^{-1}(\mathbf{z}_k)) \prod_{i=1}^{k} \left| \det \frac{\partial f_i^{-1}}{\partial \mathbf{z}_i} \right| = p_0(\mathbf{z}_0) \prod_{i=1}^{k} \left| \det \frac{\partial f_i}{\partial \mathbf{z}_i} \right|^{-1} \end{aligned} \tag{21}$$

Hence, the exact log-likelihood of the input data is evaluated as

$$\log p(\mathbf{x}) = \log p_0(\mathbf{z_o}) - \sum_{i=1}^{k} \log \left| \det \frac{\partial f_i}{\partial \mathbf{z}_i} \right| \tag{22}$$

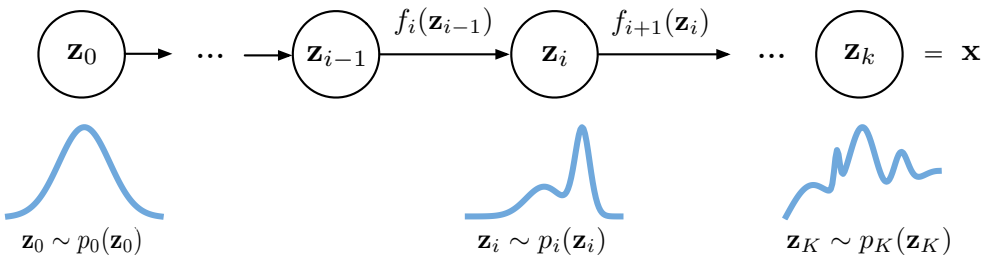


Figure 21: Illustration of chaining transforms inside normalizing flows.

In the specific context of neural audio synthesis, flows have been initially introduced as student models learned through distillation from large pre-trained WaveNets such as *Parallel WaveNet* (Ping, Peng, and Chen, 2019). Instead, more recent works directly incorporate WaveNet as a core component of the flow, removing the need for two-stage learning procedures. The authors of *FloWaveNet* (Kim et al., 2019) embed a non-causal WaveNet, which constitutes the building block of their flow architecture. Other types of architectures were recently introduced as *WaveGlow* (Prenger, Valle, and Catanzaro, 2019) or *WaveFlow* (Ping et al., 2020) trained directly by maximum likelihood on waveform data and will be both presented in Section 5.1.1.

2.3.5 *Diffusion-based models*

Diffusion-based models (DM) were originally proposed by Sohl-Dickstein et al., 2015 based on non-equilibrium thermodynamics. In their simplest form, DM define

a Markov chain of forward diffusion steps where data is iteratively corrupted by noise. Then, the aim is to learn the reverse denoising diffusion process to construct desired data samples from noise. Based on a series of latent variables $\mathbf{x}_1, \cdots, \mathbf{x}_T$ that have the same dimensionality as a given input data, which is labeled as $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the forward diffusion process $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ consists in gradually adding noise to the data, destroying the signal up to full noise. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$, so that

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \qquad q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \quad (23)$$

One important property of this forward process noted by Ho, Jain, and Abbeel, 2020 is that we can perform sampling at any arbitrary timestep t, such that

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \quad (24)$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

Oppositely, the reverse (parametric) diffusion process $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ must learn how to denoise local perturbations in order to invert this diffusion process. Hence, learning involves estimating a large number of small perturbations, which is more tractable than trying to directly estimate the full distribution with a single potential function, as illustrated in Figure 22.

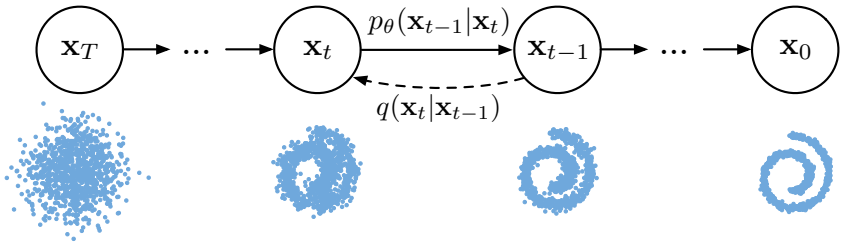


Figure 22: Illustration of the diffusion process as a generative model.

The generative distribution is trained to perform the reverse trajectory, starting from Gaussian noise to gradually remove local perturbations. Therefore the reverse process starts with a tractable distribution $p(\mathbf{x}_T) = \pi(\mathbf{x}_T)$ described as

$$p_\theta(\mathbf{x}_{0:T}) = \pi(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid \mathbf{x}_t) \quad (25)$$

Each of the transitions in this process can simply be defined as conditional Gaussians (note that this is reminiscent of the definition of VAE). Therefore, during learning, only the mean and covariance for a Gaussian diffusion kernel needs to be trained so that

$$p_\theta(x_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}, \mu_{theta}(\mathbf{x}_t, t), \sigma_\theta(x_t, t)) \quad (26)$$

DM are learned via maximum likelihood through the optimization of a variational lower bound, even though effective simplifications have been introduced to the standard training objective (Ho, Jain, and Abbeel, 2020). As a relatively recent class of models, they are yet to be extensively studied for audio generation. The seminal works are those of Kong et al., 2020 with *Diffwave* and Chen et al., 2020 with *WaveGrad*, in which a denoising diffusion model is learned through dilated convolutional architectures.

2.4 GENERATIVE AUDIO SYNTHESIS

In the specific audio context, there are other models that we could not classify in the previous section due to their specific architectures. For example, the Symbol-to-Instrument Neural Generator (SING) (Défossez et al., 2018) relies on an overlap-add convolutional architecture, which constructs sequences of overlapping audio segments and uses a sequential embedding to produce audio. The model is limited to producing individual pitched instrumental notes of fixed duration. Similarly, the WaveNet AE from Engel et al., 2017 proposes using a WaveNet-style autoencoder architecture to model the underlying structure of audio signals, allowing for more accurate and detailed synthesis and manipulation of audio. Several models, like Neural Source-Filter (NSF) (Wan et al., 2020) and Differentiable Digital Signal Processing (DDSP) (Engel et al., 2017), have extended this idea by using stronger assumptions and inductive biases from digital signal processing to model different types of signals. The NSF model splits the generation between successive source and filtering modules, mimicking traditional source-filter models. Hence, a sinusoidal (voiced) and noise (unvoiced) excitation are fed into separate filter modules, allowing to model different signals. The DDSP model implements a harmonic additive synthesizer and a filtered noise synthesizer, allowing to control the synthesis process based on frequency, loudness, and latent features from the input waveform.

We provide in Table 1 a summary of all the end-to-end waveform synthesis models that are considered in this thesis. Among those 18 models, some of them have been specifically designed to generate either speech or music, while some are able to model both. Nonetheless, these models all target audio content learning. We decided to omit TTS synthesis as it is not the primary target of this work. However, they do represent an important class of neural audio synthesis tasks.

Date	Model	Type	Speech	Music
avr.-17	WaveNet AE	AE		•
oct.-18	SING	AE+AR		•
nov.-19	NSF	-	•	
janv.-20	DDSP	AE		•
sept.-16	WaveNet	AR	•	•
févr.-17	SampleRNN	AR	•	•
févr.-22	Sashimi	AR	•	•
mai-20	Jukebox	(VQ)-VAE		•
déc.-21	RAVE	VAE+GAN	•	•
févr.-19	WaveGan	GAN	•	•
avr.-19	GanSynth	GAN		•
déc.-19	MelGAN	GAN	•	•
oct.-20	Hifi-GAN	GAN	•	
oct.-18	WaveGlow	Flow	•	
mai-19	FloWaveNet	Flow	•	
janv.-20	WaveFlow	Flow	•	
mars-21	Diffwave	Diffusion	•	
oct.-20	WaveGrad	Diffusion	•	

Table 1: All models considered in this work, including their publication date, generation strategies and reference applicability (speech, music or both).

3

BACKGROUND ON EVALUATION METRICS

In the previous chapter, we have discussed deep learning model concepts and reviewed various types of generative modeling techniques. However, a critical aspect of research studies is to find robust and adequate evaluation measures in order to compare the performance of various models. Hence, this chapter focuses on an analysis of various methods used by practitioners to evaluate deep generative models. In Section 3.1, we present *quality* metrics that are used to evaluate the artifacts produced by the model. Then, in Section 3.2, we present several measures of *efficiency* that are used to account for the computational costs of deep models. Finally, we end this chapter by discussing the concept of Pareto optimality (Section 3.3), which is the central concept in multi-objective evaluation.

3.1 MEASURES OF QUALITY

Given the complexity of their goals, generative models still lack universal metrics for the assessment of their quality or accuracy (Theis, Van Den Oord, and Bethge, 2016). Indeed, most objective metrics often do not capture the precise perceptual characteristics of the generated content. As a result, models that produce high-quality outputs may receive low scores on these metrics, while the best metrics are usually hand-crafted and often problem-specific. In this work, we refer to *automatic* metrics for evaluation measures that can be computed without the need for human intervention, and *perceptual* metrics, which rely at least to some extent on human judgments to be computed.

3.1.1 *Automatic metrics*

On the one hand, some of the quality metrics depend on the definition of the model. Sometimes, it can correspond to the training criterion of the model such as the *log-likelihood* for models trained to maximize it, or the *reconstruction* error for models that are trained to reconstruct the data. On the other hand, there are indirect quality metrics for generative models that rely on embeddings from learned representations (e.g. external classifiers) to evaluate the diversity of samples along with the generalization ability of a model. We summarize all of these different families of metrics in Figure 23 and we explain our notations along their definition in the following.

LIKELIHOOD As discussed earlier, generative models aim to model the real distribution of the data $p_r(\mathbf{x})$ by optimizing an approximate generative distribution

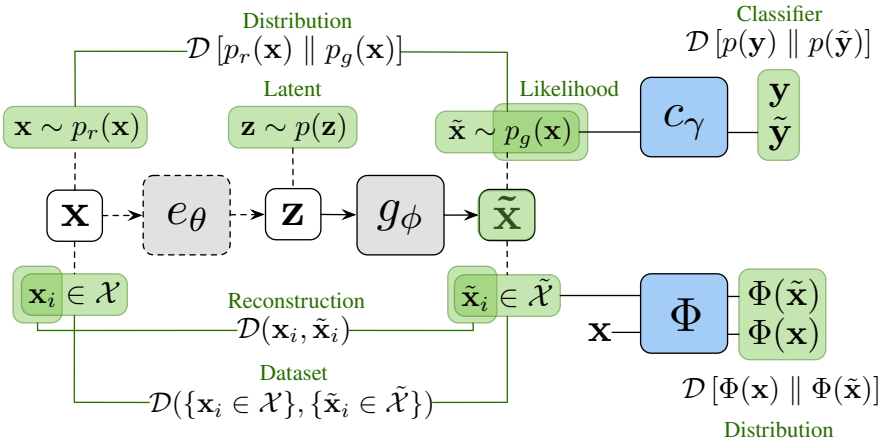


Figure 23: Summary of all generative models metrics based on our taxonomy. Green boxes depict evaluation sources, while blue ones exhibit the reliance on external models that require training.

$p_g(\mathbf{x})$. Hence, relying on the log-likelihood to evaluate a model might appear as a logical choice as it allows to evaluate their ability to explain the real (test) data $\mathbf{x}_i$ using a density estimated from the generated data such that

$$\mathcal{L}_g = \frac{1}{N} \sum_i \log p_g(\mathbf{x}_i) \tag{27}$$

However, for most of models, $p_g(\mathbf{x}_i)$ is not tractable to compute. Hence, approximation of this quantity has been proposed by relying on *Kernel Density Estimation (KDE)* (Kendall and Stuart, 1953) or *Parzen window* estimation (Parzen, 1962). However, these approximations come with two major shortcomings. First, performing kernel estimation is extremely complicated in the high-dimensional setup of the inputs $\mathbf{x}_i$. Second, it has been demonstrated in Theis, Van Den Oord, and Bethge, 2016 that they produce uninformative assessments that do not match the actual quality of the generated samples (i.e., high likelihood models can have low sample quality and conversely).

RECONSTRUCTION As we mentioned earlier, a common metric for objective evaluation is the reconstruction error, specifically for models that are trained to reconstruct input data, so that $\tilde{\mathbf{x}_i} \approx \mathbf{x}_i$. It usually involves the computation of a distance $\mathcal{D}_r(\tilde{\mathbf{x}}_i, \mathbf{x}_i)$. For instance, it is frequent to use the Euclidean distance and compute the Mean Squared Error (MSE) as follows

$$\mathcal{D}_{\text{MSE}}(\mathbf{x}_i, \tilde{\mathbf{x}}_i) = \parallel \mathbf{x}_i - \tilde{\mathbf{x}}_i \parallel_2^2 \tag{28}$$

While this metric is rather consistent with the one-by-one pixel differentiation (in the computer vision domain), in the particular context of audio this could lead to significant discrepancy due do phase distortions. Indeed the human ear

is unable to make the difference between two signals shifted in phase, however it leads to a high MSE. An alternative is to compare the log-differences of the Short-time Fourier Transform (STFT) to the reference signal and its reconstruction, generally expressed as

$$\mathcal{D}_{\text{STFT}}(\mathbf{x}_i, \tilde{\mathbf{x}}_i) = \| \, l(\mathbf{x}_i) - l(\tilde{\mathbf{x}}_i) \, \|_1 \tag{29}$$

$$\text{with } l(\mathbf{x}) = \log\left(\epsilon + |\text{STFT}[\mathbf{x}_i]|^2\right) \tag{30}$$

where ϵ acts as a trade-off between low and high energy coefficients of the spectrogram. However, these distances can only be computed based on existing samples, and not on completely unconditional generated samples which appears limiting for GAN models that are not able to perform reconstructions.

CLASSIFIER While reconstruction and likelihood can be used to evaluate the quality of a generation directly in the data domain, some techniques also propose to use alternate low-dimensional embeddings as a proxy for specific high-level attributes. It consists in an external pre-trained *classifier* $\mathcal{C}$, that produces a set of labeled data $\tilde{\mathbf{y}}$ based on the generated samples $\tilde{\mathbf{x}}$, such that $\tilde{\mathbf{y}} = \mathcal{C}(\tilde{\mathbf{x}})$.

A well-known approach using a classifier is the Inception Score (IS), proposed by Salimans et al., 2016 and was proven to be a great evaluation method for GAN models. The IS method relies on one key element: a *classifier*, which is used to assign a class label to each generated sample. The underlying assumption is that a successful generative model should produce examples examples that are classified with high confidence (as a proxy to *quality*) and from all classes evenly (as a proxy to *diversity*), such that the distribution of classes among the generated samples is uniform. Formally, the conditional probability $p(\tilde{\mathbf{y}} \mid \tilde{\mathbf{x}})$ must have a low entropy (single images classified with high confidence) and the marginal probability $p(\tilde{\mathbf{y}})$ must have a high entropy (i.e., the model generates all classes with a probability equivalent to the original label distribution) Hence, the IS is based on the KL divergence between the two distributions, and is expressed by

$$\text{IS} = \exp\left[\mathbb{E}_{\tilde{\mathbf{x}}}\left[\mathcal{D}_{\text{KL}}\left[p(\tilde{\mathbf{y}} \mid \tilde{\mathbf{x}}) \parallel p(\tilde{\mathbf{y}})\right]\right]\right]. \tag{31}$$

However, several limitations exist. First, this measure is both biased by the accuracy of the classifier used, and also by the dataset it has originally been trained on. Second, it does not capture *intra-class* diversity, as the generative models could easily generate the same image in each class. Several extensions of this idea were proposed, such as the *modified*-IS, which encourages the diversity within samples generated inside a particular category

$$\text{m-IS} = \exp\left[\mathbb{E}_{\tilde{\mathbf{x}}_i}\left[\mathbb{E}_{\tilde{\mathbf{x}}_j}\left[\mathcal{D}_{\text{KL}}\left[p(\tilde{\mathbf{y}} \mid \tilde{\mathbf{x}}_i) \parallel p(\tilde{\mathbf{y}} \mid \tilde{\mathbf{x}}_j)\right]\right]\right]\right]. \tag{32}$$

In the same line of thought, the Mode Score (MS) was proposed by Che et al., 2017 as an extension that could take into account the prior distribution of the original data labels

$$\text{MS} = \exp\left[\mathbb{E}_{\mathbf{x}}\left[\mathcal{D}_{\text{KL}}\left[p(\tilde{\mathbf{y}} \mid \tilde{\mathbf{x}}) \parallel p(\mathbf{y}_t)\right]\right]\right] - \mathcal{D}_{\text{KL}}[p(\tilde{\mathbf{y}}) \parallel p(\mathbf{y}_t)] \tag{33}$$

where $p(\mathbf{y}_t)$ denotes the marginal distribution of the classifier training data labels.

Finally, the Activation-Maximization (AM) Score (Zhou et al., 2018) aims to integrate both aspects by using the KL divergence between the distributions of training and generated labels, along with the entropy of the predictions

$$\text{AM} = \mathcal{D}_{\text{KL}}\left[p(\tilde{\mathbf{y}}) \parallel p(\mathbf{y}_t)\right] + \mathbb{E}_{\mathbf{x}}\left[\mathbb{H}(\tilde{\mathbf{y}} \mid \tilde{\mathbf{x}})\right]. \tag{34}$$

DISTRIBUTION The use of distribution metrics relies on the strong hypothesis that the distribution of samples generated $p_g(\tilde{\mathbf{x}})$ by a successful model should closely match the distribution of real samples $p_r(\mathbf{x})$. Instead of working on these direct distributions, distribution metrics use *embeddings*, which are projections of the data in a lower-dimensional space we denote $\Phi(\mathbf{x})$. Then, the idea is either to compare the samples in this space as $\mathcal{D}_r(\Phi(\mathbf{x}), \Phi(\tilde{\mathbf{x}}))$ or directly their distribution with $\mathcal{D}_d[\Phi(\mathbf{x}) \parallel \Phi(\tilde{\mathbf{x}})]$.

