


Multi-scale latent diffusion using hierarchical models for interpretable music generation

Alfred Pichard

Supervisors:
David Genova
Nils Demerlé
Philippe Esling

IRCAM - Public
from March 10th 2022 to August 27th 2024

Abstract

Recent advances in deep learning have introduced powerful new tools for audio synthesis, providing musicians with unprecedented means to explore and manipulate soundscapes. In particular, the rise of generative models has opened new creative perspectives and enabled novel forms of musical expression. Despite their success, one critical limitation of these models is their lack of expressive control, making it difficult for musicians to willingfully guide the generation.

To address this issue, researchers have explored various strategies to control generative models based on user-specified inputs. Amongst different approaches, diffusion models have emerged as a promising solution to synthesize high-quality data, while also offering robust conditioning mechanisms to guide the generative process. Such models have led to numerous new applications, notably in the audio domain, with the recent advent of text-to-music models such as Stable Audio 2 [1], where users can generate an entire song from a short textual description. While such applications illustrate the complex control possibilities offered by diffusion models, they yet fail to capture the full depth of musical intentions and have a very limited artistic reach. Indeed, the low expressiveness of small text prompts, as well as their static nature, comes in contradiction with the idea of precisely manipulating complex attributes over time, which is required to integrate neural networks within novel musical instruments. Consequently, multiple approaches have tried to condition generative models with more creatively adapted controls, such as melody or rhythm [2]. However, these approaches still fail to account for the inherent time and frequency hierarchies within musical compositions. Indeed, music is structured across multiple temporal scales, where short-term information, such as pitch or onsets, intertwine and lead to intricate patterns, such as melody or groove, which eventually give rise to complex long-term dependencies like structure or mood. Ignoring these hierarchical elements limits the expressive power of generative models for audio, as they do not fully capture the layered complexity of music. We believe that incorporating this hierarchical nature offers a more meaningful and sophisticated approach to control music generation.

This research aims to create a tool for music composition by enabling finer control over audio synthesis. We state that current architectures for images manipulation such as *Hierarchical Diffusion Auto-Encoders* (HDAE) [3] are suboptimal for audio applications. Consequently, we propose a novel hierarchical diffusion model that leverages a specific architecture and training scheme based on these considerations. This model incorporates multiple encoders to capture distinct levels of musical abstraction and employs a progressive training approach. Finally, we asses the effectiveness of our model on a custom evaluation framework, including various temporal ranges for tasks through feature manipulation and interpretability experiments.

Résumé

Les récents progrès en apprentissage profond ont introduit des outils novateurs et puissants pour la synthèse audio, donnant aux musicien.ne.s des moyens inédits pour explorer les paysages sonores. En particulier, l'essor des modèles génératifs a ouvert de nouvelles perspectives créatives ainsi que de nouvelles formes d'expression musicale. Malgré ces résultats, une limite concrète de ces modèles est leur manque de contrôle expressif, rendant difficile pour les artistes d'avoir la main sur le processus de génération.

Pour remédier à ce problème, diverses stratégies visant à contrôler les modèles génératifs à partir de descripteurs ont été explorées. Parmi plusieurs approches différentes, les modèles de diffusion se sont imposés comme une solution prometteuse pour synthétiser des données de haute qualité, tout en offrant des mécanismes de conditionnement robustes pour guider le processus génératif. Ces modèles ont donné lieu à de nombreuses applications, notamment dans le domaine de l'audio, avec le développement récent de modèles de texte-vers-musique, tel que Stable Audio 2 [1], qui rend possible la génération d'une musique entière à partir d'une courte description textuelle. Bien que ces applications illustrent les possibilités de contrôle complexes offertes par les modèles de diffusion, elles ne parviennent pas encore à saisir pleinement la profondeur des intentions musicales et restent très limitées pour des utilisations artistiques. En effet, la faible expressivité et le caractère statique des contrôles textuels sont en contradiction avec l'idée de manipuler précisément des attributs audios complexes dans le temps, ce qui est nécessaire pour intégrer les réseaux de neurones dans de nouveaux instruments de musique. Par conséquent, plusieurs approches ont tenté de conditionner les modèles génératifs avec des contrôles plus pertinents pour la création, tels que la mélodie ou le rythme [2]. Cependant, ces approches ne tiennent toujours pas compte des hiérarchies temporelles inhérentes à la musique. En effet, les signaux sonores musicaux sont structurés à plusieurs échelles temporelles, où des informations à court terme, telles que la hauteur ou les attaques, se combinent pour former des motifs complexes, comme la mélodie ou le rythme, qui, à leur tour, donnent naissance à des motifs se manifestant à l'échelle du morceau, tels que la structure. Nous pensons que prendre en compte cette hiérarchie permettrait de développer une approche plus sophistiquée pour contrôler la génération musicale.

Ce stage vise à créer un outil de composition musicale en permettant un contrôle plus fin de la synthèse audio. Nous pensons que les architectures actuelles pour la manipulation d'images, telles que les *Auto-Encodeurs de Diffusion Hiérarchiques* (AEDH) [3], ne sont pas adaptées aux applications audio. Par conséquent, nous proposons un nouveau modèle de diffusion hiérarchique, reposant sur une architecture et un schéma d'entraînement prenant en compte ces considérations. Ce modèle intègre notamment plusieurs encodeurs pour capturer différents niveaux d'abstraction musicale, et utilise une approche d'entraînement progressive. Enfin, nous évaluons l'efficacité de notre modèle à travers une stratégie d'évaluation adaptée à notre objectif, incluant plusieurs tâches d'extraction d'information à différentes échelles temporelles, ainsi que par l'étude de l'effet de la manipulation de représentations latentes sur l'audio généré.

Contents

1 Introduction

1.1 Audio synthesis for music

Over the last century, artistic creation and technological innovations have been deeply and reciprocally linked. Music, as a deeply expressive and widely embraced form of art, has closely followed this pattern and has witnessed profound transformations with the advent of *audio synthesis*. This modern revolution has been led through experimental work on hardware components by pioneers like Iannis Xenakis, who explored *granular synthesis*, and foundational theoretical writings such as Max Matthews' influential article, *The Digital Computer as a Musical Instrument* [4]. These new approaches to sound production naturally benefited from the constant increase in computers' capacities, and milestones in this domain have regularly been achieved by the scientific community, leveraging knowledge crossing the fields of computer science, signal processing and music theory. For example, the development of *Max/MSP* by Miller Puckette and David Zicarelli at IRCAM between 1993 and 1994 [5] has expanded the possibilities of computer music by creating an interactive visual programming language based on the signal processing principles established in the preceding years. These innovations have pushed creative boundaries and empowered composers to explore new genres, continuously reshaping the definition of music up to the present day.

In recent years, sound design became a vital part of music creation through the use of modern tools that can be integrated in *Digital Audio Workstations* (DAWs). Hence, innovative technical approaches have increasingly taken part in the creative process, profoundly influencing the music we experience today and will experience in the future.

1.2 Deep neural audio synthesis

More recently, *machine learning* has rapidly advanced and drove numerous innovations across various fields, including signal processing and audio analysis. These advances were fueled by the exponential growth in data availability and computational power. Specifically, the development of *deep learning* approaches introduced powerful *generative models*, able to synthesize new samples from a set of selected examples, while providing novel types of interaction for audio generation [6] [7].

While historical synthesis methods often rely on the *mimesis* of specific phenomena, such as modeling a guitar string coupled with its sounding board, deep learning models learn audio properties directly from the observation of a given set of examples. Indeed, generative models approximate the underlying characteristics of a dataset and can be used to create new examples by sampling from the learned distribution. This shift introduces novel interactions capabilities that can be exploited as these models are not produced by the same set of rules. Various generative paradigms such as *Generative Adversarial Networks* (GAN) [8] or *Variational Auto-Encoders* (VAE) [9] have been proposed, and are now capable of generating realistic samples. However, first attempts with these models faced issues with quality and inference time, due to the high dimensionality of audio data. Hence, generative models in

audio were originally designed to process simpler representations, such as spectrograms [10]. However, by leveraging a combination between a *multi-band decomposition* of audio signals, *variational auto-encoding* and *adversarial training*, the recent *RAVE* model [7] achieved real-time generation of high-quality audio samples on generic laptops.

Follow-up research lead to the development of a specific class of models capable of reducing the original audio dimensionality by learning mappings between the waveform and a deterministic *latent* representation. Such models provide a very useful representation tool for audio compression, introducing latent models for generation such as *Encodec* [11]. More recently, the *diffusion* framework [12] has further improved representation learning [13] and resulted in even faster and higher quality compression models such as *Music2latent* [14].

1.3 The issue of control in deep generative models

While deep learning methods for audio generation have reached satisfying results in quality and efficiency, they still lack meaningful and expressive controls, which is a critical component for the development of creative tools for artists. Indeed, the interaction between musicians and their instruments is a key factor in composition and performance, especially in electronic and experimental music. For instance, the design of the first sequencers, notably the *TR series* by Roland, played a significant role in defining the core elements of the early house and techno sound in the mid-80s. However, defining such relevant controls for deep learning models is a challenging task, and certain aspects of music composition are difficult to capture in model design.

This challenge to manipulate the generative process has already been explored via the use of *conditioning* [15]. This approach aims to guide the generation with additional categorical data and tags. Using this extraneous information to categorize the training data for deep models is called *supervised learning*. Specifically for music, the *DDSP* [16] model proposed to use *audio descriptors* alongside audio representations to enrich the latent information. The advent of *diffusion models*, which offered qualitative results over complex data generation, pushed these methods a step forward thanks to the accessible control they offer through *classifier-free guidance* [17]. Moreover, using diffusion networks as *priors* for efficient and qualitative compression models allows the generative process to be centered on a more elementary task. Hence, *latent diffusion* approaches have become increasingly popular for conditional generation. This has facilitated the use of more abstract conditioning representations such as genre or style in various design frameworks [2]. Finally, the recent development of powerful language models for audio [18] and *diffusion auto-encoders* (DAE) [13] for representation learning introduced new perspectives for deep neural audio synthesis. Indeed, their combination reached impressive performances in text-conditioned audio generation [19, 20]. Yet, the musical and creative impacts of theses approaches remain to be questioned, as some abstract musical concepts such as timbre or colour do not easily fit with language descriptors. Moreover, supervised approaches are limited by their dependence on explicit tags and constrained by the need for annotated datasets.

In order to define more flexible tools for deep audio synthesis, *unsupervised* approaches should be privileged, meaning that training a model from custom datasets should only require non-annotated data. These models allow for more exploratory learning by revealing underlying patterns or structures, potentially enabling high-level controls that may not be easily defined with common descriptors or labels. The original formulation of VAEs [9] was designed in this spirit, and the latent space they induce can be explored for control. However, features in such spaces often exhibit dependencies by being correlated to each others. *Disentanglement* aims to reduce these dependencies by isolating the different factors of variation in the learned latent representation. The goal is for each latent variable to correspond to a distinct attribute. This results in greater control and flexibility when manipulating generated outputs, as observed in β-VAEs [21]. With diffusion models, DAEs [13] introduced another type of latent space representation with a compression of the high-level features into *semantic* vectors, and also benefited from a disentanglement extension with Hierarchical Diffusion Auto-Encoders (HDAE) [3].

In audio, unsupervised approaches have already achieved qualitative results [7] [22]. However, unlike images where features are easily separable, the complexity of audio data introduces analysis challenges and often leads to entanglement. This makes latent spaces harder to interpret or control, warranting the development of meaningful control mechanisms as a crucial area for scientific research.

1.4 The hierarchical nature of music

One of the major challenges in understanding audio signals lies in their intricate structure, which spans multiple temporal scales. While some short-time information such as transients and onsets are contained solely in lower-level temporal scales (fig. 1b), notes length, rhythm or even higher-level data like melody and structure arrangement can be observed at larger time scales (fig. 1a and fig. 1c).