The Fréchet Inception Distance (FID) proposed by Heusel et al., 2017 is constructed on this idea of comparing the embeddings obtained from a pre-trained classifier. To do so, it first fit multivariate Gaussians to the two embedded representations (real and generated), leading to the approximation $\Phi(\mathbf{x}) \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $\Phi(\tilde{\mathbf{x}}) \sim \mathcal{N}(\mu_g, \Sigma_g)$. Finally, the quality is assessed trough the computation of the Fréchet distance between the two distributions, giving

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{tr}\left(\Sigma_r + \Sigma_g - 2\sqrt{(\Sigma_r \Sigma_g)}\right) \tag{35}$$

where $\text{tr}(\cdot)$ is the trace operator. The FID has shown to be strongly correlated to human evaluations, while being more robust to noise than IS. It also seems that FID is able to detect intra-class mode collapses (unlike IS).

The FID has been successfully applied to the audio domain by Kilgour et al., 2019, where they present the Fréchet Audio Distance (FAD). It relies on the exact same approach, but the classifier is a *VGGish* model trained on audio data instead of the Inception model. The FAD has been shown to be sensitive to a large set of common distortions (different types of noises, clicks, reverberation and pitch and speed changes) and appears to also be strongly correlated to human evaluations.

DATASETS *Dataset*-based metrics compare properties between sets $\{\mathbf{x}_i\}$ and $\{\tilde{\mathbf{x}}_i\}$ in the data domain directly. Hence, Richardson and Weiss, 2018 proposed the Number of Statistically-Different Bins (NDB) based on the computation of a z-test between the two datasets, i.e., a test determining whether two samples are drawn

from the same population. The underlying assumption is that if two sets of samples are drawn from the same distribution, then the number of samples that falls into a given bin should be equivalent. Formally, we can define the indicator function $\mathbb{I}_B(\mathbf{x})$ of bin B such as $\mathbb{I}_B(\mathbf{x}) = 1$ if the sample $\mathbf{x}$ falls into that bin and $\mathbb{I}_B(\mathbf{x}) = 0$ otherwise. Given the sets of real $\mathbf{x}_i$ and generated $\tilde{\mathbf{x}}_i$ samples, if they are drawn from similar distributions, it is expected that

$$\frac{1}{N_r}\sum_i \mathbb{I}_B(\mathbf{x}_i) \approx \frac{1}{N_g}\sum_j \mathbb{I}_B(\tilde{\mathbf{x}}_j) \tag{36}$$

with N_r and N_g the number of samples from the real and generated sets respectively. To perform binning, Richardson and Weiss, 2018 proposed the use of what they called *Voronoi cells*, based on K-means clustering, to ensure that each bin contains samples.

SIGNAL-BASED As our work is centered on audio signals, it is possible to extract signal processing metrics. Here, we present quality evaluation metrics used for blind audio source separation as proposed by Vincent, Gribonval, and Févotte, 2006. We reformulate the problem with our previous notations and define $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{e}_{\text{interf}} + \mathbf{e}_{\text{noise}} + \mathbf{e}_{\text{artif}}$ the reconstructed signal, with $\mathbf{e}_{\text{interf}}$, $\mathbf{e}_{\text{noise}}$ and $\mathbf{e}_{\text{artif}}$ the error of respectively interference, noise and artifacts. According to this notation, we could reformulate the Signal-to-Distortion Ratio (SDR) as

$$\text{SDR} = 10\log_{10}\frac{\|\mathbf{x}\|^2}{\|\mathbf{e}_{\text{interf}} + \mathbf{e}_{\text{noise}} + \mathbf{e}_{\text{artif}}\|^2}, \tag{37}$$

the Signal-to-Interference Ratio (SIR)

$$\text{SIR} = 10\log_{10}\frac{\|\mathbf{x}\|^2}{\|\mathbf{e}_{\text{interf}}\|^2}, \tag{38}$$

the Signal-to-Noise Ratio (SNR)

$$\text{SNR} = 10\log_{10}\frac{\|\mathbf{x} + \mathbf{e}_{\text{interf}}\|^2}{\|\mathbf{e}_{\text{noise}}\|^2}, \tag{39}$$

the Signal-to-Artifact Ratio (SAR)

$$\text{SAR} = 10\log_{10}\frac{\|\mathbf{x} + \mathbf{e}_{\text{noise}} + \mathbf{e}_{\text{interf}}\|^2}{\|\mathbf{e}_{\text{artif}}\|^2}, \tag{40}$$

3.1.2 *Perceptive metrics*

In parallel, previous works have relied on perceptual evaluations of generated samples to assess model quality. Indeed one of the core goals of generative models for audio, is to produce sound artifacts that are of sufficient quality to be indistinguishable from real ones, even when judged by human evaluators.

MEAN OPINION SCORE The Mean-Opinion Score (MOS) is one of the most popular evaluation approach used in listening experiments (Faviola Rodrigues, Riley, and Luján, 2018). It consists in asking participants to rate the sound they hear on a 5-point scale, ranging from bad quality to excellent quality (see Table 2).The final score is computed through

$$\text{MOS} = \frac{1}{N}\sum_{n=1}^{N} R_n \tag{41}$$

where R_n is one rating and N is the number of trials. Therefore, the MOS provide an absolute grade for each sound rather than a comparative one. For example, a person may give the same rating to two sounds separately, but this does not indicate preference for one or the other.

Rate	Quality
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

Table 2: MOS rating

PESQ Perceptual Evaluation of Speech Quality (PESQ) is a metric proposed by Rix et al., 2001 specifically designed for testing voice quality on low bandwidth devices, like mobile phones and smartphones. It does not require any human intervention and aims to predict the MOS through an algorithm based on psychoacoustic modeling (Herre and Dick, 2019). It produces a score from 1 to 4.5 , aligning with the MOS except the range from 4.5 to 5 which is not addressed. Although this metric has many advantages such as reproducibility and efficiency, its speech-based nature makes it unsuitable for other types of audio signals.

3.2 MEASURES OF EFFICIENCY

Although evaluating the quality of a given model is a central point in current research, it is also essential to cover the model efficiency in terms of *computation* cost and thus on *energy* and *environmental* cost. This is a major challenge with modern deep learning that tends to be resource-intensive. In this context, we distinguish two modes of evaluation: computing the efficiency for *training* a model until convergence or computing the efficiency for *inferring* new samples by the model. In the following, we present metrics related to the computational cost, and then metrics specific to the energy and environmental cost.

3.2.1 *Computation costs*

NUMBER OF PARAMETERS The most straightforward metric of efficiency is the total number of model parameters (i.e., number of weights) as it is quite easy to determine and usually directly correlated with computational complexity. A very recent and successful approach in audio generation precisely attempted to reduce this number of parameters by using the *lottery ticket hypothesis* (Esling et al., 2020a). Lighter models require less memory space (especially crucial for embedded devices) but also incur less energy consumption. Nonetheless, the number of parameters does not accurately reflect power consumption as some operations consume more than others.

FLOATING-POINTS OPERATIONS The best way to overcome this inaccuracy in power consumption is to consider the number of Floating-Points Operations (FLOP)of a model, as proposed in Schwartz et al., 2020. It is the total number of operations to compute a task (e.g., of producing one sample or of a whole training) and depends on the size of the input/output that we consider. We provide an example of an affine function (such as the sum in a perceptron) with dot product $y = \mathbf{x} \cdot \mathbf{w} + b$ where x and w are two vectors of size n. We have

$$y = w_0 * x_0 + w_1 * x_1 + w_2 * x_2 + \ldots + w_{n-1} x_{n-1} + b \tag{42}$$

The above formula has n multiplications and $n-1$ additions, plus 1 addition for the bias term, thus it uses $2n$ FLOP. We can extend this calculation to every types of neural layers, including activation layers, which count as zero parameters but not zero FLOP.

REAL-TIME FACTOR Another metric of efficiency if the Real-time Factor (RTF), a ratio that measures how much a process can be used in real-time setups. Considering that it takes a time T_{proc} to process an input of duration T_{inp}, the real time factor is defined as

$$\text{RTF} = \frac{T_{proc}}{T_{inp}} \tag{43}$$

If, for example, it takes 8 hours of computation time to produce a sample of duration 2 hours, the real-time factor is 4. When the real-time factor is 1 or less, it implies that the processing can be performed in real-time. Even though this metric is by essence hardware-dependent, it provides a first step in the direction of assessing computational efficiency quite simply.

3.2.2 *Energy cost*

Measuring the exact energy consumption of any type of computer software is an extremely challenging task, as it is usually intertwined with other processes (e.g.

cache hits and misses, memory accesses). First, we present general notions surrounding energy and power measurement in order to clarify these concepts. The energy E (in Joules) is defined as the effort to perform a task during a certain period of time T (in seconds). This can be expressed as the integral of the instantaneous power $p(t)$ during that period as

$$E = \int_0^T p(t)dt \tag{44}$$

The resulting average power (in Watts) is defined as

$$P_{avg} = \frac{E}{T} \tag{45}$$

Although we have access to power and energy metrics in some areas of electronics (e.g., with powermeters), measuring the energy consumption of any kind of computer program is already a challenging task, since there are many variables involved (e.g. cache hits, cache misses, DRAM accesses). To approximate the energy and carbon cost of training models, Strubell, Ganesh, and McCallum, 2020b decided to sample GPU, CPU and DRAM power consumption, respectively named p_g, p_c, and p_r, using the NVIDIA System Management Interface and the Intel's Running Average Power Limit. The sum of these three components is then multiplied by the Power Usage Effectiveness (PUE) coefficient, which estimates additional energies required to sustain the computing infrastructure (mainly cooling). They relied on a PUE coefficient of 1.58 as it is the 2018 global average for data centers, and end up with the following formula for the total average power

$$P_{avg} = \text{PUE} \cdot (p_c + p_r + g p_g) \tag{46}$$

with g the total number of GPUs used for training. The energy consumed is obtained as the multiplication of this total average power by the training time in seconds according to Equation (45). A popular metric for energy measurement is the kiloWatt-hour (kWh). As its name suggests, it is the multiplication of the power in kilo-Watts by the time in hours, given that 1 kWh = 3600 kJ.

CARBON COST Finally, to link kilowatt-hour and CO_2 equivalent, Strubell et al. use the carbon emission *intensity factor* (in $kgCO_2eq/kWh$). This factor is location-dependent, but can be captured in real-time thanks to the online electricity map[1].

1 https://www.electricitymap.org/map

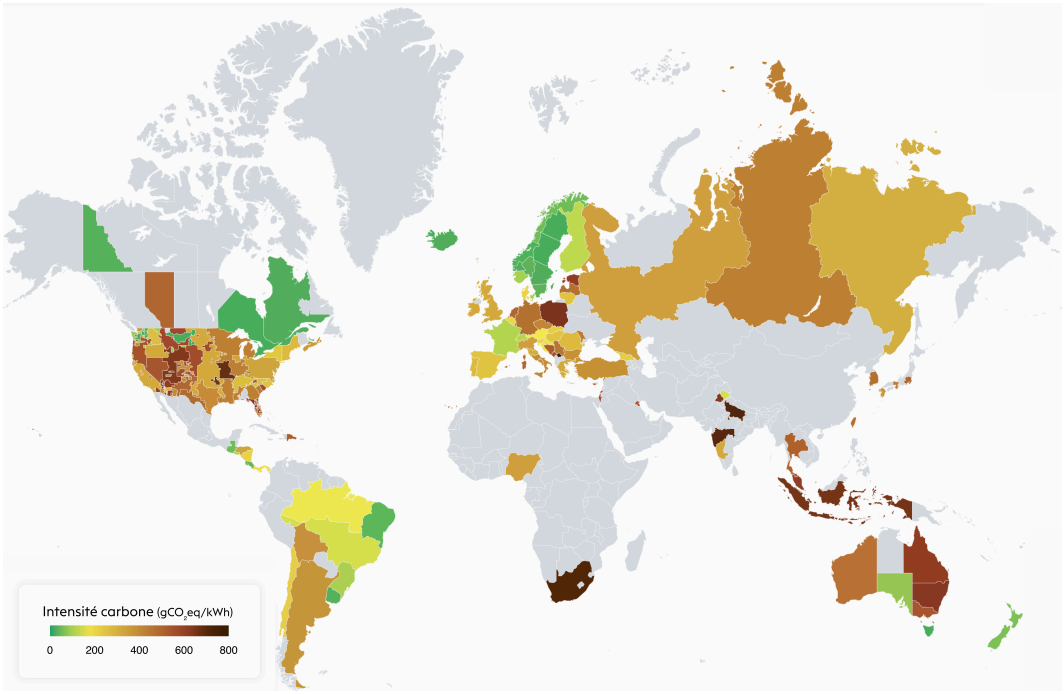


Figure 24: Snapshot of the instant emission intensity world-wide map (in gCO_2eq/kWh) on September 21, 2022

Recently, Lacoste et al., 2019 proposed an online tool called the *Machine Learning Impact Calculator*[2], which estimates carbon emissions produced while training deep leaning models according to the overall time, hardware and geographic position. In the same spirit, Anthony, Kanding, and Selvan, 2020 developed an open-source Python package called *Carbontracker*[3], which tracks energy consumption of one epoch and predicts what the entire training will consume. This provides a more accurate estimation while being user-friendly. Other methods exist to account for the on-device consumption, like the *pyJoules*[4] python package that monitors GPU, CPU and DRAM energy usage.

2 https://mlco2.github.io/impact/
3 https://github.com/lfwa/carbontracker
4 https://github.com/powerapi-ng/pyJoules

3.3 MULTI-OBJECTIVE EVALUATION

Even though we are now equipped with evaluation strategies that could account for either quality or energy, we can still only wonder how these could be linked. A quite straightforward hypothesis is that increasing the size of the network or the training time of the learning procedure often improve their quality at the expanse of larger energy costs. As these objectives seem to be conflicting, we propose to introduce the use of multi-objective evaluation criteria, also known as the Pareto principle, named after the Italian economist Vilfredo Pareto. Formally, we consider a multi-objective optimization problem as

$$\min_{x \in X} \{f_1(x), f_2(x), \ldots, f_k(x)\} \tag{47}$$

where k is the number of objective functions and x the feasible solutions.

A feasible solution $x_a \in X$ is said to *dominate* another feasible solution $x_b \in X$, notated $x_a \prec x_b$, if :

- $\forall i \in \{1, \ldots, k\}, f_i(x_a) \leqslant f_i(x_b)$
- $\exists j \in \{1, \ldots, k\}, f_j(x_a) < f_j(x_b)$

A solution $x^* \in X$ is a *Pareto optimal* solution if there are no $\hat{x}$ such that $\hat{x} \prec x^*$. The set of all these optimal solutions is called the *Pareto front*. An example is given on figure 25, red points are representing Pareto optimal solutions, and white ones non-optimal solutions.

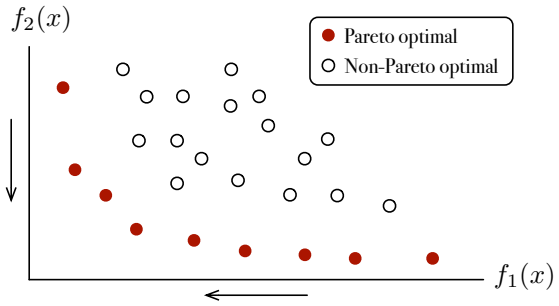


Figure 25: Example of a Pareto front where we seek to minimize two functions $f_1(x)$ and $f_2(x)$. Red points are Pareto optimal solutions while white ones are non optimal.

Now, consider two generative audio models A and B. Pareto optimality can be used to measure the dependency of two main factors: quality (f_1) and energy (f_2). For example, if A and B have the same perceptual quality, but A consumes less than B (so A *dominates* B), if there is no better solution than A, the model is Pareto optimal. Conversely, if A and B have the same energy footprint, but B produces a better sound quality, then B is optimal.

Then, if energy consumption is less important than sound quality, the goal is to find a generative model that produces the best quality while also consuming as little energy as possible. In this case, the Pareto optimal models will be those that produce the highest sound quality while consuming the least energy. These models would be located on the lower right portion of the Pareto front. On the other hand, in the case where sound quality is less important than energy consumption, the goal is to find a model that consumes the least energy while also producing acceptable sound quality. In this case, the Pareto optimal models will be those that consume the least energy while producing the least worse sound quality. These models would be located on the upper left portion of the Pareto front. Note that Pareto optimality doesn't necessarily mean that a model is the best in terms of both quality and energy, but it means that there's no other model that's better in one aspect and not worse in the other aspect.

Part II

CONTRIBUTIONS

4

TOWARDS A NEW METHODOLOGY

In the previous chapter, we discussed various strategies for the assessment of deep generative models for audio. Here, we propose an in-depth investigation of their use in the literature, as well as a first attempt to include carbon costs of training models inside the evaluation procedure. First, we present a vast review of the use of evaluation metrics in the literature (Section 4.1). Second, we address the lack of literature training costs by providing a first estimation for training state-of-the-art models based on their training details (Section 4.2). Third, we propose a new methodology based on Pareto Optimality to include the costs alongside the quality of the generated sound (Section 4.3).

4.1 DISTRIBUTION OF EVALUATION METRICS

In this section, we aim to provide an exhaustive review of evaluation metrics used in the neural audio synthesis research literature and provide an analysis of different trends regarding evaluation. Specifically, we report on nineteen publications that perform end-to-end waveform synthesis (e.g., we do not include spectrogram or text-to-speech synthesis). Then, based on the taxonomy presented in Section 3, we sort relevant evaluation metrics in two main categories : *quality* and *efficiency*. Inside the quality class, we retain six metrics : (1) the MOS or any other perceptual metrics, (2) the likelihood and its derivatives (e.g., negative log-likelihood), (3) the reconstruction scores (all distances between two samples), (4) the IS, (5) the FID and (6) the NDB as exposed in the previous section. To evaluate efficiency we retain (1) the RTF or other time-based inference metrics, (2) the number of parameters, (3) the training time , (4) the memory size and (5) the number of floating points operations. Other metrics can be used for evaluation, but they are too specific for this broad analysis. We display our analysis on Figure 26[1] and recall the meaning of the acronyms used.

The first result of this analysis shows that the MOS is the most popular quality metric used for evaluation, although it is entirely based on human judgments. Furthermore, other *objective* quality measures are largely less frequently encountered. Regarding efficiency, the RTF seems to be the reference metric for efficiency, as it is quite important in the field of neural audio synthesis to process sound in real-time. Unfortunately, there are no study mentioning the number of FLOP, authors only indicate the number of parameters of the model. Finally, although some studies

1 All details are provided in Appendix A

do mention the training time, energy consumption has simply never been taken into account, neither for training nor for inferring new sample.

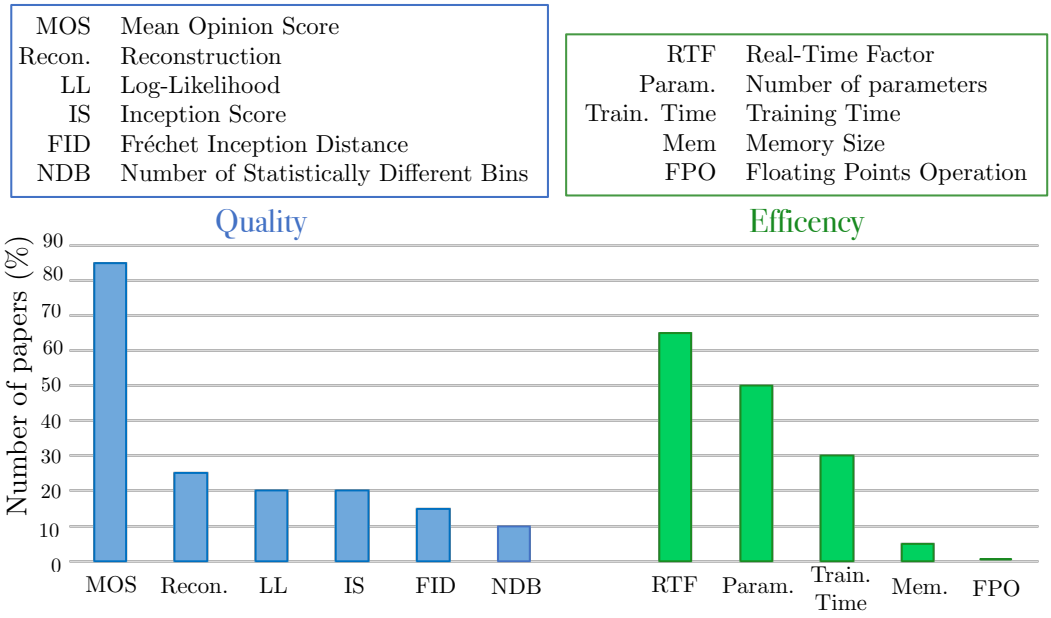


Figure 26: Distribution of commonly-used measures to compare and evaluate generative audio models. In blue (left) those that refer to the quality of the generated samples, and in green (right) those that refer to their efficiency.

4.2 ESTIMATIONS OF CARBON COSTS

4.2.1 *Models*

After this literature review, we perform the first study on the environmental impact of neural audio synthesis techniques. Therefore, we selected models for which we had enough information to estimate their energy consumption for training, including the *hardware* used to train the model, such as the type of GPU and total training time (in hours). We found out that only seven of them specified all information for both criteria. Here, we present a short description of these models and the details of their training procedure according to the original papers:

- *SampleRNN* introduced by Mehri et al., 2017 is an auto-regressive model predicting one sample at a time. It is composed of hierarchical recurrent layers working at different temporal scales. This model is trained for "about one week" on a GeForce TITAN X on three different datasets containing speech, vocal sounds and piano sonatas.

- *SING* proposed by Défossez et al., 2018 is a convolutional audio synthesizer that generates waveform given desired categorical examples. The training is composed of three parts: first, an auto-encoder is trained for 12 hours, then a

sequence generator for 10 hours and finally an end-to-end fine-tuning for 30 hours. All parts are trained on 4 NVIDIA P100 GPU on the *NSynth* dataset.

- *WaveGAN* from Donahue, McAuley, and Puckette, 2019 is a GAN that performs raw waveform synthesis using transposed convolutions acting as upsampling modules. The network is trained on a single NVIDIA P100 GPU and converges within 4 days on the *Speech Commands Zero Nine (SC09)* dataset.
- *GANSynth* by Engel et al., 2019 also uses GAN to generate log-magnitude spectrograms and instantaneous frequencies instead of modeling the raw waveform directly. The training lasts for 4.5 days on a NVIDIA V100 GPU on a subset of the *NSynth* dataset.
- *FloWaveNet* proposed by Kim et al., 2019 is a flow-based model for parallel waveform speech synthesis using the WaveNet architecture as an inverse transformation function. The training requires 11.3 days on a NVIDIA Tesla V100 GPU and operates on the *LJSpeech* dataset.
- *JukeBox* from Dhariwal et al., 2020 synthesizes sound thanks to a multi-scale Vector Quantized-Variational AutoEncoder (VQ-VAE) trained for 3 days on 246 NVIDIA V100 on a large dataset composed of 1.2 million songs where they also perform data augmentation. They also train a prior model and a lyric conditioner, which we did not consider as we solely aim to evaluate the cost of neural synthesizer modules.
- *RAVE* by Caillon and Esling, 2021 is a promising neural synthesizer and is based on a two-stage learning procedure, where the first part consists in training a VAE, then fine-tuning with a GAN. The total has been trained 6 days on a single TITAN V GPU, on a dataset composed of 30 hours of raw recordings of strings in various configurations.

Although we selected these seven models for the availability of their training details and not for their specific architecture, we believe that they form a representative set of the current state of research in generative models.

4.2.2 *Training costs*

Here, we want to estimate the carbon emissions of various training procedures for each of the selected models. Since we did not have access to all of the previously-mentioned specific hardware, some hypotheses had to be taken into account. First, we make the assumption of the worst-case scenario, as does the Machine Learning Impact Calculator: we take the maximum power consumption p_{max} for each of the GPUs according to their technical specifications, and multiply it by g, the number of GPUs used for training and by t in hours, to get the kilo-Watt hours consumption. We assume that the models are optimal and take most of the GPU resources. A further justification for our hypothesis is that we omit DRAM, CPU and additional energies for cooling while training.

Second, as carbon emissions are location-dependent, we selected a carbon intensity factor of 0.437kgCO_2eq/kWh as it is the global yearly average of 2018[2] to convert kilowatt-hours to carbon emissions. We finally obtain the following formula to estimate the carbon emission of an entire training as

$$CO_2e = 0,437 \times g \times p_{max} \times t \tag{48}$$

4.2.2.1 *Results*

Estimations of training costs are shown in Table 3. We summarize the training details, and display the corresponding kilo-Watt hours and carbon footprint estimations for each of the seven studied models.

Model	Hardware	Power	Hours	kWh	kgCO2eq
FloWaveNet	1 Tesla V100	300 W	272	81.6	35.66
Jukebox	256 Tesla V100	300 W × 256	72	5529.6	2416.44
GANSynth	1 Tesla V100	300 W	108	32.4	14.16
WaveGAN	1 Tesla P100	250 W	96	24	10.49
RAVE	1 TITAN V	250 W	168	42	18.35
SampleRNN	1 GTX TITAN X	250 W	168	42	18.35
SING	4 Tesla P100	250 W × 4	52	52	22.72

Table 3: Approximated energy consumption for training several state-of-art generative audio models. Power is expressed in Watts and energy in kWh.

Consumption	lbsCO2eq	kgCO2eq
Air travel, 1 passenger, NY ↔ SF	1984	899.9
Human live, avg, 1 year	11023	4999.9
American live, avg, 1 year	36156	16400.1
Car, avg incl. Fluel, 1 lifetime	126000	57152.6

Table 4: Estimated CO2 emissions from familiar consumption, from Strubell, Ganesh, and McCallum, 2020b

As we can see, the energy consumption of training neural synthesis models ranges from 24 kWh to 5529.6 kWh, and the corresponding carbon footprint estimation from 10.49 kgCO_2eq to 2416.5 kgCO_2eq. To provide clearer comparisons, we display a set of well-known carbon costs in Table 4 adapted from Strubell, Ganesh, and McCallum, 2020b in our metric system (i.e. 1 lbs = 0.45359237 kg).

2 https://www.carbonfootprint.com

First, we can clearly see that the carbon cost of all of the training procedures is quite high and should not be overlooked. Second, the particular example of Jukebox questions the effectiveness and worthiness of such extensive training, as it is almost on par with the emissions of a human being for a whole year. Furthermore, these estimations do not include the total amount of energy used upstream to train and test all the different versions to find adequate architectures and configurations, or any other ablation studies. It only considers the baseline model as published, generally used as is in the publication.

To confirm that our predictions are close to the real energy demand, we measure the consumption of training SING (auto-encoder, sequence generator, and fine-tuning) on a single TITAN X with the same configuration as the original paper. We use Carbontracker Anthony, Kanding, and Selvan, 2020 to predict the energy consumption of the whole learning process by only computing one epoch and multiplying it by the number of epochs from the paper. We found out the training consumes 64.8 kWh, which is slightly higher than our initial estimation of 52 kWh, but stays within the same range.

At this point, we believe that comparing models purely on the basis of these estimations is questionable, and argue that the real energy should have been recorded. Moreover, the estimations presented in Table 3 are linearly dependent on the training time, which is itself linearly dependent on the number of epochs before convergence and thus on the accuracy of the model. In other words, although heavy models trained on large datasets may consume more, they are often more accurate, but is this gain worth the additional energy consumption? As a result of this reasoning, we advocate using a multi-criteria evaluation based on Pareto optimality (Section 3.3) to find the best compromise between quality and energy consumption, which is the purpose of the next section.

4.3 PROPOSED METHODOLOGY

As we just discussed, although heavy models trained on large datasets emit more carbon, they are often more accurate. Therefore, we propose a new methodology that considers the best trade-off between quality and energy consumption. In the following, we rely on our previous estimations and start with an inter-model study (e.g. different neural synthesis models), and then conduct an intra-model study (e.g. same models but alternative configurations) on the *Waveflow* model.

4.3.1 *Inter-model study*

As discussed earlier, measuring the quality of generative models remains a daunting task. The plurality of metrics used in the literature (Section 3.1) comes with the diversity of architectures proposed (Section 2.3). The most popular and relevant measure across the audio generation literature is the MOS, as exposed in

Section 4.2. Hence, the first part of this study consists in gathering all the MOS from the literature we dispose of. As SampleRNN and GANSynth use pairwise comparison instead of MOS, we withdraw them from the study. We also exclude Jukebox, that only uses objective reconstruction scores for its evaluation. We summarize in Table 5 the MOS of the models MOS_M and those of the ground truth MOS_GT reported in each original paper. As models are compared across different experimental setups, we propose to compute the $\%\text{MOS} = \text{MOS}_\text{M}/\text{MOS}_\text{GT}$ for our quality metric, to have a relatively reasonable point of comparison. Note that the better the quality, the closer %MOS is to 1. We also add the number of parameters used by each model to infer new samples according to their original architectures for the inference efficiency metric.

Model	$\textbf{MOS}_\text{GT}$	$\textbf{MOS}_\text{M}$	**%MOS**	**Param.**
FloWaveNet	4.67	3.95	0.85	186 M
RAVE	4.21	3.01	0.71	17.8 M
SING	3.86	3.55	0.92	64 M
WaveGAN	2.3	3.9	0.59	89 M

Table 5: Comparative Mean Opinion Scores ratios (%MOS) and number of parameters of several state-of-the-art neural audio synthesis models.

We display in Figure 27 the multi-objective space, where we plot the Pareto front for training (up) and for inference (bottom). We reverse the quality axis to have the optimal point at the bottom left corner for both fronts. Therefore we can highlight models that are Pareto efficient (with a red circle). On the one hand, we see that RAVE, SING and WaveGAN are Pareto optimal in training, whereas FloWaveNet is not (as it is dominated by SING and RAVE). Note that we would have the same conclusions if we had considered the energy cost instead of the carbon cost, as it is linearly related by the carbon intensity factor. On the other hand, WaveGAN and FloWaveNet are both dominated by RAVE and SING in inference, which indicates that those optimal models are to be preferred for the deployment of corresponding audio applications on a large scale, depending on whether we are looking for more quality (SING) or more efficiency (RAVE). Since our goal is to propose a new tool for sustainable evaluation of models, we did not retrain the models to make our work more consistent and greener. Therefore, we would like to clarify that we rely on approximations and hand-crafted measures of quality; these figures support our overall approach, and warrants more extensive and reliable analyses.

4.3.2 *Intra-model study*

Since the training energy consumption is highly dependent of the dataset on which the model was trained on, and the MOS of the experimental setup, we propose to conduct an intra-model study. Therefore, we rely on a single model that provides

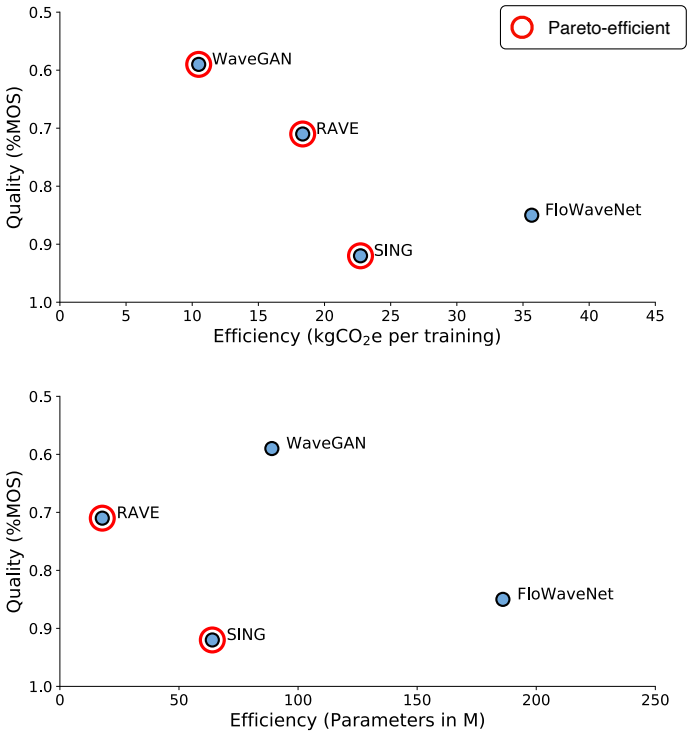


Figure 27: Example of two Pareto fronts (in red). The objective is to minimize the quality score (%MOS) along with the energy efficiency of either the training (top) with the measure of the carbon emission ($kgCO_2e$) per training, or the inference (bottom) with the number of parameters.

a large number of MOS for a wide range of configurations : WaveFlow (Ping et al., 2020). We rely on a PyTorch implementation[3], and use the same configurations as the original paper for all of our experiments.

To measure the energy consumption of each of the learning procedure, we train models on 4 TITAN V with a batch size of 8. As previously, we use Carbontracker to predict the training energy and carbon cost for each configuration. We compute the number of iterations per epoch and derive the number of epoch needed since the original paper does not specify the number of epochs but rather the number of steps: lighter models (res. channels 64 and 96) are trained for 3M steps, medium ones (res. channels 128 and 96) for 2M and the heaviest one (res. channels 256) for 1M. We summarize in Table 6 the MOS from the original paper Ping et al., 2020 along with the number of parameters for each of the five configurations studied. We also include our measurement of the energy E_{train} in kWh required to train as it does not depend on the location of the experiments, and the energy to generate 10 audio clips of 10 seconds at 22.05 kHz on a single TITAN V GPU E_{gen} in Wh.

Model	**Param.**	**MOS**	$\mathbf{E}_{train}$ (kWh)	$\mathbf{E}_{gen}$ (Wh)
WaveFlow 1 ($h = 8$, $r = 64$)	5.91 M	4.26	407.7	1.349
WaveFlow 2 ($h = 16$, $r = 64$)	5.91 M	4.32	437.6	1.382
WaveFlow 3 ($h = 16$, $r = 96$)	12.78 M	4.34	725.4	2.382
WaveFlow 4 ($h = 16$, $r = 128$)	22.25 M	4.38	644.8	2.512
WaveFlow 5 ($h = 16$, $r = 256$)	86.18 M	4.43	1011.2	3.871

Table 6: Subjective score (MOS) for multiple configuration from Ping et al., 2020 and their number of parameters. E_{train} and E_{gen} stands respectively for the amount of energy required for a whole training, and the amount of energy to produce 100 seconds of raw audio at 22.05 kHz. h is the squeezed height and r the residual channels, for more information see Ping et al., 2020.

We display in Figure 28 three multi-objective space. The one on the top accounts for training efficiency, while the two others account for generation efficiency, one with the number of parameters, and the other with the real on-device energy consumption. As before, we highlight Pareto optimal models with a red circle. The multi-objective analysis shows that the third configuration of Waveflow, WF_3, is dominated by others in training. However, it seems like in the inference fronts, this configuration is Pareto-optimal. If we take a closer look at the difference between the parameters front and the energy front, we can see that the gaps between the models are increasing. This justifies that taking into account the energy rather than counting the number of parameters is highly valuable. Still, in the last energy front, all models are optimal, which shows that internally to a model, the linear relationship between accuracy and energy cost might hold.