(a) 30 milliseconds (b) 300 milliseconds (c) 3 seconds

Figure 1: Different temporal scales for the same musical audio sample

Hence, an audio signal is inherently defined as a hierarchical source of information. It is composed of multiple layers, each one with its own level of complexity. At the lowest level, audio signals are composed of fundamental elements that define core characteristics of sound, such as volume and spectral placement. These elements combine to form more complex features, such as harmonics, which create the timbre of a sound, and envelopes, which describe how the amplitude of the signal changes over time (e.g., attack, decay,

sustain and release in musical notes). As we move up the hierarchy, these features combine to create *patterns*, such as rhythms, and contribute to the recognition of musical melodies. At the highest level, these patterns form entire *musical phrases* or soundscapes. This level of the hierarchy carries the most complex and meaningful information, like the emotional tone or genre of a music. Beyond the signal itself, there is a layer of contextual and semantic information that the listener perceives, such as the cultural significance of a musical piece.

Each layer builds upon the lower layers, creating a multi-tiered structure where higher levels provide context and meaning derived from the integration of simpler, more fundamental components. This hierarchical nature allows for complex and nuanced interpretation and analysis of audio signals. Yet, it is important to consider all of these scales simultaneously to correctly model audio, and eventually control its generation, as the low-level features of a musical sample are inherently linked with the context they emerge from.

1.5 Control through hierarchical learning

As we just discussed, the temporal structure of an audio signal contains rich information at different scales. Hence, we think that an effective control framework for deep generative models must account for this hierarchy to enable precise and expressive manipulation of the synthesis process. Hence, the objective of this research is to design a deep learning model that leverages hierarchical separation of musical features as conditioning inputs. To maintain adaptability and avoid the limitations of labeled data, the model should employ self-supervised learning for feature extraction directly from the audio signals, allowing it to be trained on diverse, unannotated datasets while preserving generalization capabilities.

Drawing inspiration from Diffusion Auto-Encoders (DAE) [13], we propose a model incorporating multiple encoders in order to capture information at various levels of abstraction, and use them as conditioning information for diffusion networks trained to reconstruct the original data. To ensure disentangled representations, distinct features are assigned to different encoders, with a multi-stage training strategy inspired from progressive GANs [23] employed to refine this separation. The effectiveness of our model is tested through feature manipulation tasks and interpretability experiments, and we benchmark its performances against the Hierarchical Diffusion Auto-Encoder (HDAE) [3] framework, which we adapted to audio processing. We expect our architecture to provide a manipulation tool for audio samples, able to extract different hierarchical features and recombine them into a novel snippet. We think this multi-layer network is particularly relevant for music composition and sound design, and will allow finer sampling of examples from different parts of the hierarchical latent spaces.

The remainder of this report is organized as follows. First, we provide a brief presentation of core machine learning principles and deep generative frameworks with their application to audio. Then, we discuss the theory behind diffusion models, specifying their different formulations, and the control capabilities they provide. After a focus on hierarchical

generative models, we present our contribution : a multi-scale spectral hierarchical diffusion auto-encoder. We detail our model design and characteristics and, then, further explain the choices made for this specific architecture and training process. Finally, we present the experiments designed to evaluate our model, including baseline comparisons and feature analysis. We conclude with a discussion on the contributions and limitations of our work, opening with some ideas to continue this research in the future.

2 State of the art

In this section, we outline the foundational principles of machine learning and neural networks that underpins our research. Additionally, we aim to provide some insights into control mechanisms for deep neural data generation, which are essential to fully understand the methodology presented in our work. We begin by recalling the general machine learning techniques and provide an overview of deep neural networks, with a specific emphasis on audio processing. Following this, we delve into diffusion models and explore their various formulations. Lastly, we examine current techniques for the control of diffusion models and their applications in the generative audio domain.

2.1 Machine learning

2.1.1 Machine learning foundations

The core principle of machine learning revolves around building models that can estimate data distributions. These models rely on identifying patterns in datasets and aim to generalize them to unseen cases, approximating the relation between the two spaces $\mathcal{X}$ (data) and $\mathcal{Y}$ (predictions) that we are interested in. They are typically represented as parametric functions that map inputs $\mathbf{x} \in \mathcal{X}$ to outputs $\hat{\mathbf{y}} \in \mathcal{Y}$, such that

$$f_\theta : \mathbf{x} \to \hat{\mathbf{y}} \tag{1}$$

The process of learning involves adjusting the model parameters θ to minimize the difference between its predictions and the actual outcomes corresponding to the training pairs $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, which is quantified by a loss function $\mathcal{L} : (\mathbf{x}, \mathbf{y}) \to \mathcal{L}(f_\theta(\mathbf{x}) = \hat{\mathbf{y}}, \mathbf{y})$. The definition of a loss function is dependent on the specific task being performed, but the most commonly used loss is the Mean-Squared-Error (MSE), defined as

$$\mathcal{L}_{MSE}(\mathbf{x}, \mathbf{y}) = || f_\theta(\mathbf{x}) - \mathbf{y}||^2. \tag{2}$$

The nature of the training data defines the two main approaches in machine learning : *supervised* learning, where the model is trained on labeled data, and *unsupervised* learning, where instead of being given explicit instructions on what to predict, the model's objective is to discover patterns or structures within the data. Common applications for supervised tasks are *regression* and *classification*, whereas unsupervised models are better suited for *clustering* and *representation learning*. In both cases, in order to find the set of parameters θ that minimizes the loss function $\mathcal{L}$, machine learning approaches use iterative optimization techniques. The most common method rely on *gradient descent*, where the model updates its parameters in the direction that reduces the loss function at each iteration

$$\theta_{n+1} = \theta_n - \eta \Delta_\theta L_{\mathcal{X},\mathcal{Y}}(\theta_n) \tag{3}$$

Computing the gradient can generally be done efficiently with the chain rule

$$\frac{\partial \mathbf{y}_N}{\partial \theta}(\theta) = \frac{\partial \mathbf{y}_N}{\partial \mathbf{y}_{N-1}}(\mathbf{y}_{N-1}(\theta)) \frac{\partial \mathbf{y}_{N-1}}{\partial \theta}(\theta) \tag{4}$$

The coefficient η is an hyperparameter, called *learning rate*, which controls the magnitude of the model parameters updates at each iteration. If the learning rate is too high, training might be prone to loss instability or divergence. On the other hand, if the learning rate is too low, the training will be slow, potentially causing the loss to get stuck in a local minimum and fail to reach the optimum. Hence, choosing an appropriate learning rate is crucial for efficient and effective training. Specific methods like learning rate schedules, where the rate is gradually reduced during training, or adaptive learning rate methods like Adam [24] can adjust the learning rate dynamically based on the gradients.

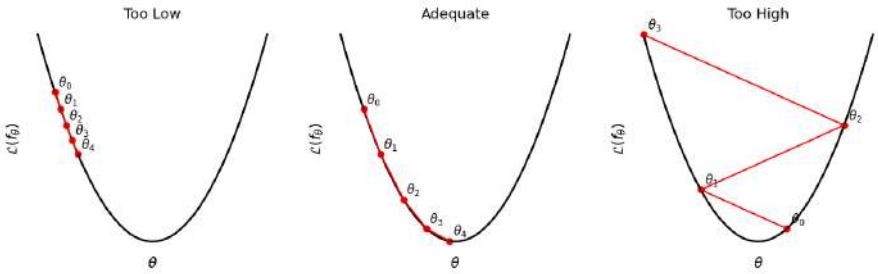

Figure 2: Different learning rates and their effect on gradient descent

2.1.2 Neural networks

Machine learning models f_θ must be differentiable with respect to their parameters θ, as their gradient Δ_θ is used for optimization. For instance, polynomial models can be used for simple scalar regressions. Yet, while any differentiable parametric function can be used regardless the task, some architectures are particularly suited for complex problems thanks to their structural biases. While *neural networks* do not encompass the full range of parametric functions available in machine learning, they provide flexible function and serve as the fundamental building blocks upon which our subsequent research is based.

Multi-Layer Perceptrons

Neural networks (NN) were originally designed as a mathematical model of a brain cells. A network is a structure composed of a set of interconnected instances called neurons, organized in layers. Each neuron receives inputs from the preceding layer, and processes them with an operation defined by weights $\mathbf{w} = (w_1, ..., w_n)$, a bias b and a non-linear activation function σ, and passes its output to the next layer of neurons. The transformation applied to an input vector $\mathbf{x} = x_1, ..., x_n \in \mathbb{R}^{\mathcal{D}_\mathbf{x}}$ by a single neuron can be written as

$$y = \sigma\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{5}$$

Commonly used activation functions are the *sigmoid* or *tanh*, defined as

$$\begin{aligned} \sigma_{Sigmoid}(x) &= 1/(1+e^{-x}) \\ \sigma_{Tanh}(x) &= tanh(x) \end{aligned} \tag{6}$$

A *dense*, *fully-connected* or *linear* layer is usually a function of another previous layer output. It is defined by its weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$, and maps an input $\mathbf{y}_N$ to an output $\mathbf{y}_{N+1}$ with

$$\mathbf{y}_{N+1} = \sigma(\mathbf{W}\mathbf{y}_N + \mathbf{b}) \tag{7}$$

Stacking N layers produces a network of depth N, called *Multi-Layer Perceptron* (MLP), as depicted in fig. 3. They can be written as the following composition of functions

$$\mathbf{y} = f_\theta(\mathbf{x}) = \mathbf{y}_N \circ \mathbf{y}_{N-1} \circ ... \circ \mathbf{y}_2 \circ \mathbf{y}_1(\mathbf{x}) \tag{8}$$

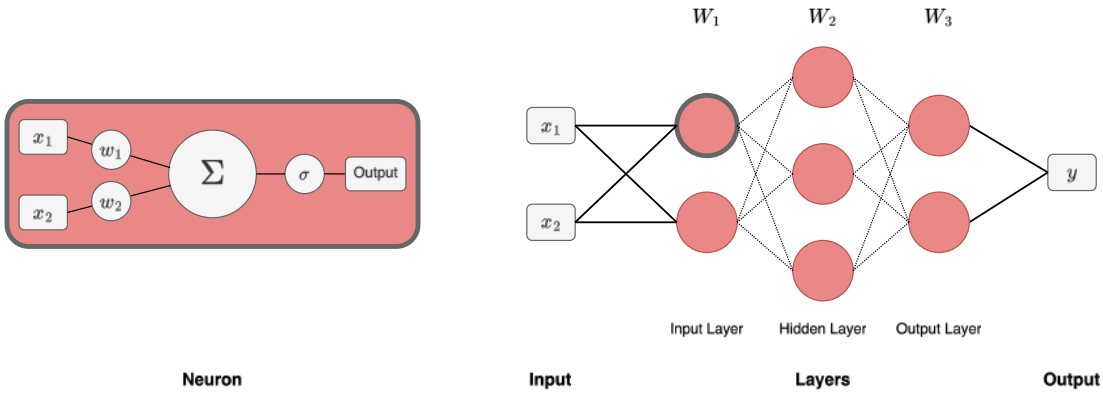

Figure 3: An input *neuron* and a 3-layers *MLP* network

Even with a small number of layers, these structures can represent highly complex functions while being composed of many simple singular units. Hence, by considering $\theta = \{\mathbf{W}, \mathbf{b}\}$, the neural network defines a differentiable parametric function $f_\theta : \mathbf{x} \to \mathbf{y}$ that can be optimized to model complex relationships between different spaces. Thanks to the chain rule, the gradient of each layer with respect to its parameters can be expressed based on the gradient of the next layer. When used in an optimization problem, this iterative technique is called *backpropagation*. This greatly reduces the amount of computation required, which is a critical aspect of machine learning as it allows for a relatively fast training process.

Convolutional Neural Networks

While the MLP formulation is well-suited for simple machine learning problems, it may be less efficient for extracting features from high-dimensional input data. Indeed, the large number of neurons required for this task often leads to redundancy, and their inherently

unidimensional nature can result in the loss of critical spatial or temporal information. Convolutional Neural Networks (CNNs) address these limitations by replacing affine transforms with learnable convolutional filters, known as *kernels*. These kernels are convolved with the input data to produce *feature maps*, thereby preserving spatial and/or temporal structures. Hence, these models are particularly valuable for signal processing, especially in computer vision tasks and audio treatment.