3 https://github.com/L0SG/WaveFlow

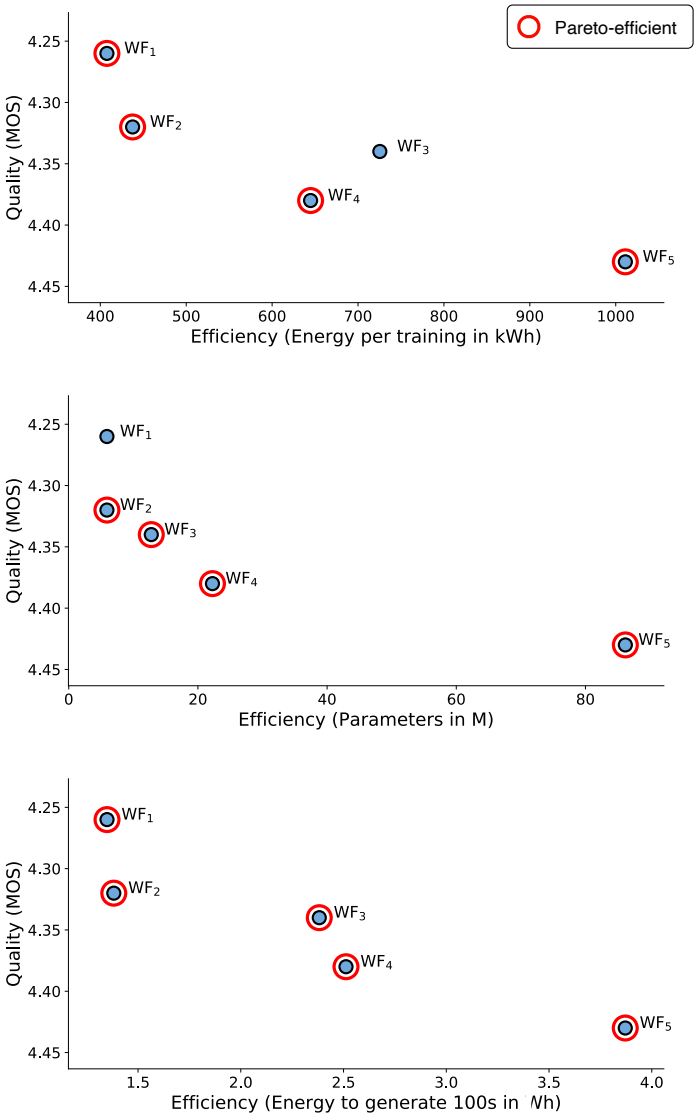


Figure 28: Representation of three Pareto space for optimizing quality (MOS) and energy of either the training cost (top) or the inference cost (middle and bottom) of generative audio models. WF_i stands for WaveFlow i. In red, all optimal solutions, while in red dominated ones.

4.4 CONCLUSION

In this chapter, we presented the first comprehensive cartography of evaluation metrics in the deep audio literature. By collecting specificities from existing generative audio model papers, we approximate the energy consumption of these models by considering factors such as hardware and training time. We highlight the need to link this calculation to the quality of the models and propose using a new evaluation metric based on Pareto optimality, which gives equal importance to both model quality and energy consumption. We illustrate this concept through an inter-model analysis based on well-known neural synthesizer, and an intra-model analysis focusing on the WaveFlow model. Our results show that this 2D representation facilitates the overall evaluation of research across multiple objectives, allowing for the rapid identification of optimal Pareto models. Furthermore, it provides more comprehensive and visual evaluations. As far as we know, this is the first study on energy consumption for neural audio generation and a pioneering attempt to include energy efficiency in the entire evaluation procedure. However, we acknowledge that this study is only a preliminary analysis and that further research is warranted. Indeed, several factors can impact the accuracy of our results, such as the approximations of training costs and handcrafted metrics used for quality assessment. Additionally, a denser evaluation with more reference methods would be beneficial for this evaluation, which will be the focus of the next chapter.

5 LARGE-SCALE BENCHMARK EVALUATION

One of the major limitations in understanding the current impact of deep audio models is the lack of comprehensive benchmark analysis of generative models for audio. In this chapter, we aim to address this limitation by conducting a full benchmark analysis of neural vocoders, whose goal is to synthesize waveforms from spectral representations such as mel-spectrograms (see Figure 29). We aim to show that our proposed multi-objective Pareto optimality criterion can provide a more comprehensive comparison of generation quality and energy efficiency across a large number of models and configurations. In Section 5.1, we describe the experiments we conducted, and in Section 5.2, we present the results of our large-scale evaluation.

5.1 NEURAL VOCODERS BENCHMARK

5.1.1 *Models*

In order to account for the energy cost inside the evaluation of neural vocoders, we consider six state-of-the-art models belonging to three major families of generative models discussed in Section 2.3: GAN, NF and diffusion-based models. Within these groups, we respectively consider *MelGAN* (Kumar et al., 2019) and *HiFi-GAN* (Kong, Kim, and Bae, 2020), *WaveGlow* (Prenger, Valle, and Catanzaro, 2019) and *WaveFlow* (Ping et al., 2020) and, finally, *WaveGrad* (Chen et al., 2020) and *DiffWave* (Kong et al., 2020). The choice of these models was dictated by their impact in the community and their rather recent introduction. At the time of these experiments, all of these models have been introduced within the previous three years. Our choice to exclude autoregressive models is due to their prohibitively long

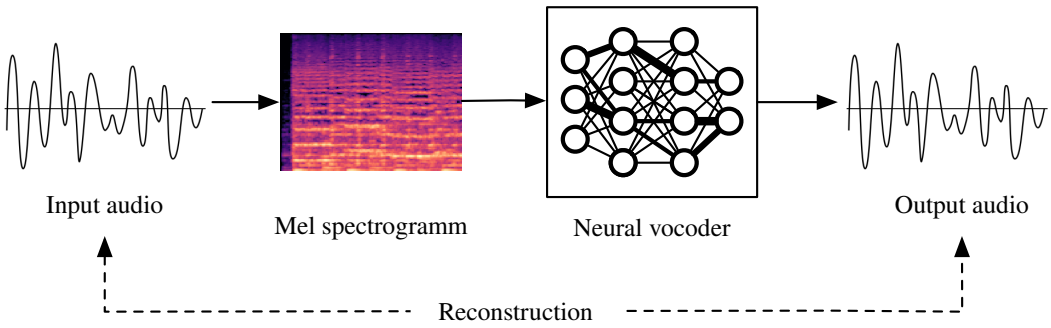


Figure 29: Scheme of a simplified neural vocoder

inference time (hence rarely used in applications). For each of these six models, we consider three different configurations: *small*, *medium* and *large* configuration. Most variants of these architectures were already proposed in the original papers, while others are our own suggestion following the same logic as in the considered papers (e.g. channel or depth variations). Here, we present a short description of those models where we detail the architectures and configurations used.

- *MelGAN* (Kumar et al., 2019) uses a multi-scale architecture for the discriminator, with three identical convolutional neural networks operating at three different time scales (raw audio, 2x and 4x downsampling respectively). The generator alternates upsampling layers and residual blocks with dilated convolutions in order to increase the induced receptive fields and obtain better long-range audio modeling. In our experiments, we took the default configuration proposed in the paper as our *medium* configuration (4.27M parameters), while we propose the *large* and *small* configurations based on variations of the number of channels and residual layers in the generator. The final small model has 1.03M parameters, and the large has 18.21M.

- *HiFi-GAN* was proposed by Kong, Kim, and Bae, 2020 as a follow-up to the MelGAN model. They show that the introduction of a multi-period discriminator in addition to the multi-scale discriminator helps modeling highly periodic signals. This new discriminator combined with an improved generator architecture yields state-of-the-art results in spectrogram inversion, both computationally and perceptually. In our study, we chose to keep the three configurations proposed by the original paper, namely V1 (13.92M), V2 (0.92M) and V3 (1.46M) which we use for our large, small and medium configurations respectively. Although the small and medium versions have a close number of parameters, they still lead to very different quality. Therefore we still consider both of these models.

- *WaveGrad* follows the original diffusion training scheme and was introduced by Chen et al., 2020, inspired by a U-Net inspired architecture for spectrogram to waveform inversion. They show that conditioning the model on a continuous noise level instead of the discrete iteration index allows the use of various noise schedules during inference. This reduces the number of denoising iterations down to 6 while still synthesizing high-quality audio signals. They compare two network size variations: *base* (15M) and *large* (23M). We attempt to consider both, however the *large* architecture appeared too unstable in training. Thus, we propose two other configurations based on the *base* model (our medium) with a variation of channel dimensions but keeping the same number of layers (small and large).

- *DiffWave* proposed by Kong et al., 2020 employs an architecture inspired by previous works on source separation. Mel-spectrograms are first upsampled to the length of the desired waveform through the action of transposed 2D-convolutions. Then, they employ a WaveNet-like network composed of

a stack of layers with residual channels, based on non-causal bidirectional dilated convolutions. We use the two configurations provided in the original paper (Base 2.64M and Large 6.91M), and add our own *medium* configuration based on varying the size of residual channels.

- *WaveGlow* (Prenger, Valle, and Catanzaro, 2019) was among the first flow-based model trained directly by maximum likelihood on waveform data. They build their model on top of Glow Kingma and Dhariwal, 2018 and WaveNet, alternating affine coupling layers and 1×1 invertible convolutions: the transformation inside the coupling layer, which they refer to as *squeezing* employs a WaveNet-like architecture that takes as input the first half of the input channels and the mel-spectrogram. As they only propose one implementation in the original publication, we took the configurations used in *WaveFlow* that we present in the following.

- *WaveFlow* (Ping et al., 2020) extends the previous works on single-architecture and single-loss flow models. They argue that the application of the (channel-wise) *squeeze* operation introduced in WaveGlow may lead to the loss of temporal order information. Instead, they propose to squeeze the input waveform $\mathbf{x} \in \mathbb{R}^n$ into an h-row matrix $\mathbf{X} \in \mathbb{R}^{h \times w}$ and employ 2D dilated convolutions (with causal constraints enforced on the height dimension). They tested lots of configurations but only four were used for real evaluation. Among them, we took the three smaller configurations (small 5.91 M, medium 12.78 M and large 22.25 M) because of training time constraints.

5.1.2 *Dataset*

We train and evaluate all models on one of the reference datasets in speech generation, namely the *LJSpeech* dataset (Ito, 2017). It is a publicly available dataset of 13,100 short audio clips of a single speaker reading passages of different texts, totaling more than 24 hours of audio data. The passages were selected to include a variety of different styles and genres, such as poetry, children's books, and scientific articles. The recordings are in the WAV format and have a sample rate of 22,050 Hz. In our experiments, we apply the preprocessing strategy which is the most commonly used across the tested models (WaveGlow, MelGan and WaveFlow). Hence, after downsampling the data to 16,000 Hz, we keep only the first $N = 2^{14}$ samples from each clip and then extract an 80-bands mel-spectrogram $\mathbf{s}$ from this audio with a FFT of size 2048 and a hop size of 256. We then perform *min-max* normalization on each spectrogram $\mathbf{s}$ as follows

$$\mathbf{s}' = \frac{\mathbf{s} - \mathbf{m}_{\mathcal{D}}}{\mathbf{M}_{\mathcal{D}} - \mathbf{m}_{\mathcal{D}}} \tag{49}$$

where the minimum $\mathbf{m}_{\mathcal{D}} = \min_{\mathbf{s}_i \in \mathcal{D}} \mathbf{s}_i$ and maximum $\mathbf{M}_{\mathcal{D}} = \max_{\mathbf{s}_i \in \mathcal{D}} \mathbf{s}_i$ are computed across the entire dataset $\mathcal{D}$. Finally, we split the data between *training* (80%) and *testing* (20%) sets.

5.1.3 *Training*

All models are trained with a maximum time budget of 120 hours on a single NVIDIA RTX A5000 GPU. We believe that this training time is a consistent upper bound to ensure that all models converge, although certain models such as diffusion, converge faster than flow models. Furthermore, the choice of using a single GPU allows both to simplify the energy consumption cost, and also represent the minimal computational capacities of research institutions. The batch size is automatically scaled to maximize the GPU memory usage in order to enhance parallelization and minimize the convergence time. For all models, we use the ADAM optimizer Kingma and Ba, 2015 and rely on the respective learning rate of each of the tested models in their original implementations. All code is available here : https://github.com/ConstanceDws/neural-audio-energy.

5.2 LARGE-SCALE EVALUATION

5.2.1 *Monitoring convergence*

First, we aim to ensure that all of the evaluated models have converged correctly. To do so, we rely on metrics from the *Speechmetrics*[1] toolbox in order to monitor in-training convergence. We display the results in Figure 30 for the MOSNet metric (Lo et al., 2019, which is a neural network predicting the MOS scores, which aims to emulate MOS scores on speech. We also provide the full range of metrics used to monitor convergence in Appendix B.

1 https://github.com/aliutkus/speechmetrics

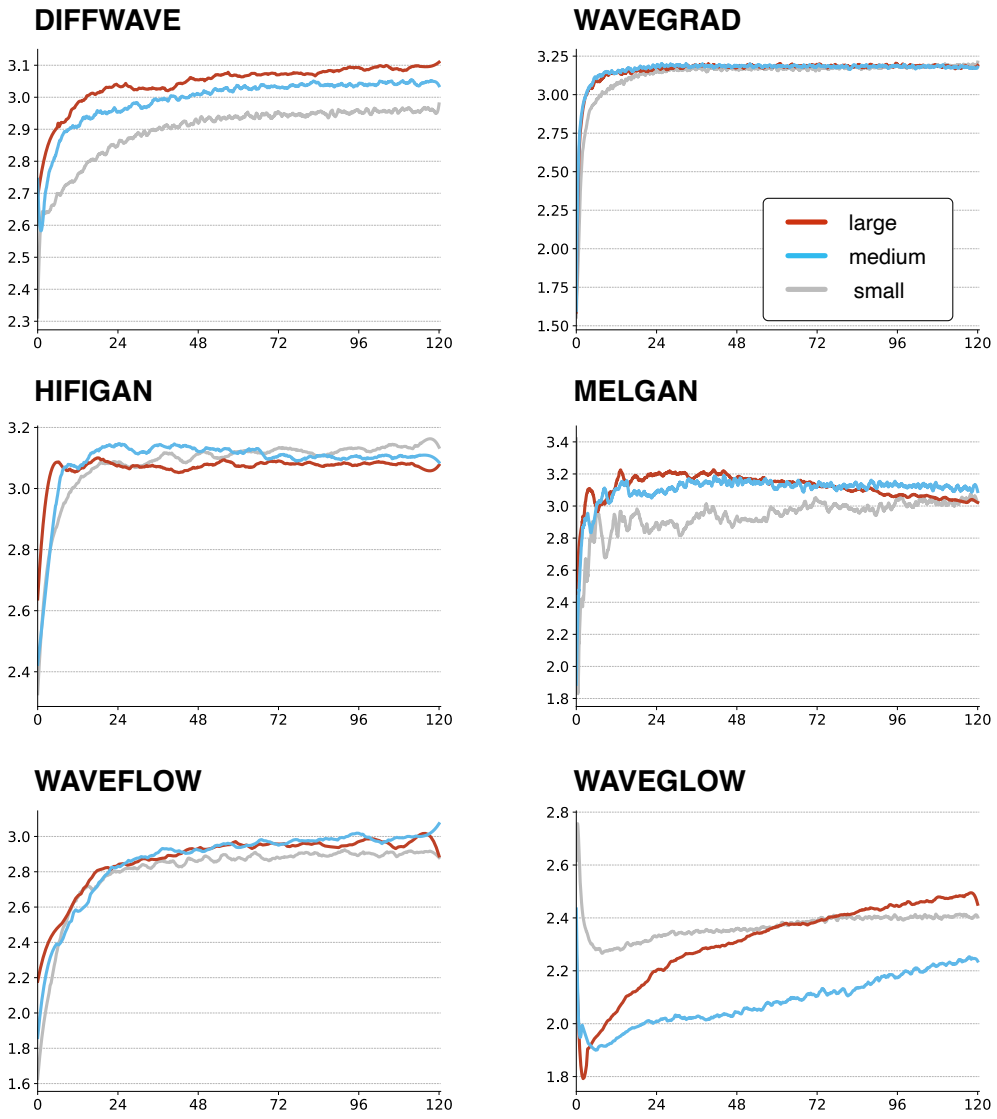


Figure 30: Mean Opinion Score estimation using MOSNet throughout the training procedure in the validation step computed at the end of each epochs. The x-axis is the training time in hours.

As we can see on Figure 30, the speed of convergence clearly differs from one model to another. Some models like *WaveGrad* tend to reach a plateau quite quickly after only a few hours of training, while other models such as *WaveFlow* have a more steady curve of improvement across the entire training time budget. While being an excellent indicator for monitoring, the MOSNet estimation is not fully reliable for perceptual quality evaluation, since it was trained on voice conversion types of sound instead of generative models samples, which may produce different types of artifacts.

5.2.2 *Synthesis quality*

In order to provide a more accurate estimation of perceptual audio quality, we performed a human-based perceptual quality evaluation based on the *pymushra*[2] web application. Participants were asked to rate the naturalness of sounds, by grading each samples between 1 ("*bad*") and 5 ("*perfect*"). In this analysis, we include both the ground truth data (from the test set) and each model reconstruction. A total of 41 participants undertook the complete test, the majority of whom were audio professionals. We present the results of this MOS evaluation in Table 7, alongside the STFT reconstruction quality, denoted as $\mathcal{D}_{\text{STFT}}$. We also compute the Inception Score (IS), the Activation Maximization Score (AM) and the Fréchet Inception Distance (FID) using a pre-trained Vggish[3] classifier.

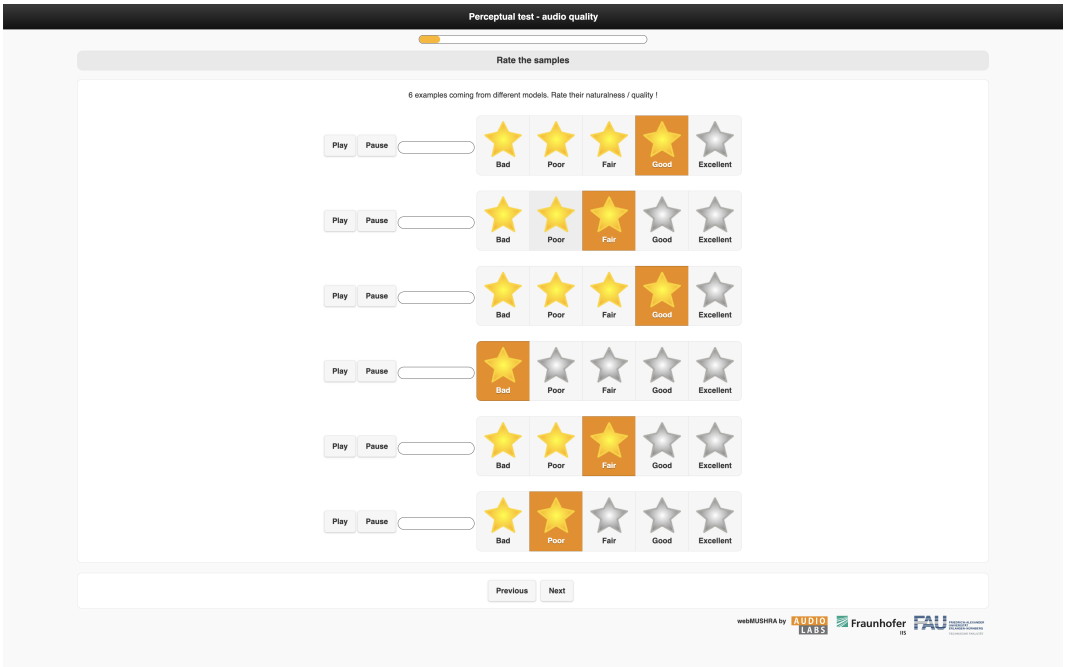


Figure 31: Screenshot of the Mean Opinion Score (MOS) survey.

As we can see, there are large discrepancies between the MOS and $\mathcal{D}_{\text{STFT}}$ when evaluating the generation quality. This underlines the need for human-based evaluation, as slight reconstruction artifacts can have a large perceptual impact. Indeed, although the DiffWave model have the lowest $\mathcal{D}_{\text{STFT}}$ reconstruction error, it exhibits quite low MOS scores. Overall, the WaveGrad and HiFi-GAN models (all configuration included) have largely higher MOS scores than other models. On the other hand, WaveGlow has rather poor MOS results, which could be explained by the lack of sufficient training time when compared to the original paper. Across

2 https://github.com/nils-werner/pymushra
3 https://github.com/tensorflow/models/tree/master/research/audioset/vggish

Model	**MOS** (↑)	$\mathcal{D}_{\text{STFT}}$ (↓)	**IS** (↑)	**AM** (↓)	**FID** (↓)
Ground truth	4.34 (±0.005)	0	1.0048	4.7590	$\sim 10^{-5}$
MelGAN* SMALL	1.37 (±0.004)	0.1496	1.0043	4.8119	3.9202
MelGAN MEDIUM	2.12 (±0.007)	0.1199	1.0037	4.7917	1.4035
MelGAN* LARGE	2.22 (±0.007)	0.1146	1.0038	4.7872	1.1023
HiFi-GAN SMALL	3.90 (±0.007)	0.0804	1.0042	4.7831	1.0486
HiFi-GAN MEDIUM	3.59 (±0.008)	0.0791	1.0044	4.769	0.4773
HiFi-GAN LARGE	**4.12** (±0.007)	0.0712	1.0045	**4.7654**	**0.2232**
WaveGrad* SMALL	3.24 (±0.007)	0.0758	1.0047	4.7881	2.0988
WaveGrad MEDIUM	3.66 (±0.008)	0.0736	**1.005**	4.7748	1.3816
WaveGrad LARGE	3.59 (±0.007)	0.0709	1.0047	4.778	1.5758
DiffWave SMALL	1.89 (±0.006)	0.0838	1.004	4.7934	4.86
DiffWave MEDIUM	2.18 (±0.006)	0.0725	1.0042	4.7918	5.3131
DiffWave LARGE	2.41 (±0.007)	**0.0698**	1.0042	4.7949	5.6829
WaveFlow SMALL	1.50 (±0.005)	0.1192	1.0038	4.8041	2.7953
WaveFlow MEDIUM	2.44 (±0.010)	0.1059	1.0037	4.7885	1.2247
WaveFlow LARGE	2.77 (±0.008)	0.1180	1.0038	4.789	1.2211
WaveGlow SMALL	1.07 (±0.002)	0.1518	1.0033	4.8411	8.3917
WaveGlow MEDIUM	1.52 (±0.004)	0.1177	1.0035	4.8143	3.7899
WaveGlow LARGE	1.80 (±0.006)	0.1136	1.0037	4.8092	3.1702

Table 7: Perceptual (Mean Opinion Score) and reconstruction ($\mathcal{D}_{\text{STFT}}$) qualities of neural vocoders conditioned on mel-spectrogram. (*) indicate configurations that we suggest in addition to those of the original papers.

all models, it appears that increasing the size of the architecture consistently increases the quality of the corresponding generation, which is coherent with the current trend in the scientific literature and was expected at this point of the study. We also plot the correlations between our perceptual results and the three metrics from the pre-trained classifier on Figure 32. Surprisingly, we find that AM and MOS were notably highly correlated, while the other two metrics show less correlation but still confirm their use in the evaluation of audio generative models.

5.2.3 *Energy efficiency*

In order to better understand the tradeoff between increased size (and quality) of the models and their corresponding energetic impact, we compute the num-

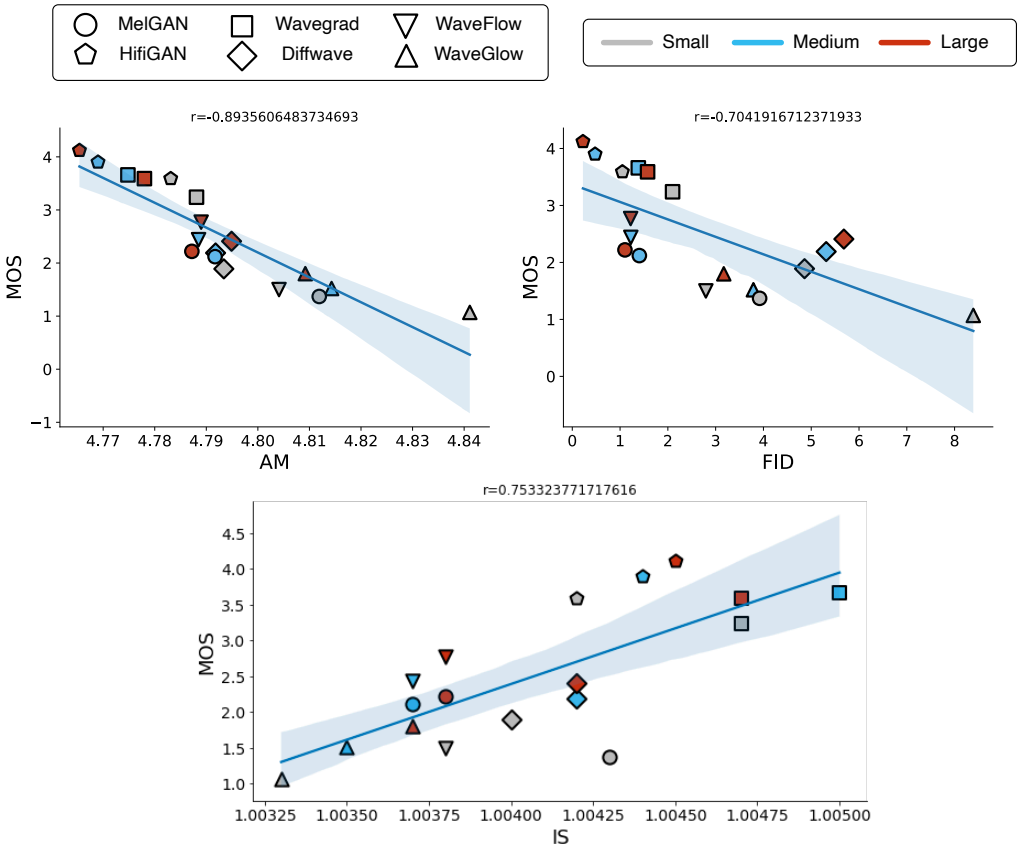


Figure 32: Correlations between quality metrics computed from a pre-trained classifier and perceptive MOS score.

ber of parameters as well as the number of floating point operations per second of generated content. We then record the energy for our models to generate 10 audio clips of 10 seconds at 16 kHz on a single NVIDIA RTX A5000 GPU using the *pyJoules* package[4] that estimates energy consumption of the CPU, RAM and integrated GPU. We also include the Real-Time Factor (RTF) from producing those samples for both GPU and CPU[5]. For flows models, we remove the weight normalization layers as they slow down the audio generation without impacting the corresponding inference quality. We summarize all of these energy footprint metrics in Table 8.

By analyzing energy footprint and GFLOPs, we observe significant differences among various generative models. GANs tend to be highly efficient, while diffusion models have an inference energy cost around 100 times higher. The RTFs also confirm these conclusions; GANs are already real-time on CPU, while diffusion-based models are not yet real-time. It's important to note that these figures depend

4 https://pyjoules.readthedocs.io/en/latest/
5 Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz

Model	# Param	# GFLOPs	E_{gen} (mJ)	RTF GPU / CPU
MelGAN* SMALL	1.03M	1.39	49.08	× 0.008 / 0.01
MelGAN MEDIUM	4.27M	7.02	69.46	x 0.009 / 0.02
MelGAN* LARGE	18.21M	33.98	95.14	x 0.01 / 0.11
HiFi-GAN SMALL	0.928M	2.56	59.28	x 0.008 / 0.02
HiFi-GAN MEDIUM	1.46M	3.22	64.59	x 0.009 / 0.02
HiFi-GAN LARGE	13.94M	40.57	96.61	x 0.010 / 0.04
WaveGrad* SMALL	4.18M	890.44	3398.11	x 0.157 / 2.23
WaveGrad MEDIUM	17.12M	3498.64	5439.46	x 0.247 / 3.82
WaveGrad* LARGE	33.91M	8522.08	7833.62	x 0.348 / 4.45
DiffWave SMALL	1.23M	79.43	769.82	x 0.041 / 1.14
DiffWave MEDIUM	2.62M	255.78	1458.17	x 0.070 / 3.47
DiffWave LARGE	6.89M	899.61	2937.61	x 0.135 / 5.47
WaveFlow SMALL	5.95M	852.85	599.95	x 0.033 / 0.34
WaveFlow MEDIUM	12.86M	3419.39	1102.88	x 0.055 / 0.62
WaveFlow LARGE	22.39M	6063.60	1408.01	x 0.069 / 1.05
WaveGlow SMALL	17.56M	45.84	181.98	x 0.015 / 0.13
WaveGlow MEDIUM	34.76M	116.21	496.72	x 0.019 / 0.21
WaveGlow LARGE	87.73M	333.04	283.07	x 0.028 / 0.51

Table 8: Comparison of computation and energy footprints of various generative models for speech synthesis conditioned on mel-spectrogram. (*) indicate configurations that we suggest in addition to those of the original papers.

heavily on the hardware, and results on other types of processors such as embedded devices can be significantly different due to their lighter architectures. We also confirm that GFLOPs are positively correlated to the energy cost, while the number of parameters is not a good indicator of efficiency, as shown by the absence of correlation in Figure 33.

5.2.4 *Pareto analysis*

In order to fully understand the tradeoff between quality and energy impact, we display our proposed multi-objective analysis in Figure 34. We separate this analysis between the inference energy costs (top), the hardware-agnostic metric GFLOPs (bottom). In both cases, we plot different models depending on their corresponding MOS evaluation. We depict the optimal Pareto models, which are circled in red. A first noticeable result of this study is that our multi-objective analysis produces coherent results, since we can directly find optimal models with low energy

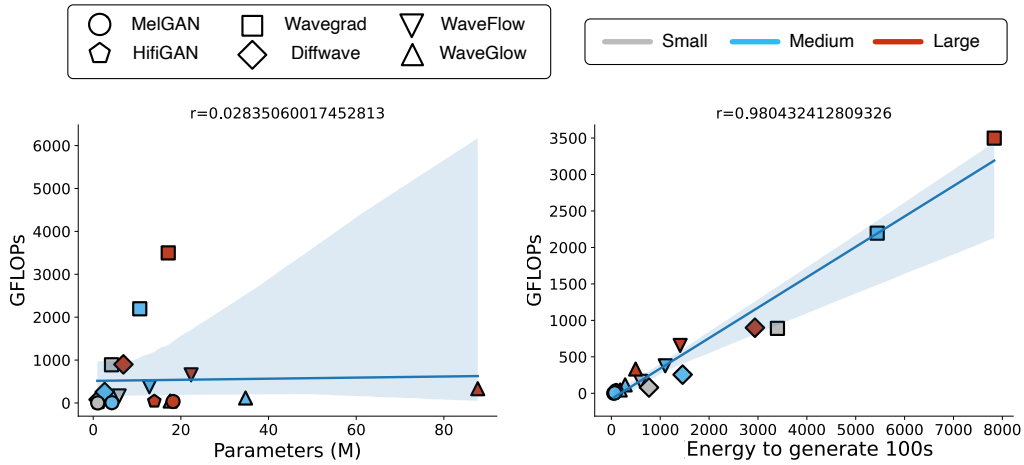


Figure 33: Correlation between GFLOPs and (left) the number of parameters of the network, (right) the energy required generate 100s, in mJ.

consumption but high quality score. A second result of this analysis is that only few models are included inside the Pareto front, with the vast majority of models being dominated in both aspects simultaneously. This means that our proposed multi-objective approach allows to efficiently discriminate between different models on both their audio quality and energy impact. A third key component of this study is that the hardware-agnostic metric and the GPU metric are consistent, with slight shifts showing that energy and GFLOPs are not linearly correlated.

If we take a closer look at the per-model inference tradeoff (by considering only the same symbol), we can see that it also forms what we can call sub-Pareto front, from lighter to larger configurations (from light green to dark blue), but it is only when we look at the big picture that it reveals disparities of generative models architectures and configurations. Hence, our analysis allows to raise attention and provide new keys for researchers to evaluate their models within the context of a multi-objective analysis rather than comparing quality and efficiency separately. Furthermore, we believe that it is through this research effort that we will be able to achieve a more sustainable computing.

5.3 CONCLUSION

In this chapter, we proposed a large-scale evaluation of neural vocoders while integrating their energy footprint. We relied on six stat-of-the-art models and evaluate their quality according to three different configurations from lighter to larger architectures. Then, we proposed a multi-objective analysis of both quality from human-based evaluation (MOS) and energy consumption. Within this framework, we showed that this energy footprint must be linked to the model perceptual quality and that, small models can perform better than larger and more costly models. We believe this is the first attempt to integrate both energy consumption and quality in neural audio synthesis models, taking a step forward against blind eval-

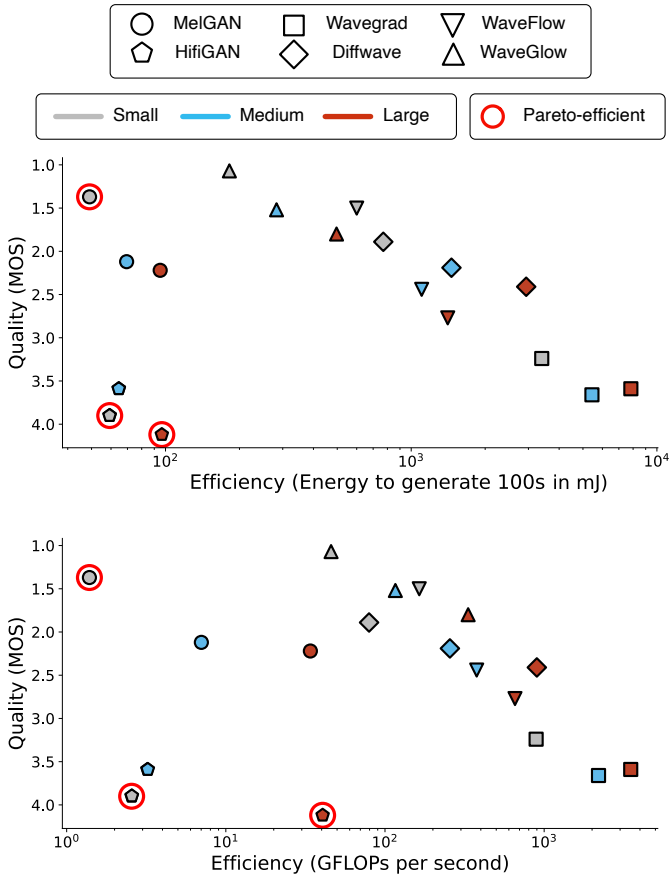


Figure 34: Representation of Pareto Frontier for efficiency vs quality. The objective is to maximize the quality (MOS) and minimize the energy cost of inference (top) and the number of GLFOPs (bottom).

uations that only take into account audio quality. This, in the future, can become increasingly important, since lightweight models are fundamental for real-time embedded systems. It should be noted that our approach is generic and could be applied to any type of model or input data.

6

PERSPECTIVES : LOWERING THE ENERGY COST

In the previous chapters, we have proposed a new methodology to put computational costs at the same level as quality and we have conducted a validation of these concepts trough a large-scale evaluation of neural vocoders. In this chapter, we intend to lower the inference cost of those networks by applying compression techniques through the use of quantization. First, we present the motivation of this technique in Section 6.1 and expose its formalism in Section 6.2. Second, we run experiments on our neural vocoders benchmark and apply the method of *quantization-aware* training in Section 6.3. Third, we conclude this chapter with a discussion on the possibilities to embed these systems, which can be valuable for audio synthesis, and present other types of compression techniques in Section 6.4.

6.1 MOTIVATIONS

In deep learning, there are several ways to compress neural networks (Menghani, 2021). Quantization is one of these techniques and consists in reducing the precision of a model weights and activations. This is typically done by storing the model parameters as lower-precision numbers, such as 16-bit or 8-bit integers instead of 32-bit (or 64-bit) floating-point numbers. The benefits of this quantification is that it leads to more efficient memory usage and computation time, as lower bit-precision implies fewer bits to store and manipulate. Hence, this and can also directly lead to a reduction in energy consumption. Additionally, faster inference time is also beneficial for real-time applications like audio synthesis. By reducing the computational requirements of the model, quantization allows the model to be more easily deployed on embedded devices, making it possible to create more powerful tools for audio synthesis.

One of the main inconvenient of quantized neural networks is that they suffer from a loss of performance compared to full precision networks. Indeed, reducing the bit-precision of the weights and activation introduces quantization errors, which can degrade the performance of the network. Also, the energy savings from using quantized neural networks can depend on the specific application and the details of the implementation. In some cases, the trade-off between accuracy and efficiency might even worsen, and the energy savings may not be significant enough to justify the use of quantized neural networks. Therefore, our proposed methodology based on Pareto optimality could help us find suitable models for a given task and will be the subject of our experiments in the following.