Each layer of a CNN is composed of N kernels that are convolved across the input. Hence, the i^{th} layer outputs N feature maps $(o_n^i)_{n\in[1;N]}$ corresponding to the sum of the convolutions $(k_n)_{n\in[1;N]}$ with each of the M input channels $(x_m)_{m\in[1,M]}$. An additional bias b_n^i can also be used. Formally, we have

$$o_n^i = \sum_{m=1}^{M} k_n^i * x_m + b_n^i \tag{9}$$

Again, a graphical representation is provided for greater clarity (fig. 4). Similar to MLPs, a non-linear activation function is applied after the summation operation. Additionally, *pooling* may be employed to reduce the number of parameters in the model.

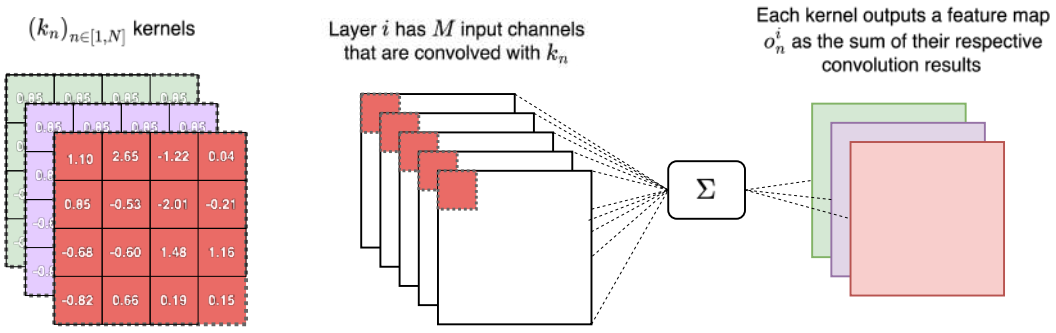

Figure 4: A CNN Layer operation with 3 kernels and a 5-channel input data

2.1.3 Deep learning

Although shallow neural networks already achieve remarkable success in modeling complex processes, increasing their depth usually comes with significant limitations. Notably, as gradients are back-propagated through numerous layers and activation functions, they often greatly diminish in magnitude, becoming too small to be effectively interpreted. This can hinder the model's ability to converge accurately, ultimately compromising its performances. This issue is called the *vanishing gradients*, and the introduction of softer and non-saturating activation functions such as *Rectified Linear Unit* (*ReLU*) [25] or *Sigmoid Linear Unit* (*SiLU*) [26] successfully started to resolve this particular constraint. Furthermore, numerous strategies were introduced to address other training issues, leading to the development of

the *deep learning* field.

Batch normalization is a technique that stabilizes and accelerates the training of deep neural networks by normalizing the inputs before each layer. It standardizes the inputs across successive mini-batches to a unit Gaussian distribution by adjusting their mean and variance, which reduces internal covariate shift. This process enables faster convergence, allows for higher learning rates and reduces sensitivity to weight initialization, making it essential for training deep models effectively [27]. This technique also provides a regularization effect, which is crucial in enhancing the generalization ability of deep learning models by preventing *overfitting*. In this spirit, *Dropout* is one of the most common regularization techniques. It consists in randomly masking neurons or dropping output values during each training iteration. By doing so, the model is forced to learn robust features that are not overly reliant on any single weight, thereby improving generalization [28].

2.1.4 Probabilistic formulation

In the context of our work, we specifically concentrate on generative models, the principles of which will be detailed in the following sections (section 2.2 & section 2.3). While some generative approaches are build on traditional machine learning foundations, most are grounded in probabilistic formulations. Therefore, before delving into their core principles it is essential to first define the probabilistic framework that underpins these models.

While we are still considering a dataset of features and predictions $\mathcal{X}, \mathcal{Y}$, inputs $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$ are now defined as random variables drawn from probability distributions, so that $\mathbf{x} \sim p(\mathbf{x})$ and $\mathbf{y} \sim p(\mathbf{y}|\,\mathbf{x})$. Consequently, we now aim to model the joint probability distribution $p(\mathbf{x},\, \mathbf{y})$, and our model $p_\theta(\mathbf{x},\, \mathbf{y})$ is then derived from a family of parametric probabilistic functions that characterize the relationship between $\mathbf{x}$ and $\mathbf{y}$. The process of predicting an output $\hat{\mathbf{y}}$ from a given sample $\mathbf{x}$ is called *inference*, and is done by computing the posterior probability $p_\theta(\hat{\mathbf{y}}|\,\mathbf{x})$. For differentiable models, optimization is still usually done with gradient-based techniques. However, the loss is different from the deterministic definition. Indeed, we usually rely on maximum likelihood estimation, which is defined for a dataset composed of independent and identically distributed values $(\mathbf{x}_i,\, \mathbf{y}_i)$ of size n as

$$l(\theta) = \prod_{i=1}^{n} p(\mathbf{y}_i,\, \mathbf{x}_i \,|\, \theta) \tag{10}$$

To facilitate computations, the negative log-likelihood is often used :

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} log\, p(\mathbf{y}_i,\, \mathbf{x}_i \,|\, \theta) \tag{11}$$

Minimizing the negative log-likelihood is equivalent to maximizing the likelihood, but it often simplifies mathematical expressions and aligns with minimization algorithms.

2.2 Established generative models

The advent of deep learning have not only broadened the application scope of generative models, but also enhanced their potential and capabilities, enabling breakthroughs across various scientific fields. In this section, we introduce the core concepts behind deep generative models and their historical applications. As our work mainly relies on diffusion models, which have been defined recently, they are presented in a dedicated section (see 2.3).

2.2.1 Deep generative models

The majority of generative models are grounded in the probabilistic framework outlined previously. In this context, each individual example $\mathbf{x} \in \mathcal{X}$ is treated as a random variable drawn from the underlying probability distribution $p(\mathbf{x})$. A probabilistic generative model seeks to approximate this distribution through a parametric function, with the objective of achieving $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$. In *latent* variables models, the variations in $\mathbf{x}$ are represented within a lower-dimensional latent space, where latent variables $\mathbf{z}$ encapsulate a high-level abstraction of the original data. This framework allows to assume that the generation of $\mathbf{x}$ is conditioned on $\mathbf{z}$, defining a new formulation of our model $p(\mathbf{x}, \mathbf{z})$. Hence, the marginalized distribution that we aim to model can also be written as

$$p(\mathbf{x}) = \int p(\mathbf{x} \,|\, \mathbf{z}) p(\mathbf{z}) d\mathbf{z} \tag{12}$$

However, this integral is generally impossible to solve analytically. Sampling methods can be used to estimate it, but as the dimension of $\mathbf{x}$ is generally large, it introduces great computational costs. Hence, researchers have proposed a variety of techniques to approximate $p(\mathbf{x})$ more efficiently, with the two most popular approaches being Variational Auto-Encoders (VAEs) [9], and Generative Adversarial Networks (GANs) [8].

2.2.2 Variational Auto-Encoders

VAEs [9] extend traditional *auto-encoders* [29] by incorporating the concept of *variational inference* [30]. Standard auto-encoders are neural network composed of an *encoder* that compresses input data into a lower-dimensional latent representation, and a *decoder* that reconstructs this latent representation back to the original data space. However, they are deterministic and do not inherently support the generation of new data samples. In contrast, VAEs use a probabilistic formulation that enables the generation of new data by sampling from a prior distribution. This approach allows VAEs to learn a continuous, probabilistic latent space, facilitating both the accurate reconstruction of input data and the generation of novel, similar data. This formulation places VAEs in a category called *likelihood-based models*, in which we can also find auto-regressive models [31] and normalizing flows [32].

VAE models rely on *variational inference*, which is a method to approximate complex probability distributions. It is especially useful when dealing with intractable integrals, like the *marginalized distribution* $p(\mathbf{x})$ (Equation 12). Variational inference approximates the true posterior $p(\mathbf{z} \,|\, \mathbf{x})$ with a simpler, tractable distribution $q_\phi(\mathbf{z} \,|\, \mathbf{x}) \in \mathcal{Q}$, where ϕ

are its parameters. The objective hence becomes to find the best possible q_ϕ distribution that is close to the true posterior, measured with a *Kullback-Leibler* (KL) *divergence*. However, since directly minimizing the KL divergence between $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ and $p(\mathbf{z}\,|\,\mathbf{x})$ is difficult, variational inference instead maximizes a related quantity known as the *Evidence Lower Bound* (ELBO). This is defined as the following objective

$$\mathcal{L}(\phi, \theta) = \underbrace{\mathbb{E}_{\mathbf{z}}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]}_{\text{reconstruction}} - \beta \underbrace{\mathcal{D}_{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})]}_{\text{regularisation (ELBO)}} \tag{13}$$

The first term can be interpreted as a reconstruction error, as it measures the likelihood of the observed data $\mathbf{x}$ being generated from the latent variables $\mathbf{z}$, while the second term assesses the divergence between the approximate posterior distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}) \in \mathcal{Q}$ and the prior distribution $p(\mathbf{z})$. A largely adopted choice for the family of distributions $\mathcal{Q}$ is to consider latent variables as independent and parameterized as *Gaussian distributions*. The parameter β was introduced alongside the regularization term to better control the trade-off between the two terms of the equation, and improves latent space disentanglement. This modification resulted in a new formulation known as the β-VAE [21].

VAEs use neural networks to parametrize the distributions p_θ and q_ϕ, and optimization is done using gradient descent on the joint loss (Eq. 13). The encoder learns to approximate the posterior $q_\phi(\mathbf{z}\,|\,\mathbf{x})$, while the decoder takes a latent $\mathbf{z}$ drawn from this distribution and learns to reconstruct the sample $\overline{\mathbf{x}}$, approximating $p_\theta(\mathbf{z}\,|\,\mathbf{x})$ (See fig. 5).

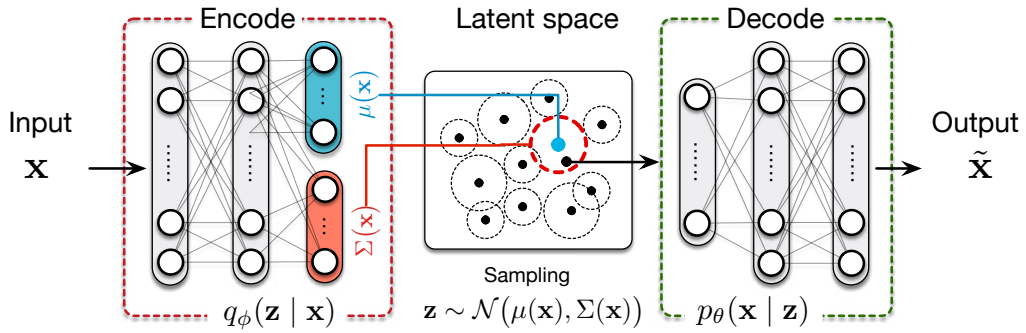

Figure 5: Variational Auto-Encoder architecture

As the sampling operation to obtain the latent variables $\mathbf{z}$ is not differentiable, VAEs use a technique called the *reparameterization trick* to back-propagate through the stochastic sampling process. Instead of sampling directly from $\mathcal{N}(\mu, \sigma^2)$, the model samples from a standard normal distribution $\mathcal{N}(0, 1)$ and shifts and scales it using the learned μ and σ: $\mathbf{z} = \mu + \sigma \odot \epsilon$ where ϵ is sampled from $\mathcal{N}(0, 1)$.

2.2.3 Generative Adversarial Networks

Generative adversarial networks (GANs) [8] are another family of deep generative models, part of the wider class of *implicit generative models* [33]. They are particularly well-known for their ability to generate realistic images, but they were also used in various other tasks including audio processing. A GAN consists of two neural networks

- The *generator* (G) generates data given a latent $\mathbf{z}$ drawn from a normal distribution. It is trained to output samples that closely align with the true data distribution $p(\mathbf{x})$.
- The *discriminator* (D) is trained to evaluate whether a given input data sample is *real* (from the training dataset) or *fake* (produced by the generator).