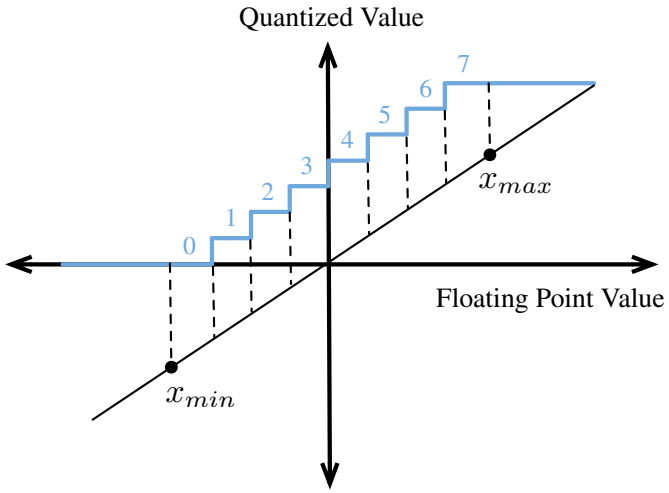


Figure 35: Example of quantization, where the continuous values are clamped between x_{min} and x_{max}. These values are mapped to discrete values in the range $[0, 2^b - 1]$ where here, $b = 3$. Figure from Menghani, 2021.

6.2 FORMALISM

Formally, let us consider a floating-point value $x \in [x_{min}, x_{max}]$, that we want to quantize to b-bit integer $x_q \in [0, 2^b - 1]$. The quantization process is done using the formula

$$\text{quantize}(x) = x_q = \text{round}\left(\frac{1}{s}x + z\right) \tag{50}$$

where s is a positive floating-point value called the *scale*, and z is an integer called the *zero point*. The scale s determines the range of values that can be represented by the quantized values, and is defined by

$$s = \frac{x_{max} - x_{min}}{2^b - 1} \tag{51}$$

The zero point z is an integer, which defines what quantification value is assigned to $x = 0.0$. It is chosen to ensure an equal spacing between the quantization levels within the range of floating-point values. Then, the dequantization process transforms the value x_q back to a floating-point value x such that

$$\text{dequantize}(x_q) = x = s(x_q - z) \tag{52}$$

An example of quantization is shown in Figure 35.

6.2.1 *Types of quantization*

There are three main types of quantized neural networks available in *PyTorch*, which is our main development framework

1. *Dynamic quantization*: only the weights are quantized in the training, while activations are quantized during inference, which is useful for models such as LSTM, where the execution time is dominated by loading weights from memory rather than computing the matrix multiplications.

2. *Static quantization*: both weights and activations are quantized before inference. Thus, this requires calibration with a representative dataset to optimize quantization parameters (scale and bias). This is typically used for CNN where both memory and computation savings are important.

3. *Quantization Aware Training (QAT)*: this is a method that takes into account the effects of quantization during training, resulting in higher accuracy compared to other quantization methods. In QAT, all calculations are done in floating point, with fake quantization modules simulating the effects of quantization by clamping and rounding from this operation

$$x = \text{dequantize}(\text{quantize}(x)) \tag{53}$$

   After training, weights and activations are quantized, with CNNs being a typical use case.

For more details on quantization, please refer to the supporting web page[1].

6.3 APPLICATION TO NEURAL VOCODERS

Quantization has not yet been widely applied to generative models. However, a few studies in the field of computer vision have demonstrated its potential to significantly reduce model size while maintaining a high level of quality (Andreev, Fritzler, and Vetrov, 2021; Wan et al., 2020). Despite this promising research, there have been no studies on the use of quantization for generative models for audio synthesis. Therefore, this is a first effort in the literature to explore the potential of this technique for audio synthesis.

6.3.1 *Experiments*

First, we rely on our benchmark of neural vocoders from the previous chapter as a baseline and try to apply static quantization. Indeed, most of our networks use CNN and, thus, are not suitable for dynamic quantization. In Table 9, we show the gains in memory size of performing quantization in int8 on neural vocoders. As we can see, quantization is able to reduce the memory size by a factor of approximately 4. Quantization is usually believed to speed up inference by a factor of 2-4 (Krishnamoorthi, 2018), depending on the hardware, so we also tried to compute potential gains in inference speed. Unfortunately, the quantization back-end

1 https://pytorch.org/blog/introduction-to-quantization-on-pytorch/

in PyTorch is still in development and we found out that it was limited to simple models based on a succession of simple layers. More advanced models, such as our neural vocoders, have intricate layers that are currently not supported by Pytorch quantization. This means that we were not able to use quantized models from our benchmark in inference, and thus to measure the latency improvements they may offer. However, as the feature is still under development, it is likely that it will be improved in the coming months, allowing the use of quantized models in more complex cases.

Model	**Size fp32**	**Size int8**
MelGAN* SMALL	4.2MB.	1.1MB.
MelGAN MEDIUM	17.1MB.	4.3MB.
MelGAN* LARGE	72.9MB.	18.3MB.
HiFi-GAN MEDIUM	3.8MB.	1MB.
HiFi-GAN SMALL	5.9MB.	1.5MB.
HiFi-GAN LARGE	55.8MB.	14.1MB.
WaveGrad* SMALL	16.8MB.	16.3MB.
WaveGrad MEDIUM	42.5MB.	41.1MB.
WaveGrad* LARGE	68.5MB.	66.1MB.
DiffWave SMALL	5MB.	1.4MB.
DiffWave MEDIUM	11MB.	2.8MB.
DiffWave LARGE	27.6MB.	7.1MB.
WaveFlow SMALL	24MB.	6.3MB.
WaveFlow MEDIUM	51.7MB.	13.2MB.
WaveFlow LARGE	89.8MB.	22.8MB.
WaveGlow SMALL	70.6MB.	18MB.
WaveGlow MEDIUM	139.4MB.	35.3MB.
WaveGlow LARGE	351.7MB.	88.5MB.

Table 9: Impact on memory sizes of quantizing both weights and activations of pervasive neural vocoders. (*) indicate configurations that we suggest in addition to those of the original papers.

Despite those challenges, we wanted to investigate the impact of applying a fake quantization technique during the training, as developed in the QAT method. Specifically, we were interested in the effect this method could have on generative audio models and the potential loss of precision resulting from the quantization/dequantization process. Therefore, we keep our training routine of Section 5.1, with the exact same configurations and adapt it to automatically insert the fake quantization module between all layers of the models thanks to the PyTorch-lightning

Callback module. We also reduce the training time to 2 days only (instead of 5 days). This not only allows for faster experimentation, but also has the advantage of reducing energy consumption.

6.3.2 *Results*

During our experiments, only the DiffWave model achieved stable training with quantization modules and, thus, will be the focus of the following analysis. We depict in Figure 36 the evolution of the validation loss from the different trainings.

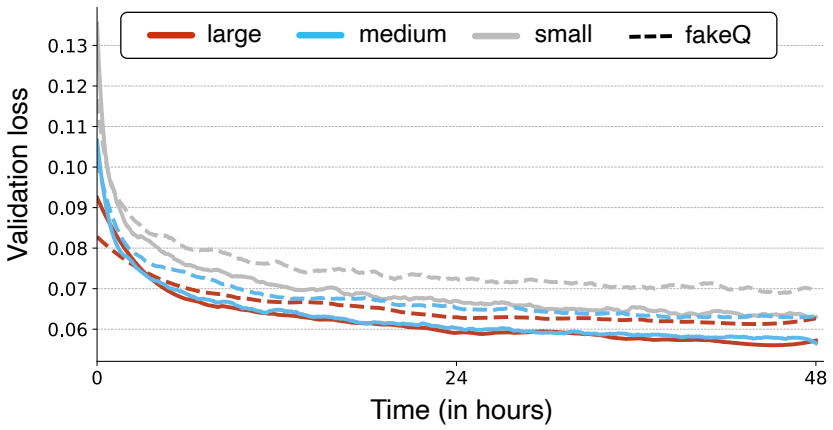


Figure 36: Evolution of the validation loss from the diffwave model when train in full precision (plain lines) and with fake-quantization modules (dotted lines).

As expected, the validation loss is higher in all three configurations using fake quantization modules rather than the full precision model. However, when looking at the monitored metrics during training from Figure 37, we were surprised to find that applying these fake quantization modules improves metrics such as MOSNet and the Narrow Band PESQ (NBPESQ) metric. This can be interpreted by the fact that adding these fake quantization modules simplifies the learning problem, as it implicitly turns regression (continuous) into classification (discrete), which could allow the model to better generalize and improve overall performance.

To confirm our intuition, we compare the spectrogram of the original signal (Figure 38a) to the full-precision (Figure 38b), and fake-quantized (Figure 38c) reconstructions signals. As we can see, although the fake quantization model has more background noise the overall spectrogram is visually closer to the original one. This confirms that fake quantized model can rightfully obtain higher MOSNet and NBPESC scores. Through a listening inspection, we can hear that results are more blurry, but the voice comprehensibility seem enhanced. Unfortunately, due to issues with the PyTorch Backend, we were unable to generate samples using the full quantized model following this QAT. However, based on this promising preliminary results, we believe it is worthwhile to continue exploring this approach,

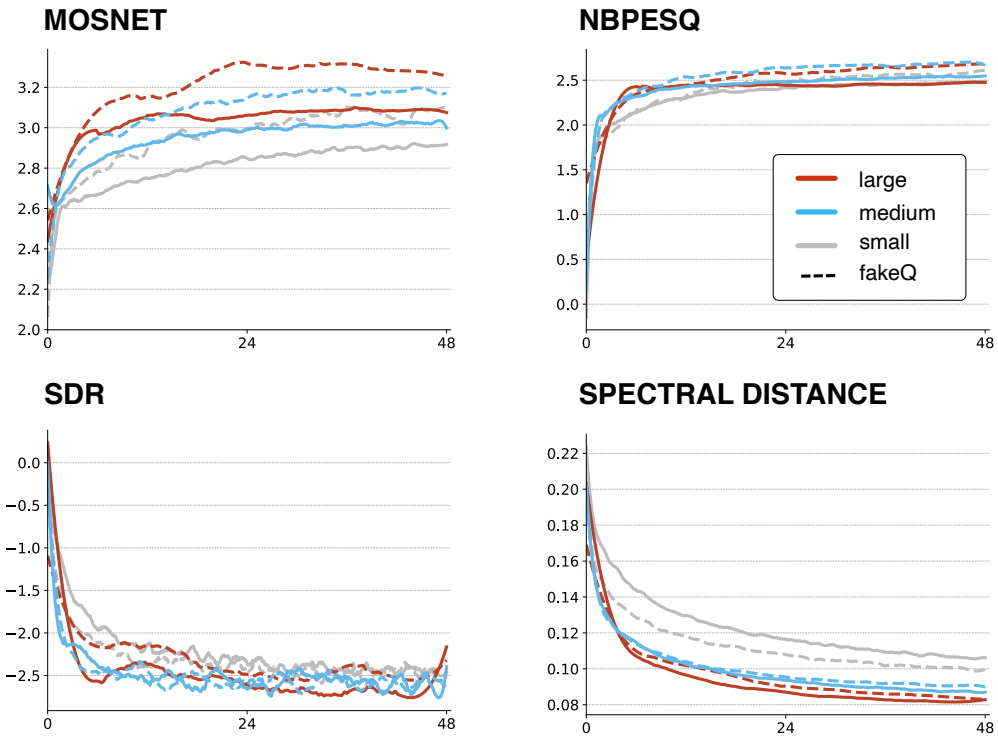


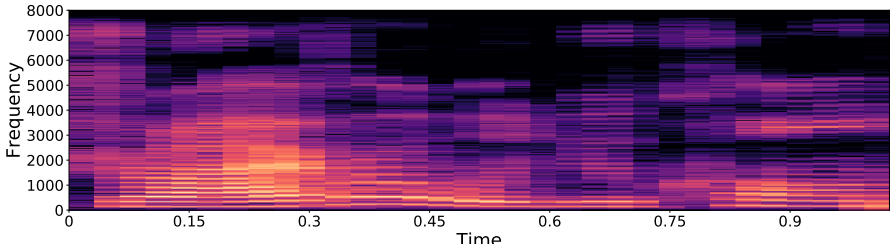
Figure 37: Evolution of the metrics computed in the validation phase of the diffwave model in Quantization-Aware Training mode. In plain lines, results from the full precision model and in dotted lines, results from the fake quantized model. The x-axis represents the time (expressed in hours).

and we are optimistic that with further development, we will be able to produce high-quality samples using quantized model.

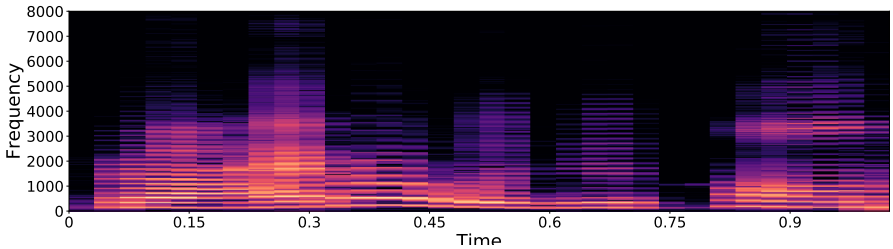
6.4 EMBEDDING DEEP GENERATIVE AUDIO

As discussed earlier, one of the goal of quantization approaches is that we could obtain deep audio models that fit on embedded hardware, such as Arduino and Raspberry Pi, both of which are popular platforms for building electronic projects. Arduino are microcontroller boards, which means that they are designed to execute a single program repeatedly. They are well-suited for projects that require real-time processing or interaction with physical devices such as sensors or motors. Raspberry Pi, on the other hand, is a single-board computer, which means it is a fully-functional computer that can run an operating system and execute multiple programs. As a result, Raspberry Pi is better suited for projects that require more computing power or that need to run more complex software. Both platforms have very strong computational constraints, as summarized in Table 10[2].

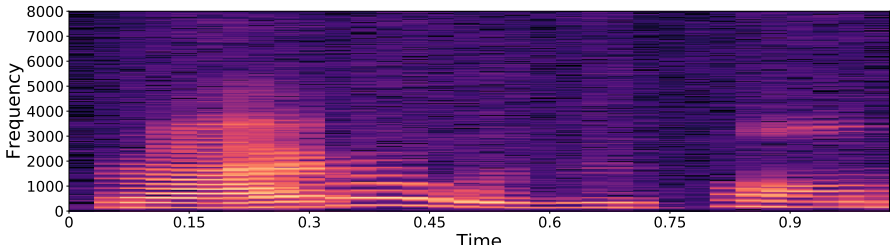
2 These properties were gathered from the user manuals and the FLOPS are inferred from the listed CPU properties



(a) Original



(b) Reconstruction (Full Precision)



(c) Reconstruction (Fake Quantized)

Figure 38: Representation of the spectrograms from the original and reconstruction from (b) the full precision network and (c) the model with fake quantized modules.

In those specifications, the *clock frequency* is a measure of the speed at which the device can execute instructions, the *GFLOPS* is a measure of the number of floating-point operations that can be performed in one second, *RAM* is the memory used for temporary storage of data currently in use by the device, and the *Drive* is a type of storage device used for long-term storage of data.

When comparing those properties to the GFLOPs from Table 8, it appears that our neural vocoder models are still quite far from being able to run on highly-constrained hardware such as Arduino. Specifically, the memory and FLOPS usage is significantly higher than what the platform can handle. However, it seems that all models could fit on a Raspberry Pi in terms of size, but they may not be able to produce real-time sound synthesis. For example, the MelGAN medium model uses 7.02 GFLOPs to generate 1 second of audio. On a Raspberry Pi 4, it would take $7.02/9.92 \sim 0.7$ seconds, which allows to perform real-time synthesis. Other models such as Hifi-GAN large would take $40.47/9.92 \sim 4$ seconds to produce 1

Model	CPU	FLOPS	RAM	Drive
Arduino				
ATMega1280	16 MHz	128K.	8 KB	128 KB
ATMega2560	32 MHz	256K.	16 KB	256 KB
Raspberry Pi				
RPi 1B	700 MHz	213M.	512MB	256MB
RPi 2B	900 MHz	1.47G.	1GB	1GB
RPi 3B	1.2 GHz	3.62G.	1GB	32GB
RPi 4B	1.5/1.8 GHz	9.92G.	1-2-4-8GB	Micro-SD

Table 10: Properties of different *Arduino* micro-controllers and *Raspberry Pi* embedded platform

second of audio, making them unusable in real-time setups. Quantization could potentially address this issue as it allows for a 2-3x reduction in inference time, but the corresponding Million Instructions per Second (MIPS) (i.e. the similar metric but for integers) depend on the types of operations and algorithm optimization, thus making it difficult to compute theoretical inference times.

6.5 CONCLUSION

In this chapter, we attempted to apply quantization to both the weights and activations of our neural vocoder benchmark, but we were limited by the PyTorch quantization back-end, which does not support the use of complex models such as our neural vocoders. Then, we explored the impact of using a fake quantization technique during training on the diffwave model and found that it considerably improved the comprehensibility of the voice, while adding background noise to the generated samples. Finally, we discussed the possibility of embedding these models in constrained architectures such as Arduino and Raspberry Pi, by testing their properties against the requirements of the architectures. We believe that further development of this quantization approach could produce high-quality samples, while lowering the cost of inference and increasing the potential for embedding deep audio models into highly constrained platforms.

7 CONCLUSION

To conclude our work, we begin by summarizing all of our contributions in Section 7.1. We then present the various axes for future work arising from our research in Section 7.2. Finally, in Section 7.3, we provide an overall conclusion on the work conducted throughout this thesis.

7.1 SUMMARY AND MAIN CONTRIBUTIONS

First, in Chapter 1, we introduced the manuscript by discussing the challenges raised by the widespread adoption of digital technologies, specifically AI algorithms, on the environment. In this work, we focused on neural audio synthesis and the specifics of generative models for audio that are now widely used for voice and music applications but have not yet been assessed in terms of computational costs and environmental impact. In 2, we exposed the core concepts or machine learning starting from the general theoretical notions up to more sophisticated tools such as generative models and their application to audio synthesis. We indicated the need of having robust and adequate evaluation measures to compare the performance of models and presented an exhaustive review of quality and efficiency metrics used in the literature in 3. We ended this background theory part with a presentation of multi-objective evaluation from Pareto, a core component we use in the manuscript.