These two networks are trained simultaneously and compete against each other during the training process. The loss functions for the generator and discriminator are set up so that the generator tries to minimize the discriminator ability to distinguish between real and fake data, while the discriminator tries to maximize its ability to do so

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{14}$$

Training GANs presents several challenges that may affect the quality of the generated data. One major issue is *instability*, where the balance between the generator and discriminator is difficult to maintain. It often results in one network overpowering the other which leads to suboptimal outcomes. Additionally, GANs are prone to *mode collapse*, where the generator produces a limited variety of samples, significantly reducing the diversity of the output. A last critical challenge is the sensitivity of GANs to hyperparameters, such as learning rate and architectures. These parameters require meticulous tuning to achieve desirable performance, and often make training GANs a complex and delicate process. However, reformulations of the training objective and advancements in architecture design have made the use of these models manageable and effective, for instance with *WGAN* [34].

Progressive growing for GANs is another technique introduced to improve the training stability and quality of generated data [23]. It relies on the idea that directly generating large, high-resolution images from scratch can lead to poor results, such as artifacts or a lack of details. Progressive growing addresses this by gradually increasing the resolution of the images as the training progresses, which stabilizes the process and improves the quality of the generated data (See fig. 6).

2.2.4 Applications to audio

Deep generative techniques have been applied to audio in different manners. Although some models were originally developed to directly generate raw waveform [16] [6], this task is generally very computationally expensive. Indeed high-quality audio waveforms are usually sampled at 44.1kHz, meaning that one second of sound is represented by a vector of 44100 dimensions. Hence, the majority of audio applications to generative models rely on

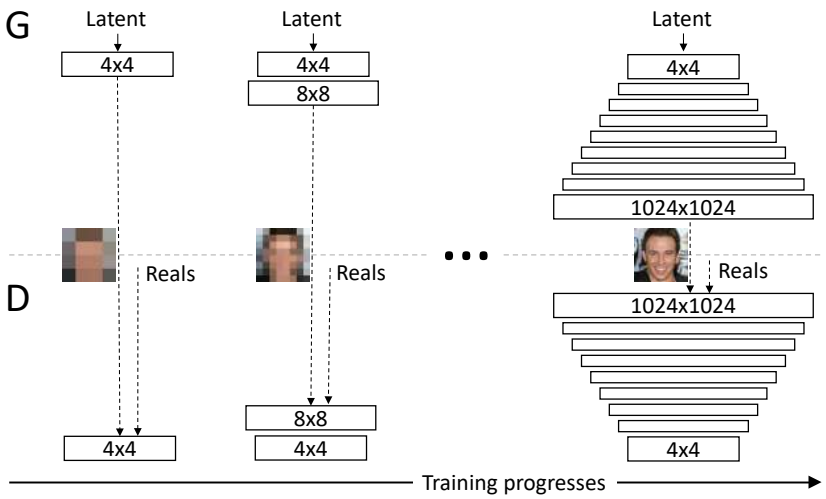

Figure 6: Progressive growing, taken from [23]. Both the generator (G) and discriminator (D) start with a low spatial resolution. As training progresses, layers are added to G and D, gradually increasing the spatial resolution of the generated images.

compression frameworks, that create a mapping between this high-dimensional original data and a lighter latent representation. It introduces a clear distinction between the generative and sampling process and with it the notion of prior model.

Oppositely, RAVE [7] achieves fast and efficient computation for generating raw audio data. It is able to generate high-quality waveforms at 48kHz on a regular laptop CPU. Moreover, this model is a powerful creative tool as it provides fast inference and real-time generation. To do so, it relies on combining different classes of generative models presented earlier, notably GANs and VAEs. It is designed to learn a probabilistic latent space, allowing to generate new audio samples by sampling directly from this space. Hence, this architecture is also compatible with a prior model for enhanced control over the generation. This model relies on the Pseudo-Quadrature-Mirror Filter (PQMF) decomposition of the signal and on a two-stages training process (See fig. 7)

- In the first stage, the VAE is optimized to encode and decode audio by minimizing the spectral distance across PQMF channels. This process constructs the latent space, which serves as the basis for subsequent generation. However, the outputs generated at this stage tend to be of suboptimal quality, often missing critical details.
- During the second stage, the focus shifts to fine-tuning the decoder using an adversarial criterion, compelling the model to produce more realistic samples. At this point, the encoder is frozen, while the decoder is trained in conjunction with a discriminator that allows to separate between real and generated samples.

This model forms the basis of our work. Indeed, this generative framework provides direct

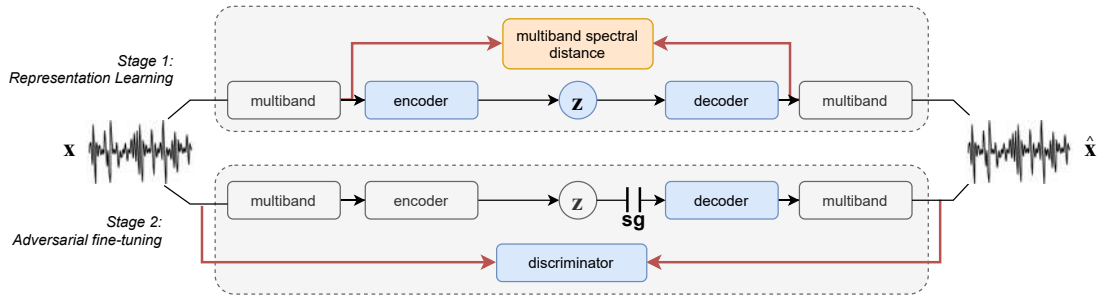

Figure 7: The RAVE training process, taken from [7]. The components highlighted in blue are those that can be optimized during their respective stages of training.

sampling from the latent space, while enhanced control can be achieved using a prior model.

2.3 Diffusion models

Diffusion models [35, 36] have recently emerged as a powerful family of generative models, demonstrating impressive performance in generating high-quality data across various domains, including image [37] and audio synthesis [38]. Inspired by the thermodynamic diffusion process [39], these models iteratively transform noise into complex data distributions through a gradual denoising operation (See fig. 8). Unlike traditional generative models such as VAEs and GANs, diffusion models are characterized by their stability in training and their ability to produce diverse and detailed outputs with expressive control. This has positioned them as a promising alternative to existing generative frameworks, particularly in tasks requiring high fidelity and robustness. Recently, diffusion-based techniques were extended to not only generate data but also provide meaningful latent representations [13], paving the way for tasks like reconstruction, compression, and data manipulation.

Figure 8: The denoising diffusion *reverse* process, mapping Gaussian noise into a structured target data distribution

2.3.1 Original formulation

Early steps on the denoising principle laid the groundwork for modern diffusion models, and has been explored through various formulations before being unified. The core idea of

the original article by Sohl-Dickstein et al. [12] is to model complex distributions through a diffusion forward and reverse process, where data is gradually transformed into noise and then reversed back to its original form (See fig. 9). It is governed by a Markovian sequence of probabilistic transforms allowing it to generate new samples once it is trained.

Diffusion forward process

By gradually converting a distribution $\mathbf{x}_0 \sim q_0$ into Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$, we define a series of latent variables $\mathbf{x}_1, \cdots, \mathbf{x}_T$ as a sequence of noisy samples. This operation is called the diffusion forward process $q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$, and applying it as a fixed Markov chain to the data according to a variance schedule $(\beta_1, \cdots, \beta_T) \in [0, 1]^T$ yields

$$q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t,\, \sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\, \beta_t \mathbf{I}) \tag{15}$$

$$\implies q(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \tag{16}$$

Reparametrizing 15 with $\alpha_t = 1 - \beta_t$ and $\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$ allows us to sample $\mathbf{x}_t$ at any arbitrary step t

$$q(\mathbf{x}_t \,|\, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t,\, \sqrt{\overline{\alpha_t}}\mathbf{x}_0,\, (1-\overline{\alpha_t})I) \tag{17}$$

$$\implies \mathbf{x}_t = \sqrt{\overline{\alpha_t}}\mathbf{x}_0 + \sqrt{1-\overline{\alpha_t}}\epsilon_t \quad \text{with} \quad \epsilon_t \sim \mathcal{N}(0, \mathbf{I}) \tag{18}$$

Diffusion reverse process

Similarly to the forward process, sampling from $q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t)$ for any t would allow to recreate any original $\mathbf{x}_0$ from a Gaussian noise input $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. However, estimating $q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t)$ it usually not possible when not conditioned with $\mathbf{x}_0$, and these probability distributions are hence approximated by a model p_θ, defining the diffusion reverse process, yielding

$$p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1},\, \mu_\theta(\mathbf{x}_t,\, t),\, \mathbf{\Sigma}_\theta(\mathbf{x}_t,\, t)) \tag{19}$$

$$\implies p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_0) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t) \tag{20}$$

Therefore, training our model is equivalent to learn the two functions $\mu_\theta(\mathbf{x}_t)$ and $\mathbf{\Sigma}_\theta(\mathbf{x}_t)$. Moreover, if $q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t)$ is usually intractable, it becomes manageable when it is conditioned with $\mathbf{x}_0$. Using Bayes' rule we obtain

$$q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t,\, \mathbf{x}_0) = q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1},\, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_0)}{q(\mathbf{x}_t \,|\, \mathbf{x}_0)}, \tag{21}$$

which leads to

$$q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t,\, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t,\, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}), \tag{22}$$

with

$$\begin{cases} \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \\ \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \end{cases} \tag{23}$$

The results of 22 are essential to optimize our model p_θ, as we can observe that our setup is similar to VAEs. Indeed, the complete reverse probability can be written as

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}, \tag{24}$$

which is also intractable. Thus, the variational lower bound can be used to optimize the negative log-likelihood, providing a tractable means of approximating the true data distribution. Using the Jensen's inequality, we obtain

$$\mathcal{L}_{LVB} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \tag{25}$$

Finally, in order for this equation to be computable, we can develop it into a last formulation for training, resulting in a comprehensive diffusion formulation [12]

$$\mathcal{L}_{LVB} \leq \underbrace{\sum_{t=2}^{T} \mathcal{D}_{KL}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \,||\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right)}_{L_1+L_2+\ldots+L_{T-1}} \underbrace{- \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}_{L_0} \tag{26}$$

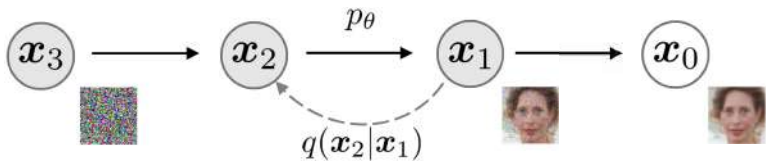

Figure 9: The original diffusion process, from [37]

2.3.2 Score-based models

Score-based models are another class of generative models that focus on estimating the score function of a data distribution, rather than directly modeling the distribution itself [40]. The score function is the gradient of the log-probability of the data with respect to the data itself, denoted as $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. It captures the direction in which the data density increases the fastest (see fig. 10), and knowing it can help to generate new samples by moving towards higher-density regions of the data distribution. Once trained, these models

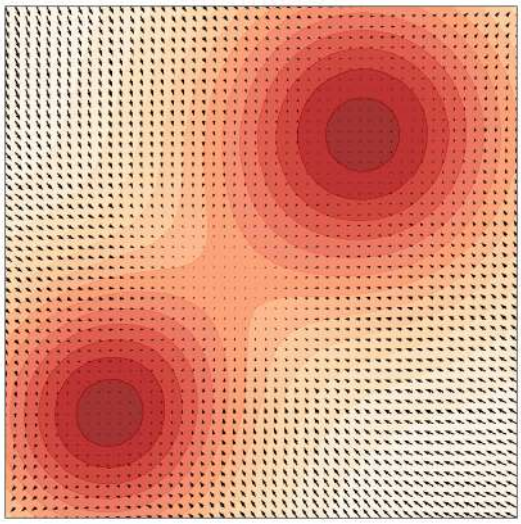

Figure 10: Score-function estimation of a simple 2D distribution

$\mathcal{F}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$ can take random noise as input and iteratively remove it using the learned score function to produce realistic data.
A naive objective to train score-based models is to minimize a MSE loss between the score function and the model

$$\mathcal{L}_{MSE} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[||\nabla_\mathbf{x} \log p(\mathbf{x}) - \mathcal{F}_\theta(\mathbf{x})||_2^2\right] \tag{27}$$

However, the score function generally cannot be expressed directly. Instead, denoising score matching [41] provides an efficient way to minimize this loss without the knowledge of $\nabla_\mathbf{x} \log p(\mathbf{x})$, and can be expressed as

$$\mathcal{L}_{SM} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[tr(\nabla_\mathbf{x}\mathcal{F}_\theta(\mathbf{x})) + \frac{1}{2}||\mathcal{F}_\theta(\mathbf{x})||_2^2\right] \tag{28}$$

This reformulation of the loss function offers a practical framework for learning to model the score function, which can be efficiently optimized using gradient-based methods.

Once trained, score-based models $\mathcal{F}_\theta(\mathbf{x})$ can be employed to generate new data points into the estimated distribution. This is typically achieved using Langevin Dynamics, a technique rooted in non-equilibrium thermodynamics, which has been adapted to machine learning for efficient sampling [42]. Formally, the process begins by initializing the chain from an arbitrary prior distribution $\mathbf{x}_0 \sim \pi(\mathbf{x})$. It then iteratively applies the following update rule

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \sqrt{2\epsilon}\,\mathbf{z}_t \quad \text{with} \quad \mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I}) \tag{29}$$

Under the condition that $\epsilon \to 0$ and $t \to \infty$, we obtain $\mathbf{x} \sim p(\mathbf{x})$. As $\mathcal{F}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$ in the case of score-based models, $\mathcal{F}_\theta(\mathbf{x})$ can be used to generate samples.

2.3.3 Extended frameworks

Suffering from long inference time and poor quality samples, the original diffusion formulation [12] was improved and linked to score-based models [39]. Further research has demonstrated that the loss employed in training DPMs could be simplified into a weighted combination of score matching objectives [36]. This equivalence highlights the shared underlying principles between methods, and has been fundamental to the development of modern diffusion models. Indeed, this connection has spurred the success of various alternative extended frameworks, as seen in works like *Denoising Diffusion Probabilistic Models* (DDPM) [36] and *Denoising Diffusion Implicit Models* (DDIM) [37], and culminated in a unified framework that comprehensively describes this class of generative networks [43]. It has not only clarified the theoretical underpinnings but also facilitated significant performance optimization, resulting in more efficient and scalable implementations. Consequently, diffusion models have emerged as the state-of-the-art in various generative tasks.

Denoising Diffusion Probabilistic Models (DDPM)

DDPM [36] is a diffusion framework directly based on the original DPM results [12], bridging score-based and diffusion models and enhancing both theoretical clarity and performance. First, it introduces a more efficient forward process parametrization for β_t, defining the variance schedule as a sequence of linearly increasing constants from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. Moreover, it presents a simplification of the original loss using score-matching. Indeed, based on 18 and 23, the model to train $\mu_\theta(\mathbf{x_t}, t)$ can be reparametrized to predict

$$\tilde{\mu_t}(\mathbf{x_t}, \epsilon_t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right), \tag{30}$$

which leads to

$$\mu_\theta(\mathbf{x_t}, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x_t}, t) \right). \tag{31}$$

The denoising model ϵ_θ training loss $\mathcal{L}$ is defined relatively to the noise $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$, minimizing the difference between $\tilde{\mu_t}(\mathbf{x_t}, \epsilon_t)$ and $\mu_\theta(\mathbf{x_t}, t)$. In its simplest form, we have

$$\mathcal{L}_{simple,t} = E_{t, \mathbf{x}_0, \epsilon_t}[\, || \, \epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}, \, t)||^2] \tag{32}$$

Finally, in this formulation $\Sigma_\theta(\mathbf{x}_t, t)$ is chosen not to be learnable and hence set to a fixed value with t

$$\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I} \text{ with } \sigma_t = \beta_t \tag{33}$$

This reparametrizations and improvements on the diffusion formulation can be linked with score-based models as for $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2\mathbf{I})$ we have $\nabla_\mathbf{x} \log p(\mathbf{x}) = -\frac{\mathbf{x}-\mu}{\sigma}$. Using 17, the score function written in the diffusion formalism hence verifies

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}, \tag{34}$$

showing that optimizing one or another of these models is similar. This also links the different sampling processes together. However, using Gaussian parameters in the sampling

Markov chain is an approximation [12], which happens to be better verified when the length T of the forward process goes to infinity [36]. As these steps have to be performed sequentially, sampling with a long forward process still implies a long sampling time, which consists of a huge limitation to DDPM usage in the generative field.

Denoising diffusion implicit models

In order to reduce the number of sampling steps, a new formulation for the denoising process has been proposed [37]. By generalizing the previous results from the particular Markovian diffusion to a class of non-Markovian ones, the sequential chain needed to produce a sample can be shortened (See fig. 11). Considering a class Q of inference distributions, indexed by a real positive vector $(\sigma_t)_{0\leq t\leq T}$, the forward process can be rewritten as

$$q_\sigma(\mathbf{x}_{1:T} \,|\, \mathbf{x}_0) = q_\sigma(\mathbf{x}_T \,|\, x_0) \prod_{t=1}^{T} q_\sigma(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t,\, \mathbf{x}_0),$$

with

$$q_\sigma(\mathbf{x}_T \,|\, \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\,\mathbf{x}_0,\, (1-\alpha_t)I),$$

and for all $t > 1$,

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t,\, \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}}\,\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\,\frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1-\alpha_t}},\, \sigma_t^2 I)$$

Each step of the forward process $q_\sigma(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{x}_0)$ can then be obtained via the Bayes' rule, and it can no longer be considered as Markovian, since each x_t depends on both $\mathbf{x}_{t-1}$ and eventually $\mathbf{x}_0$. This forward process is used as a leverage knowledge for the sampling process. Hence, from a given $\mathbf{x}_t$, and a denoising model $\epsilon_\theta^{(t)}(\mathbf{x}_t)$ we first predict an approximate $\mathbf{x}_0$, which can be written as

$$f_\theta^{(t)} = (\mathbf{x}_t - \sqrt{1-\alpha_t}\,\epsilon_\theta^{(t)}(\mathbf{x}_t))/\sqrt{\alpha_t},$$

and use it to predict $\mathbf{x}_{t-1}$

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\,f_\theta^{(t)} + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\,\epsilon_\theta^{(t)}(\mathbf{x}_t) + \sigma_t\,\epsilon_t,$$

where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$ is standard Gaussian noise independant of $\mathbf{x}_t$. Different choices for σ_t give different generative processes. Denoising diffusion implicit model (DDIM) is a special case of this result when $\sigma_t = 0$ for all t. It is an implicit probabilistic model trained with the DDPM objective. This approach is deterministic, as opposed to DDPM which is stochastic, and it allows a consistent generation and interpolation.

Elucidating the design space of diffusion-based generative models

As previously observed, the diffusion process can be formalized using various frameworks [35] [39] [36] [37], often using different scientific backgrounds, and sometimes showing connections between them. In Elucidating the design space of diffusion-based generative

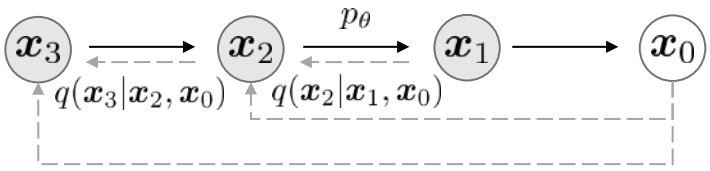

Figure 11: The DDIM diffusion process, from [37]

models (EDM)[43], the authors propose a unification of the different diffusion formulations with a clear explanation of its different components. The paper analyzes the impact of factors such as noise schedules, network architectures, and the use of different objectives, helping to refine diffusion models for more efficient and effective generative performance.

First, instead of modeling the forward and backward processes as discrete sequences of noisy steps, a stochastic differential equation (SDE) called *probability flow* is used. It makes it possible to describe the gradual transformation of data into noise (and vice versa) as a smooth, continuous process, and eliminates the need to specify a fixed number of discrete time steps, offering more flexibility. Moreover, by formulating the diffusion process as a differential equation, score-based models and diffusion probabilistic models can completely be unified under a single framework. Both types of models rely on estimating the gradient or score function of the data distribution, and the continuous nature of the SDE facilitates this connection. For a noise schedule $\sigma(t)$ and a scaling $s(t)$ applied to the input data $\hat{\mathbf{x}}$ so that $\mathbf{x} = s(t)\hat{\mathbf{x}}$, the probability flow is

$$d\mathbf{x} = \left[\frac{\dot{s}(t)}{s(t)} - s(t)^2\,\dot{\sigma}(t)\sigma(t)\,\frac{\nabla_{\mathbf{x}}(D_\theta(\mathbf{x},\,\sigma) - \mathbf{x})}{\sigma^2}\right] dt,$$

where $D_\theta(\mathbf{x}, \sigma)$ is implemented as a denoising neural network, taking an input $\mathbf{x}$ for every noise schedule σ. The denoising model also often uses a previous reparametrization with a σ-dependent scaling that makes the network able to learn either the noise n given to the input y or the objective y itself. To do so, the article introduces a network F_θ from which D_θ is then defined. If F_θ is the network being trained

$$D_\theta(x,\,\sigma) = c_{skip}(\sigma)\,x + c_{out}(\sigma)\,F_\theta(c_{in}(\sigma)\,x,\,c_{noise}(\sigma)),$$

and the overall training loss with a weight $\lambda(\sigma)$ is

$$\mathcal{L} = \mathbb{E}_{\sigma,\,y,\,n}[\,\lambda(\sigma)\,||\,D_\theta(y+n,\,\sigma) - y||_2^2\,]$$

Finally, sampling data is achieved with solving the stochastic differential equation, by using an integration scheme such as Euler or Runge-Kutta, and a discrete sampling time series $t_0, t_1, ..., t_N$. As this numerical solution is an approximation, it introduces a truncation error that accumulates over the steps. Using a Heun's second order integration scheme allows to diminish this error. The introduced new training processes and samplers achieved

excellent generation results with a smaller number of steps than DDIM [37], obtaining a state-of-the-art FID [44] on benchmark datasets such as CIFAR-10 [45] and ImageNet [46], further solidifying diffusion models as leading frameworks in the generative domain.

2.3.4 Manipulation and control

Control in generative models refers to the ability to guide the generative process toward desired outputs by manipulating conditions or intermediate representations during sampling. Diffusion models are particularly well-suited for control because of their inherent flexibility and the iterative nature of their generative process. They can easily be conditioned on auxiliary information such as class labels or text descriptions thanks to classifier-guidance and classifier-free-guidance [47] [17], enabling targeted generation [48]. The U-Net architecture [49] that is generally employed in diffusion models also plays a significant role in enabling this control by capturing multi-scale features and facilitating precise adjustments across different levels of abstraction in the data. In addition, the recent development of diffusion auto-encoders [13] provides a unique form of control by allowing direct manipulation of a latent space, which can be used to modify specific aspects of the generated data such as style or structure.

Classifier guidance and classifier-free guidance

Classifier guidance is a technique designed to direct diffusion sampling towards the direction that maximizes the probability that the generated sample is classified into a specific class [47]. During each denoising step, the update direction $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ provided by the diffusion model can be combined with the gradient obtained from a classifier, which predicts $p(\mathbf{y} \mid \mathbf{x})$, where $\mathbf{y}$ represents an arbitrary input feature such as a class label (See fig. 12). The gradient of the classifier prediction $\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$ is scaled by a factor γ called temperature, and added to the update direction of the diffusion model, resulting in the expression

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \gamma \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) \tag{35}$$

This formulation effectively adjusts the gradient to weight the model's distribution towards the expected direction, thereby increasing the likelihood of generating samples consistent with the target class. However, classifier guidance has the drawback of requiring an external classifier, which must be separately trained and robust to noise due to its application to noisy inputs. To address these limitations, classifier-free guidance offers a modern alternative that eliminates the need for an external classifier model [17]. Instead, this method involves training a single diffusion model as both a conditional generative model, computing $\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y})$, and an unconditional generative model, computing $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ (See fig. 12). Using Bayes' rule, the update direction is reformulated as

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = (1 - \gamma) \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \gamma \nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) \tag{36}$$

This approach combines the conditional and unconditional score functions, typically yielding better performance than classifier guidance because the internally integrated Bayesian

classifier is more robust than a separately trained external model. Furthermore, classifier-free guidance implementation is considered simpler, as it only requires to dropout the conditioning information during training [17].