We presented our first key contribution in Chapter 3, where we performed a large-scale analysis of the use of evaluation metrics in the generative audio models literature. Since carbon emissions were never addressed before, we evaluated the energy cost (and thus the carbon cost) of training state-of-the-art models by considering factors such as hardware and training time. We emphasized the importance of linking energy footprint with model quality and proposed a new methodology based on Pareto optimality, which considers both factors equally. Our results validated that this 2D representation makes it easier to identify optimal models on both criteria simultaneously.

Then, in Chapter 5, we exposed our second key contribution with a large-scale experiment on generative models for voice generation, known as neural vocoders. We considered six state-of-the-art models using three different configurations and developed a consistent environment to run our experiments and monitor training convergence. We performed a human-based perceptual quality evaluation and computed a wide range of quality and efficiency metrics. Through the application of our multi-objective method, we founded that our approach allows for the iden-

tification of optimal models facilitating the overall evaluation of research across multiple objectives simultaneously.

Finally, in Chapter 6, we presented one perspective to lower the energy cost based on quantizated neural networks. We explored the impact of using fake quantization modules during training and found that it can considerably improve the intelligibility of the voice while adding background noise to the generated samples. We also discussed the possibility of embedding these models in constrained architectures.

7.2 FUTURE WORKS

In future works, it would be interesting to expand the scope of our evaluation and include a wider range of generative audio models and applications, such as unconditional generative models and other types of audio signals other than voice. Additionally, to further validate the results from quantization on neural vocoders, it would be beneficial to conduct a more in-depth analysis of energy consumption by testing on real embedded hardware and applying our methodology to find models that maximize quality under the constraints of the target platform. Similarly, it would be interesting to scrutinize the energy savings of other compression techniques, such as pruning, as a follow-up of our work Esling et al., 2020a. A possible experiment could be to evaluate the impact of combining these two compression techniques on the Pareto optimality of neural audio synthesis. Furthermore, one other potential topic of interest is the exploration of newer neural network architectures, such as spiking neural networks, that have the potential to consume a largely smaller amount of energy. However, it is important to keep in mind the potential rebound effect which can offset the energy savings.

As we highlighted along this thesis, the evaluation of generative audio models is a complex task that should take into account both the quality of the generated audio and the energy consumption of the model. However, we focused on the training and inference costs of deep neural networks, omitting the entire life cycle of the technology, specifically the cost of producing the hardware. This includes, graphics processing units for training and embedded systems for inference. This aspect represents a significant contributor to the environmental impact of AI. Further experiments on this aspect would be worth conducting to fully understand and mitigate the environmental impacts of deep generative models for audio synthesis.

7.3 OVERALL CONCLUSION

The goal of this thesis was to evaluate the environmental impact of deep generative models for audio synthesis, specifically by examining the energy consumption of some generative models and integrate those measurements in the evaluation procedure. While the field of audio synthesis alone is not a major environmental concern, it is crucial to work towards more sustainable practices. Therefore, we hope that our proposed evaluation method and results will provide a useful guide for future researches in this field. The multi-objective evaluation approach presented in the thesis is intended to be adaptable to other scientific fields, and to encourage more environmentally-friendly research.

Part III

APPENDIX

APPENDIX A

Model	Quality						Efficiency				
	MOS	Recon	LL	IS	FID	NDB	RTF	Param	Train. Time	Size	FPO
DDSP		•						•			
Diffwave	•			•	•	•	•	•			
FloWaveNet	•		•				•		•		
GanSynth	•			•	•	•	•		•		
Hifi-GAN	•						•	•			
Jukebox		•							•		
MelGAN	•						•	•			
NSF	•	•			•		•	•			
Parallel WaveNet	•						•				
RAVE	•						•	•	•		
SampleRNN	•		•	•							
Sashimi	•		•	•	•			•			
SING	•						•		•	•	
WaveFlow	•		•				•	•			
WaveGan	•			•							
WaveGlow	•						•				
WaveGrad	•	•					•	•			
WaveNet	•										
WaveNet AE		•									

Table 11: Occurrences of evaluation metrics used in neural audio synthesis research literature.

B

APPENDIX

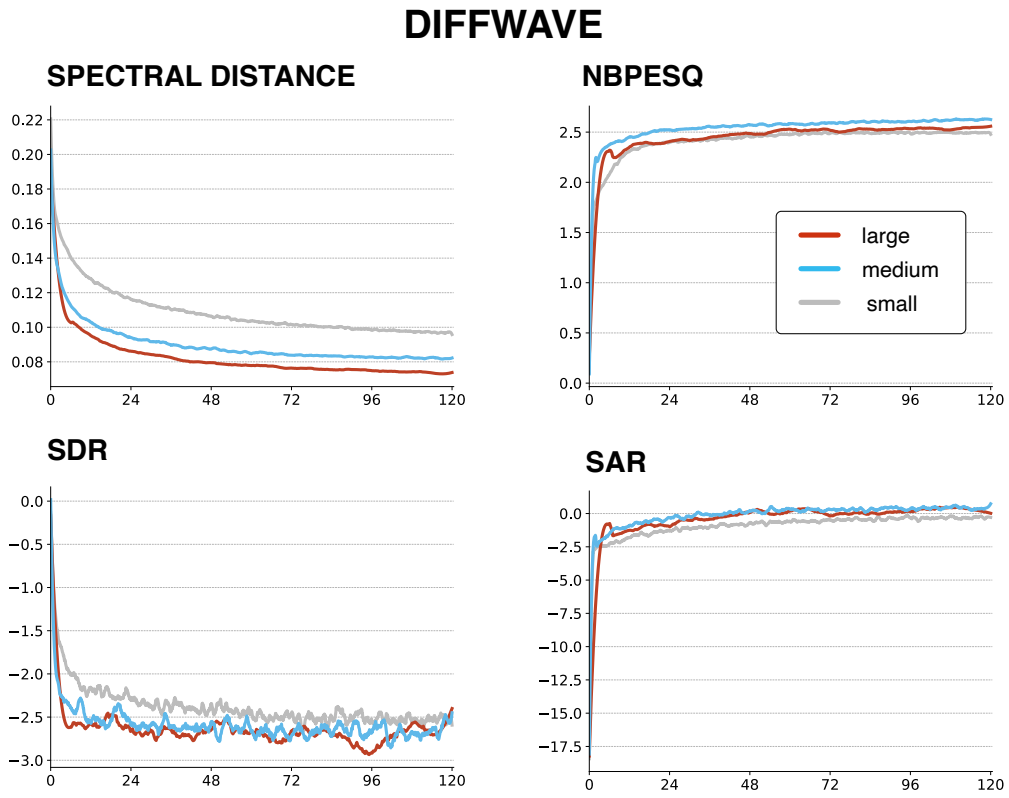


Figure 39: Evolution of the metrics computed in the validation phase of the diffwave model training. The x-axis represents the time (expressed in hours).

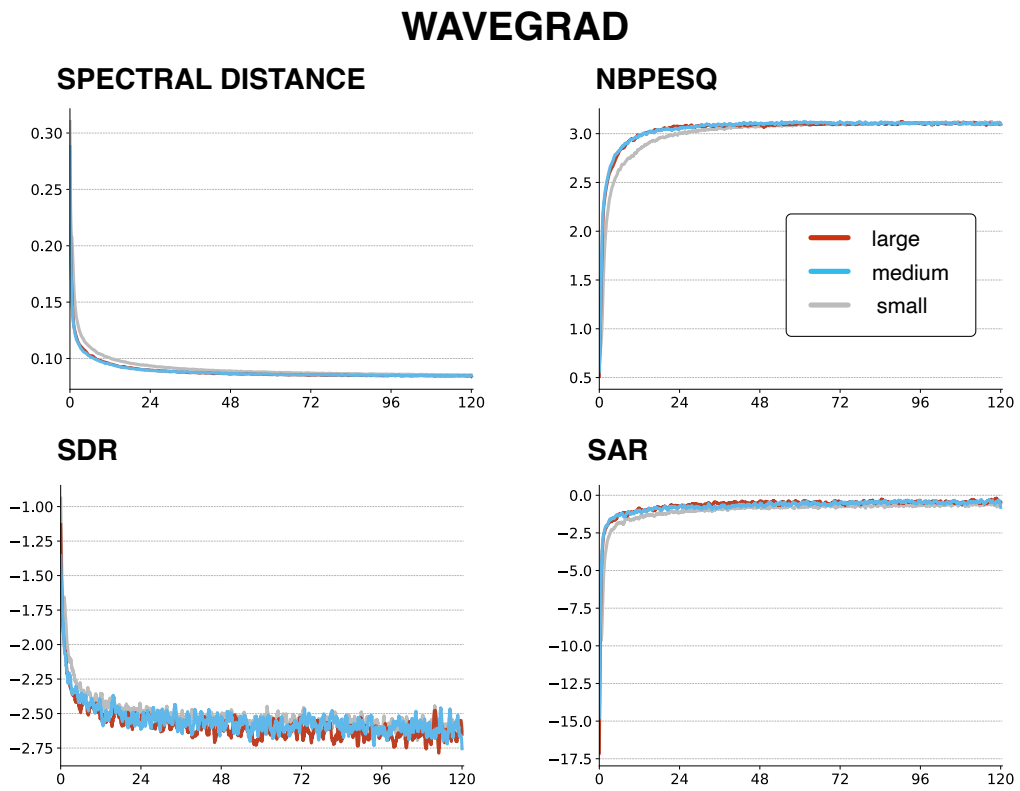


Figure 40: Evolution of the metrics computed in the validation phase of the wavegrad model training. The x-axis represents the time (expressed in hours).

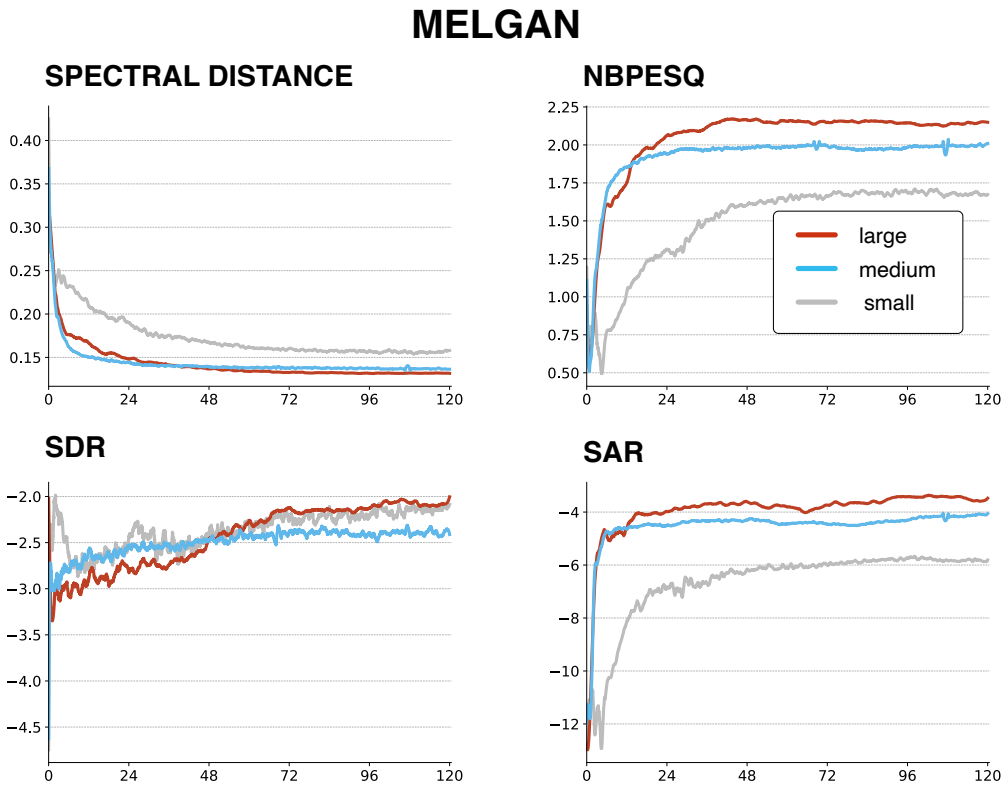


Figure 41: Evolution of the metrics computed in the validation phase of the melgan model training. The x-axis represents the time (expressed in hours).

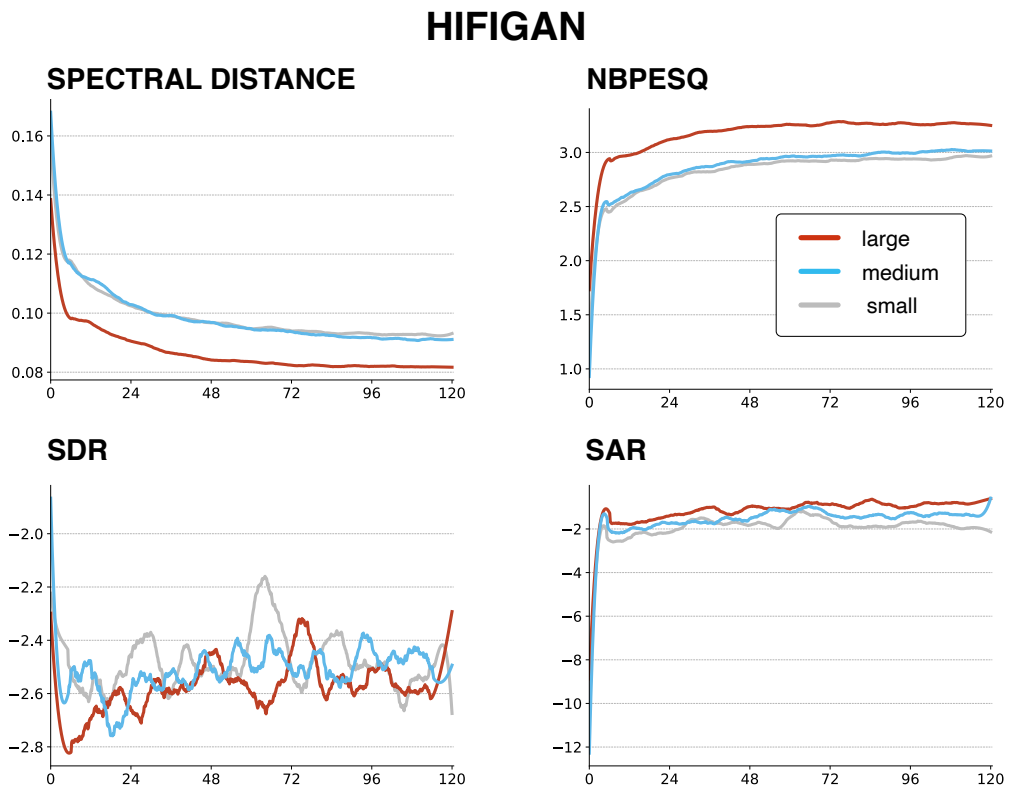


Figure 42: Evolution of the metrics computed in the validation phase of the hifigan model training. The x-axis represents the time (expressed in hours).

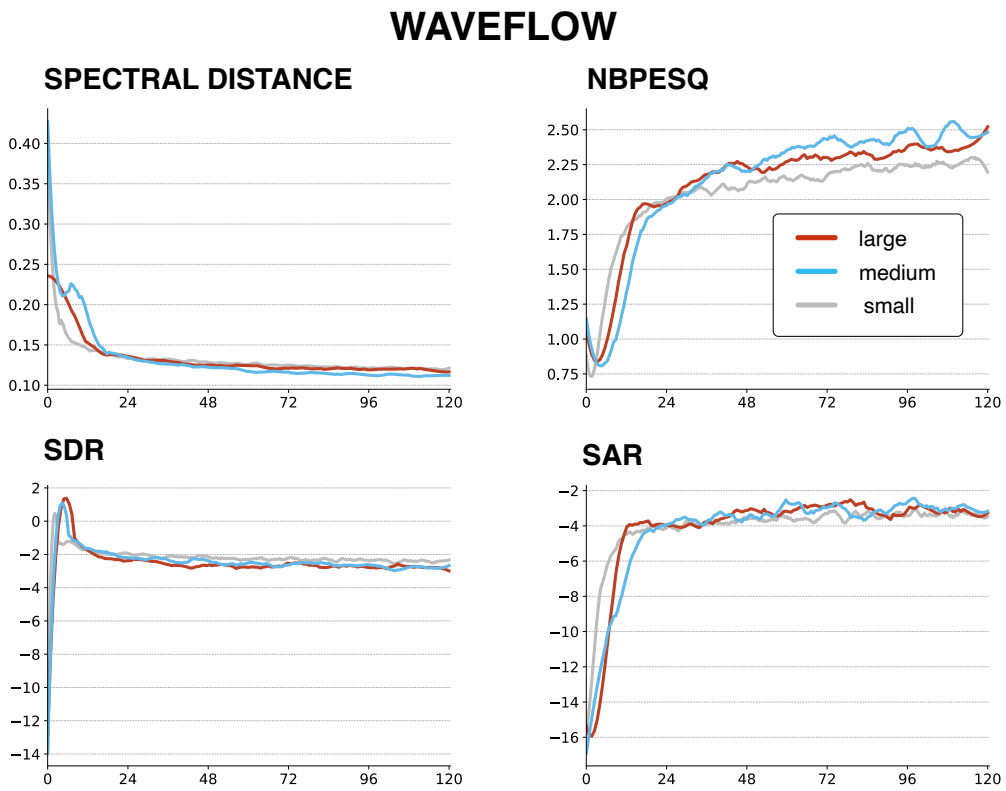


Figure 43: Evolution of the metrics computed in the validation phase of the waveflow model training. The x-axis represents the time (expressed in hours).