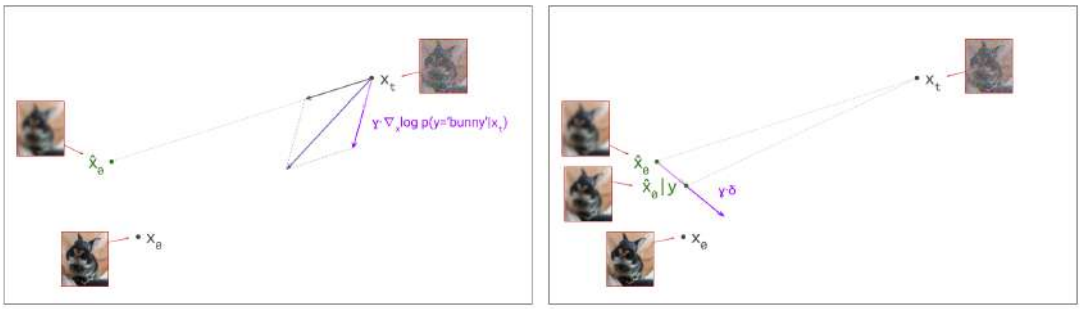

Figure 12: Classifier guidance (left) and free-guidance (right), taken from [50]

Diffusion auto-encoders

Using diffusion models for representation learning offers a powerful framework for controlled generation and manipulation. As explored in *Diffusion Autoencoders: Toward a Meaningful and Decodable Representation* [13], the proposed approach aims to extract a relevant representation of an input data into a semantic latent space.

For input data $\mathbf{x}_0$, a latent variable $\mathbf{z}_{sem}$ is obtained via a neural encoder $\mathbf{E}_\phi$ so that $\mathbf{z}_{sem} = \mathbf{E}_\phi(\mathbf{x}_0)$. A deterministic diffusion decoder is trained to be conditioned with $\mathbf{z}_{sem}$ and model $p(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t,\, \mathbf{z}_{sem})$ to match with the inference distribution $q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t,\, \mathbf{x}_0\,, \mathbf{z}_{sem})$ defined in DDIM [37]. The training objective can be written with the EDM [43] as

$$\mathcal{L} = \mathbb{E}_{\sigma,\,y,\,n}[\,\lambda(\sigma)\,||\,D_\theta(y+n,\,\sigma\,,\mathbf{z}_{sem}) - y||_2^2\,] \tag{37}$$

Supported by experiments on attribute manipulation, this design highlights a hierarchical separation between the high-level information contained in $\mathbf{z}_{sem}$ and a lower-level *stochastic* representation $\mathbf{x}_T$ (See fig. 13). Yet, the article also shows that with a sufficiently large latent space, the encoded data can almost be reduced to its latent representation and the reconstruction be weakly dependant to the original noise $\mathbf{x}_T$. Furthermore, in this context, manipulation on $\mathbf{z}_{sem}$ show that this vector contains rich, meaningful and interpretable features of the input data, justifying its *semantic* character (See fig. 14).

By integrating a reconstruction process with meaningful latent space learning, diffusion auto-encoders facilitate both high-quality sample manipulation and accurate data reconstruction in an unsupervised framework. This dual capability enhances their utility in creative tasks, including audio processing. Hence, the use of diffusion auto-encoders have recently flourished with the development of numerous tools such as *Music2latent* [14], a powerful compression framework also leveraging consistency diffusion models [51] or *Mousai*, a text-to-audio

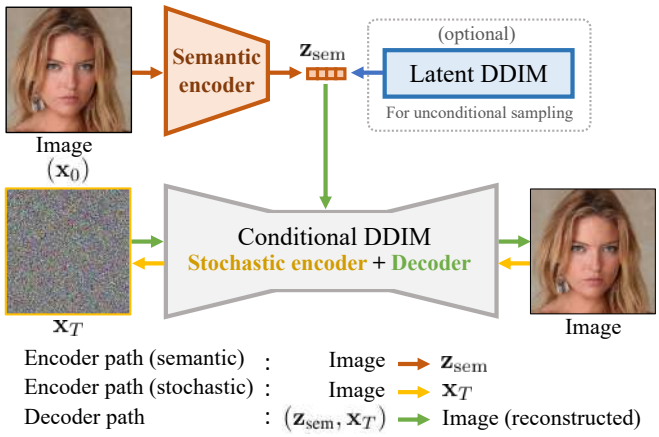

Figure 13: Diffusion auto-encoder architecture, taken from [13]. A Latent diffusion network can be trained on the semantic space for unconditional sampling.

generative framework [19]. However, the semantic latent space of diffusion auto-encoders often contains entangled information, which complicates achieving both expressive and precise manipulation. To address this challenge, hierarchical approaches were explored to better disentangle features and improve manipulation capabilities. Finally, a generarive model trained to sample latent codes from the DAE can also be trained for unconditional sampling (See fig. 13).

Figure 14: Semantic manipulation between images from CelebA [52]. Semantic vectors $z_{sem,1}$ and $z_{sem,2}$ are naivly interpolated to reconstruct a novel picture, showing that both vectors capture meaningful features.

2.4 Hierarchical generative models

2.4.1 First steps with hierarchical models

A hierarchical approach has regularly been introduced in the common generative frameworks [23], [53]. These structures present several interesting aspects. First, they increase the

capacity of the model to represent complex data distributions. Indeed, as each level of the hierarchy can learn different features, it results in richer and more nuanced representations, which is a fundamental aspect for expressive control. Moreover, they can stabilize the training process : higher-level abstractions are learned first, which provide a stable foundation for learning finer details at lower levels. This progressive refinement makes it easier to avoid abrupt changes in the learning stage, ensuring smoother and more controlled training. Finally, they are an important factor in enhancing interpretability, which aids in understanding how different components contribute to the generation or reconstruction process. NVAE (Nouveau VAE) [53] is a good example of a hierarchical VAE that benefits from its use of multi-level latent variables. Each level captures increasingly detailed information about the data, resulting in smoother training and high-quality image generation across datasets like CIFAR-10 [45] and ImageNet. Previously mentioned, progressive GANs [23] also use a hierarchical structure, increasing the complexity of the model during training for a stable process and better sampling quality.

In audio, hierarchical structures have also been exploited in different models. RAVE [7], to begin with, employs a hierarchical latent space to model different aspects of the audio signal. This hierarchical aspect allows the model to learn a low-dimensional, structured latent representation that can generate high-quality music in real time. Jukebox [54] is another hierarchical model designed for music generation. Its hierarchical structure is based on the different temporal levels of music. By using three separates vector-quantized VAEs (VQ-VAEs) [55], this model operates on three different scales, each one dealing with a specific time compression and sample rate. The top level generates the high-level structure (such as melody and harmony), the middle level generates more detailed features (like instrument texture), and the bottom level handles fine details of the waveform. However, such separations in classical architectures are often constrained by the inherent limitations of these models. These challenges could be addressed and enhanced through modern frameworks like diffusion models, which introduce greater stability in training and enable finer control over generative processes.

2.4.2 Hierarchical Diffusion autoencoders

The development of hierarchical diffusion auto-encoders (HDAE) [3] introduced a novel approach to hierarchical latent representation within the diffusion framework. Arguing that the original DAE formulation [13] could offer a more comprehensive representation of semantic features, HDAE postulates that feature maps at various depths in the encoder network capture distinct levels of abstraction. Consequently, semantic codes are extracted at different layers of the encoder, each representing progressively higher-order features, that can be integrated into the conditional deterministic U-Net denoising process for enhanced generation and manipulation (See fig. 15). This formalism introduces a finer control over the semantic representations, exploiting the *coarse-to-fine* and *low-level-to-high-level* feature hierarchy of the semantic encoder [3].

The article introduces various formulations of HDAE, using different types of encoders

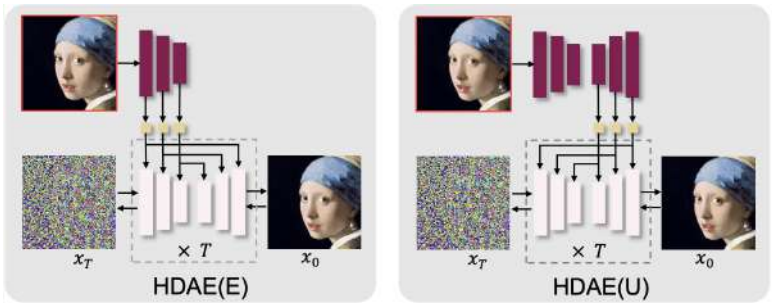

Figure 15: hierarchical diffusion auto-encoder, from [3], with a simple encoder (left) and a U-Net encoder (right)

for enhanced semantic separation (See fig. 15). Additionaly, and beyond showing better reconstruction perfomances than traditional DAE, experiments involving various manipulations like style mixing and interpolation have proven the semantic space to include comprehensive features. For example, different inputs from different encoding sources can be combined at their respective depth level to influence the generation result (See fig. 16). The feature disentanglement showed in this article hence demonstrates a powerful manipulation technique that might also work on audio.

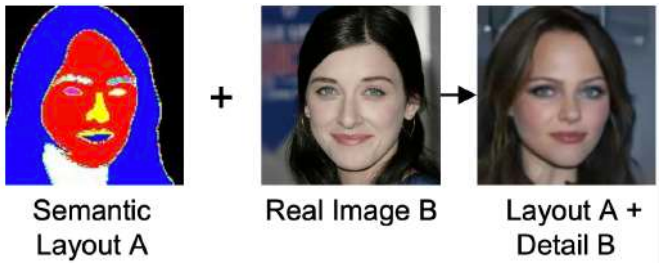

Figure 16: Multimodal semantic image synthesis, taken from [3]

3 Multi-scale spectral diffusion auto-encoder

In this section, we introduce our proposal and detail the reasoning behind its conception. Directly using a diffusion model to generate latent codes and perform manipulations is limited by entanglement in the high-level latent features, that remain beyond the scope of usual labels for conditioning. However, we showed previously that DAEs [13] and more recently HDAEs [3] are able to extract a semantically meaningful representation of the data that is used to guide the inference process. Yet, current researches have mostly pursued text-to-music applications [19], or obtaining more accurate audio codecs [14]. Although the technical performances of such results are impressive, their creative impact and control capabilities are strongly limited from a musical point of view. As discussed in the introduction of this work, we believe that relevant levels of audio information about the sound structure and character pertain to different temporal and frequential scales.

In order to manipulate audio and ultimately control the generative process, we seek to design both a model and training process that reflect these compositional time-frequency aspects. Hence, motivated by its hierarchical structure, we propose to extend the HDAE formulation with a novel *multi-scale spectral diffusion auto-encoder* formalism, working on different levels of quantification to allow for expressive hierarchical control.

3.1 Model definition

Our objective is to encode an audio signal into three hierarchical representations, progressively capturing increasing levels of detail, and leverage these representations as conditioning inputs for a U-Net denoiser D_θ. Similarly to HDAE [3], the semantic vectors are used to guide the diffusion process in its decoding phase. Hence, the low-dimensional semantic vector, which is expected to mostly contain structural information is utilized near the U-Net bottleneck, where the compression ratio is highest. In contrast, the finer-grained details, expected to be represented by the less-compressed higher-level semantic vector, are applied at the top layers of the U-Net. Our complete workflow is illustrated on figure 17. These successive semantic vectors can be grouped into a single $\mathbf{z}_{hierarchical}$, leading to the following variant of the EDM [43] formulation

$$\mathcal{L} = \mathbb{E}_{\sigma,\, y,\, n}[\, \lambda(\sigma)\, ||\, D_\theta(y + n,\, \sigma\, , \mathbf{z}_{hierarchical}) - y||_2^2\,]. \tag{38}$$

However, we think it is essential to integrate hierarchical separation at the core of the encoding process, as learning these representations with identical training processes offers no guarantee *a priori* that the model will appropriately allocate features across the correct levels of abstraction. Therefore, we choose to work with a spectral representation of the audio, leading to a more structured and interpretable representation compared to raw waveform data. Indeed, a time-frequency organization of sound is particularly well-suited for multi-scale analysis, facilitating feature extraction across varying temporal and frequency resolutions. Hence, our encoding process starts by mapping the original audio waveform into three spectral representations, obtained by first transforming the audio signal into a Mel-Spectrogram, and then successively downsampling this representation into coarser

ones. Denoting $\mathbf{x}$ the original audio signal, the multi-scales representations $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$, $\tilde{\mathbf{x}}_3$ are obtained following

$$\begin{cases} \tilde{\mathbf{x}}_3 = MelSpec_{\text{high-res}}(\mathbf{x}) \\ \tilde{\mathbf{x}}_2 = MelSpec_{\text{mid-res}}(\mathbf{x}) = downsample(\tilde{\mathbf{x}}_3) \\ \tilde{\mathbf{x}}_1 = MelSpec_{\text{low-res}}(\mathbf{x}) = downsample(\tilde{\mathbf{x}}_2) \end{cases} \tag{39}$$

To project these representations into informative condition vectors, we design encoders E_1, E_2, E_3 that progressively reduce the input dimensionality while preserving its temporal structure. These encoders map $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$, $\tilde{\mathbf{x}}_3$ into latent representations $\tilde{h}_1$, $\tilde{h}_2$ and $\tilde{h}_3$

$$\begin{cases} h_1 = E_1(\tilde{\mathbf{x}}_1) \\ h_2 = E_2(\tilde{\mathbf{x}}_2) \\ h_3 = E_3(\tilde{\mathbf{x}}_3) \end{cases} \tag{40}$$

However, our design choice for the three representations may still induce redundancy, as $\tilde{\mathbf{x}}_3$ inherently holds all the information found in both $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$. Thus, there is no intrinsic motivation for h_1 and h_2 to contain meaningful *independent* features. To ensure proper hierarchical separation, we apply a slight refinement of the learned features h_2, and h_3 before using them as conditioning information. At each representation level $k \in [\![1, 3]\!]$ we do $\tilde{h_k} = h_k + upsample(h_{k-1})$ (and initialize $\tilde{h_1} = h_1$), allowing them to capture finer variations around the low-level features rather than encoding the same information. The U-Net denoiser is finally conditioned with these modified latent vectors $\tilde{h}_1$, $\tilde{h}_2$, and $\tilde{h}_3$.

To further ensure that all representations are meaningful, we implement a progressive training scheme, drawing inspiration from *Progressive growing of GANs* developped by Karras et al. [23]. Hence, we propose our own multi-stage training sequence to support this hierarchical learning process. Specifically, we start by only training the encoder E_1 which takes a partial representation of the original data as input. Then, we progressively introduce details at each subsequent level $k + 1$ by freezing $E_1, ..., E_k$ and training only E_{k+1}. This strategy ensures that each encoder is optimized for its respective scale of representation, leading to a more disentangled and structured latent space.

When used for inference, the model is designed to operate with multiple input samples. Indeed, we can provide three distinct audio for at each level, each chosen for a specific structural or timbral characteristic. In that case, the model functions as a neural sampler capable of selecting and integrating the corresponding hierarchical features from each snippet, providing a powerful audio manipulation tool. Furthermore, by training prior generative models to sample from these ordered latent spaces, we can establish a generative framework that enables novel, controllable sampling. Manipulating these latent spaces offers a new dimension of control in neural synthesis, potentially allowing for more precise and creative audio generation.

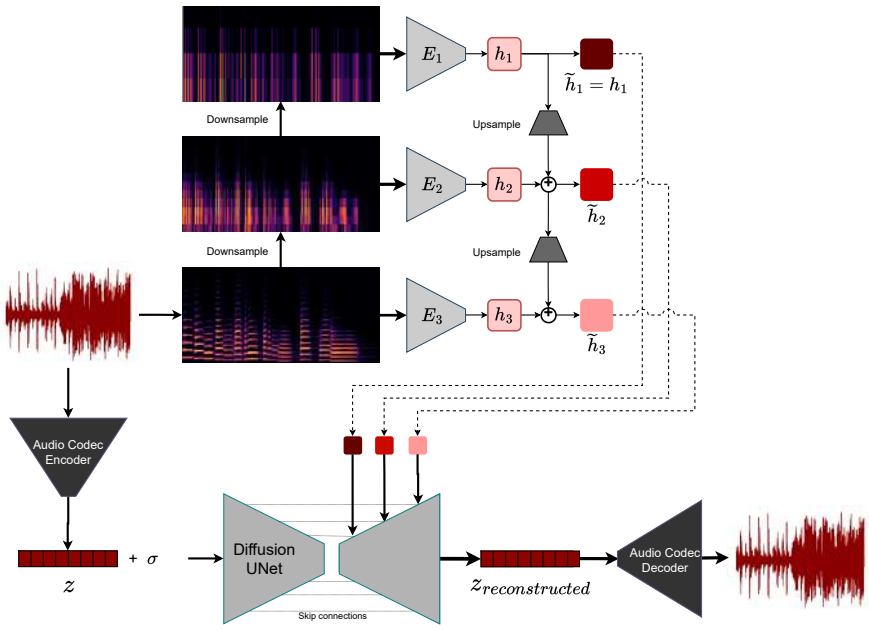

Figure 17: The complete workflow of our model proposal.

3.2 Architecture details

In this section, we detail our architectural and hyperparameter choices [1]:

- Our diffusion network D_θ is a U-Net including skip connections with five encoding and decoding stages. This network uses 1D convolutions instead of the original 2D convolutions of the original article [49], and we apply a time compression of factor 2 at each level. Conditioning information from the semantic encoder is applied at levels 1, 3, and 5 in the decoding process, letting one decoding stage between each. As the latent vectors are time-dependent, conditioning is done through simple concatenation with the current signal being processed, while noise levels are inserted into the network using adaptive group normalization [56].

- The spectral representations we utilize are Mel spectrograms. To enhance feature separation, we apply downsampling with a factor of 2 on both the time and frequency scales for each new signal.

- All encoders E_1, E_2 and E_3 are built with 2D convolutions that also downsample the time scale by a factor 2 at each stage, and are designed for their output to be easily concatenated with the U-net stage they correspond to.

[1] The complete code is available at https://github.com/AlfredPichard/multiscale-spectral-DAE

4 Experiments and results

Evaluating our model is a critical step to ensure that it performs adequately in musical applicative scenarios. In this section, we aim to validate the effectiveness of our approach, by targeting its robustness and generalization of the proposed architecture. To thoroughly assess our model, we focus on three key aspects: *reconstruction quality*, *feature analysis* and *manipulation capabilities*. Evaluating reconstruction quality is crucial to determine how accurately the model can recreate the original signal from its compressed representation. High-quality reconstruction indicates that the model effectively preserves essential information during the encoding and decoding processes, and is fundamental for potential artistic applications. For the latent analysis, we chose two salient features at different hierarchical level for each dataset and analyse how these features are distributed within the different encoders latent spaces. Finally, we evaluate the manipulation capabilities of our approach, by targeting specific transforms such as pitch or timbre modification. This assessment reflects the core objective of our model and its practical utility in music production. For each evaluation, we compare the performances of our model against a baseline HDAE method to determine their relative effectiveness.

4.1 Experimental setup

Dataset

For our experiments, we used the NSynth [10] and Slakh [57] datasets. Considered as a benchmark in the audio deep learning field, Nsynth is composed of 305,979 musical notes from ten different instruments, which allow our reconstruction process to be evaluated on a large variety of sounds. Slakh provides 2100 synthesized track stems providing full pieces for different instruments, which can extend our evaluation process on audio manipulation. We retain only the individual tracks of non-percussive instruments, leading to more than 200 hours of audio. Although these datasets are not suited for artistic applications, using them offers objective frameworks that simplify our evaluations, thanks to the accompanying metadata they provide, such as pitch for NSynth and full midi score for Slakh.

Baseline

As a baseline method, we adapted the HDAE approach [3] to audio data. Specifically, we employ 1D convolutions instead of the 2D convolutions typically used for images. Additionally, since sound contains temporal information, our semantic vectors must reflect this characteristic to effectively reconstruct the data. Consequently, the latent representations generated by the semantic encoder are time-dependent, with each representation compressing information at a distinct temporal level. We specifically designed this formulation of HDAE with three hierarchical encoding levels to ensure a fair comparison with our model.

Training Details

As mentioned in section 3, we rely on a neural audio codec to compress the raw waveform into a representation of lower dimensionality, reducing the computational complexity of the generation. We employ the recently proposed Music2Latent [14], which compresses

44.1 kHz audio into 10 Hz latent codes with 64 channels. We train both models on codes corresponding to 3-seconds long audio chunks, each for the same number of steps. We use the Adam optimizer [24] with a learning rate of $5e-5$. For the full-resolution Mel-Spectrogram, we use a hop length of 256, FFT and window size of 1024, and 128 Mel bands.

4.2 Audio reconstruction

Before delving into feature analysis and assessing the control capabilities of our proposed architecture, we first focus on estimating reconstruction quality. We use the generic Fréchet Audio Distance (FAD) [58] method, which compares embeddings extracted from original and reconstructed audio using a pretrained model, such as CLAP [18], to measure the distance between the original and generated distributions. The FAD is particularly useful for evaluating generative audio models, offering a quantitative measure of how closely the generated audio resembles real audio samples. These results are displayed in table 1. We observe that our models obtain strongly better reconstruction than the HDAE baseline on Slakh, and performs equivalently on NSynth.

	FAD Score ↓	
	NSynth	Slakh
HDAE	**0.55**	0.55
Ours	0.56	**0.41**

Table 1: Comparison of FAD scores between the baseline HDAE and our proposed approach.

4.3 Feature analysis

We built our proposed model with the very goal of disentangling low-level (e.g., notes and onsets) and high-level (e.g., timbre or instruments) features in the distinct latent spaces of our different encoders. In this section, we conduct several experiments to observe and analyze the structure of the latent spaces of our model. A common method for evaluating learned features in deep learning is to train small neural networks, often called *probes* or *auxiliary networks*, which take representations as inputs, and are then trained to extract specific information from the representations (e.g pitch, melody, instrument). Reaching a high validation score for one of these tasks indicates that the representation contains the necessary information for the prediction. Oppositely, lower accuracy would suggest that the model has not encoded those features in that specific representation. Hence, this approach lets us assess how well the learned features capture various aspects of the audio data.

In our case, we train probes on each different latent representations to classify two distinct features. As low-level feature, we focus on the pitch of the note played for NSynth and the whole melody in the case of the Slakh dataset. For both dataset, we focus on instrument labels as high-level feature.

To predict single pitch or instrument labels, we train a MLP network on time averages of the latent representations using a cross-entropy loss. We treat melody extraction as a temporal multi-label classification problem, and follow the methodology presented in [59]. Namely, we start by resampling the representation to match the length of the sequence of labels. Then, we use a two-layer BiLSTM followed by a linear layer, and train the model to extract melody from the resampled representation using binary cross-entropy (BCE) loss. We present our results for the Nsynth and Slakh datasets in tables 2 and 3 respectively.

	Pitch prediction ↑				Instrument ↑			
	h_1	h_2	h_3	Codec	h_1	h_2	h_3	Codec
HDAE	**96%**	85%	8%	97%	**99%**	83%	26%	99%
Ours	76%	82%	**87%**		**97%**	94%	94%	

Table 2: Comparison of classification accuracy for MLP classifiers trained on the three encoders representations h_1, h_2 and h_3.