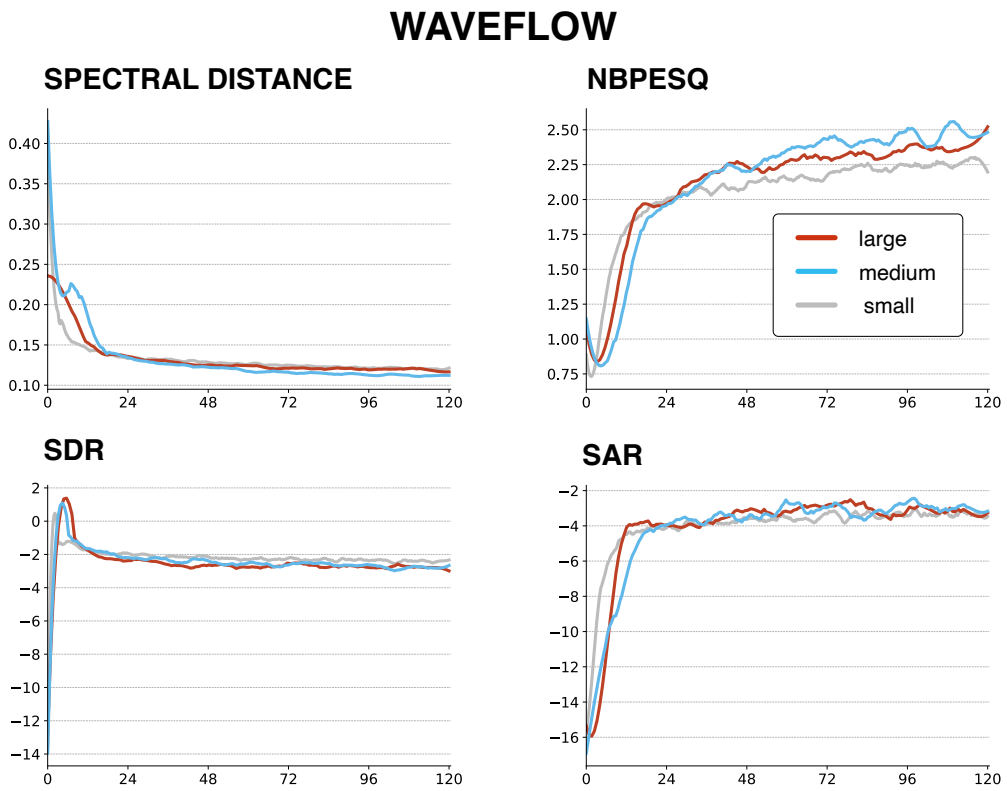


Figure 44: Evolution of the metrics computed in the validation phase of the waveglow model training. The x-axis represents the time (expressed in hours).

BIBLIOGRAPHY

Andreev, Pavel, Alexander Fritzler, and Dmitry Vetrov (2021). "Quantization of Generative Adversarial Networks for Efficient Inference: a Methodological Study." In: (cit. on p. 75).

Anthony, Lasse F. Wolff, Benjamin Kanding, and Raghavendra Selvan (2020). "Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models." In: *arXiv* (cit. on pp. 6, 45, 55).

Aouameur, Cyran, Philippe Esling, and Gaetan Hadjeres (2019). "Neural drum machine: An interactive system for real-time synthesis of drum sounds." In: *Proceedings of the 10th International Conference on Computational Creativity, ICCC 2019*, pp. 92–99 (cit. on p. 10).

Bazin, Théis and Gaëtan Hadjeres (2019). "NONOTO: A model-agnostic web interface for interactive music composition by inpainting." In: *Proceedings of the 10th International Conference on Computational Creativity, ICCC 2019*, pp. 89–91 (cit. on p. 10).

Bengio, Yoshua, Patrice Y Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult." In: *IEEE transactions on neural networks* 5 2, pp. 157–166 (cit. on p. 24).

Briot, J P, G Hadjeres, and F D Pachet (2019). *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer International Publishing. ISBN: 9783319701622 (cit. on p. 9).

Brown, Tom B. et al. (2020). "Language models are few-shot learners." In: *Advances in Neural Information Processing Systems* 2020-Decem (cit. on p. 5).

Caillon, Antoine and Philippe Esling (2021). "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis." In: (cit. on pp. 31, 53).

Che, Tong, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li (2017). "Mode regularized generative adversarial networks." In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–13 (cit. on p. 40).

Chen, Ling Hui, Zhen Hua Ling, Li Juan Liu, and Li Rong Dai (2014). "Voice conversion using deep neural networks with layer-wise generative training." In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 22.12, pp. 1859–1872 (cit. on p. 10).

Chen, Nanxin, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan (2020). "WaveGrad: Estimating Gradients for Waveform Generation." In: pp. 1–15 (cit. on pp. 34, 61, 62).

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). "On the properties of neural machine translation: Encoder–decoder approaches." In: *Proceedings of SSST 2014 - 8th Workshop on Syntax, Semantics*

and Structure in Statistical Translation, pp. 103–111. ISBN: 9781937284961 (cit. on p. 24).

Chowning, John M (1973). "The synthesis of complex audio spectra by means of frequency modulation." In: *Journal of the audio engineering society* 21.7, pp. 526–534 (cit. on p. 9).

Défossez, Alexandre, Neil Zeghidour, Nicolas Usunier, Léon Bottou, and Francis Bach (2018). "Sing: Symbol-to-instrument neural generator." In: *Advances in Neural Information Processing Systems* 2018-Decem.Nips, pp. 9041–9051 (cit. on pp. 34, 52).

Dhariwal, Prafulla, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever (2020). "Jukebox: A Generative Model for Music." In: (cit. on pp. 30, 53).

Donahue, Chris, Julian McAuley, and Miller Puckette (2019). "Adversarial audio synthesis." In: *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–16 (cit. on pp. 31, 53).

Eckel, Gerhard (1995). "Sound synthesis by physical modelling with Modalys." In: *Proc. ISMA'95*, pp. 478–482 (cit. on p. 9).

Engel, Jesse, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts (2019). "Gansynth: Adversarial neural audio synthesis." In: *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–17 (cit. on p. 53).

Engel, Jesse, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan (2017). "Neural audio synthesis of musical notes with WaveNet autoencoders." In: *34th International Conference on Machine Learning, ICML 2017* 3, pp. 1771–1780 (cit. on p. 34).

Esling, Philippe, Axel Chemla–Romeu-Santos, and Adrien Bitton (2018). "Generative timbre spaces: Regularizing variational auto-encoders with perceptual metrics." In: *DAFx 2018 - Proceedings: 21st International Conference on Digital Audio Effects*, pp. 369–376 (cit. on pp. 10, 30).

Esling, Philippe, Ninon Devis, Adrien Bitton, Antoine Caillon, Axel Chemla-Romeu-Santos, and Constance Douwes (2020a). "Diet Deep Generative Audio Models With Structured Lottery." In: *Proceedings of the International Conference on Digital Audio Effects, DAFx* 1, pp. 317–324 (cit. on pp. 43, 82).

Esling, Philippe, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos (2020b). "Flow synthesizer: Universal audio synthesizer control with normalizing flows." In: *Applied Sciences (Switzerland)* 10.1, pp. 1–11 (cit. on p. 10).

Faviola Rodrigues, Crefeda, Graham Riley, and Mikel Luján (2018). "SyNERGY: An energy measurement and prediction framework for Convolutional Neural Networks on Jetson TX1." In: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pp. 375–382 (cit. on p. 42).

Fukushima, Kunihiko (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." In: *Biological Cybernetics* 36.4, pp. 193–202 (cit. on p. 3).

Goel, Karan, Albert Gu, Chris Donahue, and Christopher Ré (2022). "It's Raw! Audio Generation with State-Space Models." In: pp. 1–23 (cit. on p. 28).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets." In: *Advances in Neural Information Processing Systems*. Vol. 3. January (cit. on p. 30).

Griffin, Daniel and Jae Lim (1984). "Signal estimation from modified short-time Fourier transform." In: *IEEE Transactions on acoustics, speech, and signal processing* 32.2, pp. 236–243 (cit. on p. 8).

Gupta, Udit, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien Hsin S. Lee, Gu Yeon Wei, David Brooks, and Carole Jean Wu (2022). "Chasing Carbon: The Elusive Environmental Footprint of Computing." In: *IEEE Micro* 42.4, pp. 37–47 (cit. on p. 4).

Henderson, Peter, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau (2020). "Towards the systematic reporting of the energy and carbon footprints of machine learning." In: *Journal of Machine Learning Research* 21, pp. 1–25 (cit. on p. 4).

Hernandez-Olivan, Carlos, Javier Hernandez-Olivan, and Jose R. Beltran (2022). "A Survey on Artificial Intelligence for Music Generation: Agents, Domains and Perspectives." In: pp. 1–26 (cit. on p. 10).

Herre, Jürgen and Sascha Dick (2019). "Psychoacoustic models for perceptual audio coding-A tutorial review." In: *Applied Sciences (Switzerland)* 9.14 (cit. on p. 42).

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter (2017). "GANs trained by a two time-scale update rule converge to a local Nash equilibrium." In: *Advances in Neural Information Processing Systems* 2017-Decem.Nips, pp. 6627–6638 (cit. on p. 40).

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models." In: *Advances in Neural Information Processing Systems* 2020-Decem.NeurIPS 2020, pp. 1–25 (cit. on pp. 33, 34).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory." In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 24).

Hubel, D H and T N Wiesel (1959). "Receptive fields of single neurones in the cat's striate cortex." In: *The Journal of Physiology* 148.3, pp. 574–591 (cit. on p. 3).

Ito, Keith (2017). *The LJ Speech Dataset*. URL: https://keithito.com/LJ-Speech-Dataset/ (cit. on p. 63).

Kendall, Maurice G and Alan Stuart (1953). "The Advanced Theory of Statistics, Vol. 1: Distribution Theory." In: *Journal of the Royal Statistical Society, Series B* 15, pp. 187–236 (cit. on p. 38).

Kilgour, Kevin, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi (2019). "Fréchet audio distance: A reference-free metric for evaluating music enhance-

ment algorithms." In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 2019-Septe, pp. 2350–2354 (cit. on p. 40).

Kim, Sungwon, Sang Gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon (2019). "FloWaveNet: A generative flow for raw audio." In: *36th International Conference on Machine Learning, ICML 2019* 2019-June, pp. 5852–5860 (cit. on pp. 32, 53).

Kingma, Diederik P. and Jimmy Lei Ba (2015). "Adam: A method for stochastic optimization." In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15 (cit. on p. 64).

Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1×1 convolutions." In: *Advances in Neural Information Processing Systems* 2018-Decem, pp. 10215–10224 (cit. on p. 63).

Kong, Jungil, Jaehyeon Kim, and Jaekyoung Bae (2020). "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis." In: *Advances in Neural Information Processing Systems* 2020-Decem.NeurIPS (cit. on pp. 31, 61, 62).

Kong, Zhifeng, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro (2020). "DiffWave: A Versatile Diffusion Model for Audio Synthesis." In: pp. 1–17 (cit. on pp. 34, 61, 62).

Krinner, Gerhard et al. (2013). "Long-term climate change: Projections, commitments and irreversibility." In: *Climate Change 2013 the Physical Science Basis: Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change* 9781107057, pp. 1029–1136 (cit. on p. 1).

Krishnamoorthi, Raghuraman (2018). "Quantizing deep convolutional networks for efficient inference: A whitepaper." In: (cit. on p. 75).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Neural Information Processing Systems* 25 (cit. on p. 3).

Kumar, Kundan, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville (2019). "MelGAN: Generative adversarial networks for conditional waveform synthesis." In: *Advances in Neural Information Processing Systems* 32.NeurIPS 2019 (cit. on pp. 31, 61, 62).

Lacoste, Alexandre, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres (2019). "Quantifying the Carbon Emissions of Machine Learning." In: (cit. on p. 45).

LeCun, Yann and Yoshua Bengio (1998). "Convolutional networks for images, speech, and time series." In: (cit. on p. 24).

Lecun, Y, L Bottou, Y Bengio, and P Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on p. 3).

Lo, Chen Chou, Szu Wei Fu, Wen Chin Huang, Xin Wang, Junichi Yamagishi, Yu Tsao, and Hsin Min Wang (2019). "MosNet: Deep learning-based objective

assessment for voice conversion." In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 2019-Septe, pp. 1541–1545 (cit. on p. 64).

Lohn, Andrew and Micah Musser (2022). *AI and Compute*. URL: https://blog.openai.com/ai-and-compute/ (cit. on pp. 3, 5).

McGinnis, Robert, Donella H. Meadows, Dennis L. Meadows, Jorgen Randers, and William W. Behren (1973). *The Limits to Growth: A Report for the Club of Rome's Project on the Predicament of Mankind*. Vol. 10. 2. New York : Universe Books, [1972], p. 295 (cit. on p. 1).

Mehri, Soroush, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio (2017). "Samplernn: An unconditional end-to-end neural audio generation model." In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–11 (cit. on pp. 27, 52).

Menghani, Gaurav (2021). "Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better." In: 1.1, pp. 1–43 (cit. on pp. 73, 74).

Ning, Yishuang, Sheng He, Zhiyong Wu, Chunxiao Xing, and Liang Jie Zhang (2019). "Review of deep learning based speech synthesis." In: *Applied Sciences (Switzerland)* 9.19, pp. 1–16 (cit. on p. 10).

Oord, Aaron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). "WaveNet: A Generative Model for Raw Audio." In: pp. 1–15 (cit. on p. 27).

Parzen, Emanuel (1962). "On Estimation of a Probability Density Function and Mode." In: *The Annals of Mathematical Statistics* 33.3, pp. 1065–1076 (cit. on p. 38).

Ping, Wei, Kainan Peng, and Jitong Chen (2019). "Clarinet: Parallel wave generation in end-to-end text-to-speech." In: *7th International Conference on Learning Representations, ICLR 2019* (cit. on p. 32).

Ping, Wei, Kainan Peng, Kexin Zhao, and Zhao Song (2020). "WaveFlow: A compact flow-based model for raw audio." In: *37th International Conference on Machine Learning, ICML 2020* PartF16814, pp. 7662–7672 (cit. on pp. 32, 58, 61, 63).

Prenger, Ryan, Rafael Valle, and Bryan Catanzaro (2019). "Waveglow: A Flow-based Generative Network for Speech Synthesis." In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2019-May, pp. 3617–3621 (cit. on pp. 32, 61, 63).

Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen (2022). "Hierarchical Text-Conditional Image Generation with CLIP Latents." In: Figure 3 (cit. on pp. 6, 7).

Ramírez, M A Martínez and J D Reiss (2019). "Modeling Nonlinear Audio Effects with End-to-end Deep Neural Networks." In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 171–175. ISBN: 2379-190X VO - (cit. on p. 10).

Renault, Lenny, Rémi Mignot, and Axel Roebel (2022). "Differentiable Piano Model for Midi-To-Audio Performance Synthesis." In: *Proceedings of the International Conference on Digital Audio Effects, DAFx* 3, pp. 232–239 (cit. on p. 11).

Richardson, Eitan and Yair Weiss (2018). "On GANs and GMMs." In: *Advances in Neural Information Processing Systems* 2018-Decem.Nips, pp. 5847–5858 (cit. on pp. 40, 41).

Rix, A. W., J. G. Beerends, M. P. Hollier, and A. P. Hekstra (2001). "Perceptual evaluation of speech quality (PESQ) - A new method for speech quality assessment of telephone networks and codecs." In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2, pp. 749–752 (cit. on p. 42).

Roads, Curtis (1978). "Automated granular synthesis of sounds." In: *Comput. Music J.* 2.2, pp. 61–62 (cit. on p. 10).

Rohde, Robert A (2005). *Global warming art project* (cit. on p. 2).

Rosenblatt, F. (1957). *The Perceptron - A Perceiving and Recognizing Automaton* (cit. on p. 3).

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature* 323.6088, pp. 533–536 (cit. on pp. 3, 23).

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen (2016). "Improved techniques for training GANs." In: *Advances in Neural Information Processing Systems*, pp. 2234–2242 (cit. on p. 39).

Schwartz, Roy, Jesse Dodge, Noah A. Smith, and Oren Etzioni (2020). "Green AI." In: *Communications of the ACM* 63.12, pp. 54–63 (cit. on pp. 4, 43).

Sevilla, Jaime, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos (2022). "Compute Trends Across Three Eras of Machine Learning." In: *Proceedings of the International Joint Conference on Neural Networks* 2022-July (cit. on p. 4).

Silver, David et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587, pp. 484–489 (cit. on p. 3).

Smith III, Julius O (1991). "Viewpoints on the history of digital synthesis." In: *Proceedings of the International Computer Music Conference*. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, p. 1. ISBN: 1026-1087 (cit. on p. 9).

Sohl-Dickstein, Jascha, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli (2015). "Deep unsupervised learning using nonequilibrium thermodynamics." In: *32nd International Conference on Machine Learning, ICML 2015* 3, pp. 2246–2255 (cit. on p. 32).

Soon, Willie, Sallie L. Baliunas, Arthur B. Robinson, and Zachary W. Robinson (1999). "Environmental effects of increased atmospheric carbon dioxide." In: *Climate Research* 13.2, pp. 149–164 (cit. on p. 2).

Steffen, Will, Wendy Broadgate, Lisa Deutsch, Owen Gaffney, and Cornelia Ludwig (2015). "The trajectory of the anthropocene: The great acceleration." In: *Anthropocene Review* 2.1, pp. 81–98 (cit. on p. 1).

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (2020a). "Energy and policy considerations for modern deep learning research." In: *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 1393–13696 (cit. on p. 4).

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (2020b). "Energy and policy considerations for modern deep learning research." In: *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence* 1, pp. 1393–13696 (cit. on pp. 44, 54).

Theis, Lucas, Aäron Van Den Oord, and Matthias Bethge (2016). "A note on the evaluation of generative models." In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–10 (cit. on pp. 37, 38).

Thompson, Neil C., Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso (2020). "The Computational Limits of Deep Learning." In: (cit. on p. 4).

Turing, Alan (1950). "Machinery and Intelligence." In: *Mind* 59.236, pp. 433–460 (cit. on p. 3).

Vincent, Emmanuel, Rémi Gribonval, and Cédric Févotte (2006). "Performance measurement in blind audio source separation." In: *IEEE Transactions on Audio, Speech and Language Processing* 14.4, pp. 1462–1469 (cit. on p. 41).

Wan, Diwen, Fumin Shen, Li Liu, Fan Zhu, Lei Huang, Mengyang Yu, Heng Tao Shen, and Ling Shao (2020). "Deep quantization generative networks." In: *Pattern Recognition* 105 (cit. on pp. 34, 75).

Zhou, Zhiming, Han Cai, Shu Rong, Yuxuan Song, Kan Ren, Weinan Zhang, Yong Yu, and Jun Wang (2018). "Activation Maximization Generative Adversarial Nets." In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–24 (cit. on p. 40).