	Melody prediction ↓				Instrument ↑			
	h_1	h_2	h_3	Codec	h_1	h_2	h_3	Codec
HDAE	**0.007**	0.009	0.022	0.002	**97%**	47%	14%	99%
Ours	0.015	**0.006**	**0.006**		**91%**	86%	79%	

Table 3: Comparison of classification scores for classifiers trained on the three encoders representations for the Slakh dataset. For melody prediction, the BCE loss value is reported instead of accuracy.

For the NSynth dataset, we can observe different behaviours for the HDAE baseline and our proposed model. The h_1 and h_2 representations for HDAE capture most pitch and timbre information, whereas h_3 has relatively low predictability scores. Interestingly, our model separates well pitch information, mainly captured in the representation h_3, while timbre is captured in the high level representation h_1. We observe a similar trend for our model trained on Slakh, with a better prediction of the melody on the h_3 and h_2 codes, while the high-level feature h_1 exhibits the highest accuracy for instrument prediction. HDAE behaves similarly, with the low-level dimension h_3 capturing very little information for both melody and instrument.

The latent codes from the codec should contain all the pitch and timbre information as they can be used to reconstruct almost perfectly the audio samples from the two datasets. The very high prediction scores obtained by the classifier probes on those latent codes demonstrate that our evaluation method is able to correctly detect the presence (or absence) of the pitch and timbre features within the multiple representations.

Next, we employ the UMAP [60] algorithm to vizualize the underlying data manifolds of each hierarchical representation. UMAP is a versatile and efficient algorithm for dimensionality reduction, offering useful insights into the structure of complex datasets. It projects high-dimensional data onto a low-dimensional manifold maintaining both local neighborhoods and global relationships. For both our model and HDAE, we plot pitch and instrument labels on top of the UMAP projection to outline how the different features are organised in the latent space.

We present the UMAP visualizations generated from the analysis of the HDAE latent spaces on the Slakh dataset, highlighting the distribution by instrument (fig. 18). Additional UMAP visualizations are provided in the appendix (see appendix A).

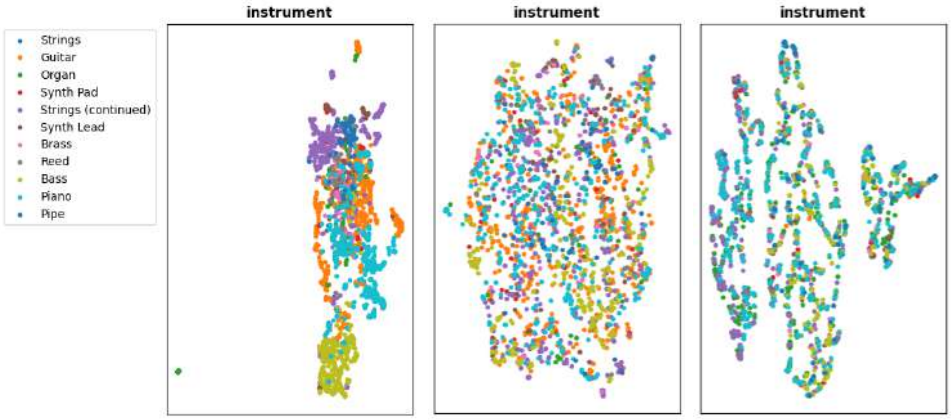

Figure 18: UMAP anlaysis of the HDAE latent spaces. The graph on the left corresponds to representation h_1, in the middle h_2 and on the right h_3.

This analysis reveals that the highest-level latent space exhibits clustered instrument representations. This suggests that this feature is likely captured by this latent space and indicates that this level has effectively learned to encode this information. Oppositely, the two other representations seem not to contain a specific *instrument-based* organization.

4.4 Latent manipulations

After analysing how the pitch and timbre information are distributed inside the different hierarchical representations of the two models, we must evaluate the performance of our approach in generation. As our goal is to build a generative model with disentangled representation spaces, mixing representations from different audio sources should generate new realistic signals that preserve the features from each source. For instance, we observed in the previous section that for our proposed model, h_1 mainly captures timbre, whereas h_2 and h_3 focus on the melodic content. Hence, we evaluate in this section whether, when conditioning our model using h_1 from a sample $\mathbf{x}_1$, and h_2 and h_3 from a sample $\mathbf{x}_2$, the

generated output matches the timbral properties of $\mathbf{x}_1$, and the melody of $\mathbf{x}_2$. For each dataset and model, we generate samples by swapping one attribute with that of another sample, denoted as the *target*, while keeping the other two untouched. For the generated audio, we evaluate the timbre similarity, as well as accuracy of the melodic content with respect to the attributes of the target.

To assess timbre similarity, we employ the metric introduced in SS-VAE [61]. This method uses a triplet network trained to identify whether samples are played by the same instrument, based on Mel-frequency cepstral coefficients 2-13. We rely on the official implementation and learn the metric using the Mixing-Secrets 4 dataset [62]. For both datasets, we extract the midi transcription of the generated audios using BasicPitch [63], and compare it with that of all three inputs. For Slakh, we compute the similarity between melodies by using the F1-score from the mir_eval package [64]. For NSynth, we compute the accuracy between the pitch of the generated audio and that of $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$.

	Pitch accuracy ↑				Timbre similarity ↑			
	h_1	h_2	h_3	Rec.	h_1	h_2	h_3	Rec.
HDAE	**0.29**	0.06	0.01	**0.74**	0.59	0.74	0.69	0.97
Ours	0.03	0.06	**0.16**	0.47	0.61	0.62	**0.84**	0.95

Table 4: Comparison of transfer scores for Nsynth dataset on the three representations h_1, h_2 and h_3, with a reconstruction reference Rec.

	Melody F1-score ↑				Timbre similarity ↑			
	h_1	h_2	h_3	Rec.	h_1	h_2	h_3	Rec.
HDAE	0.039	**0.047**	0.025	0.18	**0.81**	0.60	0.58	0.93
Ours	**0.090**	0.055	0.022	0.27	0.63	0.62	**0.75**	0.95

Table 5: Comparison of transfer scores for Slakh dataset on the representations h_1, h_2 and h_3, with a reconstruction reference Rec.

On the Slakh dataset, we observe a coherent behaviour between our previous feature analysis and the generation performance. Indeed, h_3 obtains the highest score in terms of timbre similarity, meanwhile h_1 has the most impact on the generated melody. Interestingly, on Nsynth the last layer h_3 seems to have a large effect on both timbre and pitch. As NSynth is a very simple dataset, where each sample correspond to a single note, pitch can be seen as a high-level feature, quite similar to timbre. It would be interesting to investigate how other temporal descriptors with a finer temporal definition, such as the envelope of the note played, are potentially captured by the lower-level latent variables. It is also important to note that both melody accuracy and timbre similarity metrics are affected by the audio quality of the generated samples, as both evaluation methods rely on pretrained neural

networks that might not be robust to noise and low quality samples. Our evaluation could benefit from a perceptive study with specific focus on timbral and melodic similarities.

The HDAE model does not exhibit a clear trend in the effect of latent manipulations on timbre and melody, although h_1 and h_2 seem to share most of the effect on the generated sample features. We observe that on top of improved reconstruction accuracy, our proposed model seems to be a more robust solution to perform independent attribute manipulation, and represents a promising tool for musical applications such as sample editing.

5 Discussion

5.1 Contributions and limitations

This research aimed to design a model that captures the hierarchical structure of music, providing a controllable tool for composition. Leveraging recent advancements inside the diffusion framework, such as DAE and HDAE, we initially explored the application of hierarchical approaches from image manipulation to audio synthesis. Our initial adaptation of HDAE to audio was driven by the expectation that the disentanglement observed in image data would similarly enhance audio representations.

However, we discovered significant differences in the structural hierarchies of audio and image data, which informed us to develop a novel approach tailored specifically for audio signals. This new model integrates key elements from HDAE while addressing the unique characteristics of audio through signal processing techniques. Furthermore, we introduced a progressive training strategy to effectively disentangle the various latent representations inherent in audio data.

An essential aspect of our work involved establishing a suitable evaluation framework, as traditional testing methods are not well-adapted to our objectives. Additionally, the numerous hyperparameters involved require careful optimization to achieve optimal performance. Ongoing evaluation efforts will be critical in guiding future improvements and ensuring the model's practical applicability in music composition. However, even if our model faces limitations primarily due to its training complexity and the challenge of effective evaluation, we observed that it provides a promising latent separation, and the comparison to our baseline highlights the positive results of our approach.

5.2 Future work

In conclusion, while our evaluation process incorporates promising concepts, it still requires refinement to more accurately measure the effectiveness of our proposed model. Improving the evaluation framework will be a critical next step to fully validate and optimize the ideas we have developed.

Once we establish robust evaluation methods, the next logical progression is to train our model on real musical data rather than relying on benchmark datasets. This shift will pave the way for creative applications, allowing us to test and challenge our model in real-world, artistic contexts.

Looking ahead, developing priors for the various latent spaces in our model would enable the sampling of latent vectors, transforming our approach into a fully generative and manipulable framework. This extension is an exciting direction for future exploration, with significant potential for creative and innovative uses in music composition.

References

[1] Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Long-form music generation with latent diffusion. *arXiv preprint arXiv:2404.10301*, 2024.

[2] Shih-Lun Wu, Chris Donahue, Shinji Watanabe, and Nicholas J. Bryan. Music controlnet: Multiple time-varying controls for music generation, 2023.

[3] Zeyu Lu, Chengyue Wu, Xinyuan Chen, Yaohui Wang, Lei Bai, Yu Qiao, and Xihui Liu. Hierarchical diffusion autoencoders and disentangled image manipulation, 2023.

[4] Max Matthews. The digital computer as a musical instrument. *American Association for the Advancement of Science*, 1963.

[5] Zicarelli.D. Max/msp, 2002.

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

[7] Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high quality neural audio synthesis. nov 2021.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[10] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

[11] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression, 2022.

[12] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[13] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation, 2022.

[14] Marco Pasini, Stefan Lattner, and George Fazekas. Music2latent: Consistency autoencoders for latent audio compression, 2024.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[16] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing, 2020.

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[18] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap: Learning audio concepts from natural language supervision, 2022.

[19] Flavio Schneider, Ojasv Kamal, Zhijing Jin, and Bernhard Schölkopf. Moûsai: Text-to-music generation with long-context latent diffusion, 2023.

[20] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation, 2024.

[21] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae, 2018.

[22] Nils Demerlé, Philippe Esling, Guillaume Doras, and David Genova. Combining audio control and style transfer using latent diffusion, 2024.

[23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[25] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

[26] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[29] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[30] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017.

[31] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation, 2016.

[32] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2015.

[33] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models, 2017.

[34] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[35] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv:1503.03585v8 [cs.LG]*, 2015.

[36] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[37] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[38] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis, 2021.

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

[40] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020.

[41] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 01 2005.

[42] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[43] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[44] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[45] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[47] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

[48] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[50] Sander Dieleman. The geometry of diffusion guidance, 2023.

[51] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023.

[52] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[53] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder, 2021.

[54] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[55] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.

[56] Heliang Zheng, Jianlong Fu, Yanhong Zeng, Jiebo Luo, and Zheng-Jun Zha. Learning semantic-aware normalization for generative adversarial networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21853–21864. Curran Associates, Inc., 2020.

[57] Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.

[58] Azalea Gui, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou. Adapting frechet audio distance for generative music evaluation, 2024.

[59] Ruibin Yuan, Yinghao Ma, Yizhi Li, Ge Zhang, Xingran Chen, Hanzhi Yin, Yiqi Liu, Jiawen Huang, Zeyue Tian, Binyue Deng, et al. Marble: Music audio representation benchmark for universal evaluation. *Advances in Neural Information Processing Systems*, 36:39626–39647, 2023.

[60] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[61] Ondrej Cifka, Alexey Ozerov, Umut Simsekli, and Gael Richard. Self-supervised vq-vae for one-shot music style transfer. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021.

[62] Siddharth Gururani and Alexander Lerch. Mixing secrets: a multi-track dataset for instrument recognition in polyphonic music. *Proc. ISMIR-LBD*, pages 1–2, 2017.

[63] Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore, 2022.

[64] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. Mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, volume 10, page 2014, 2014.

A Appendix

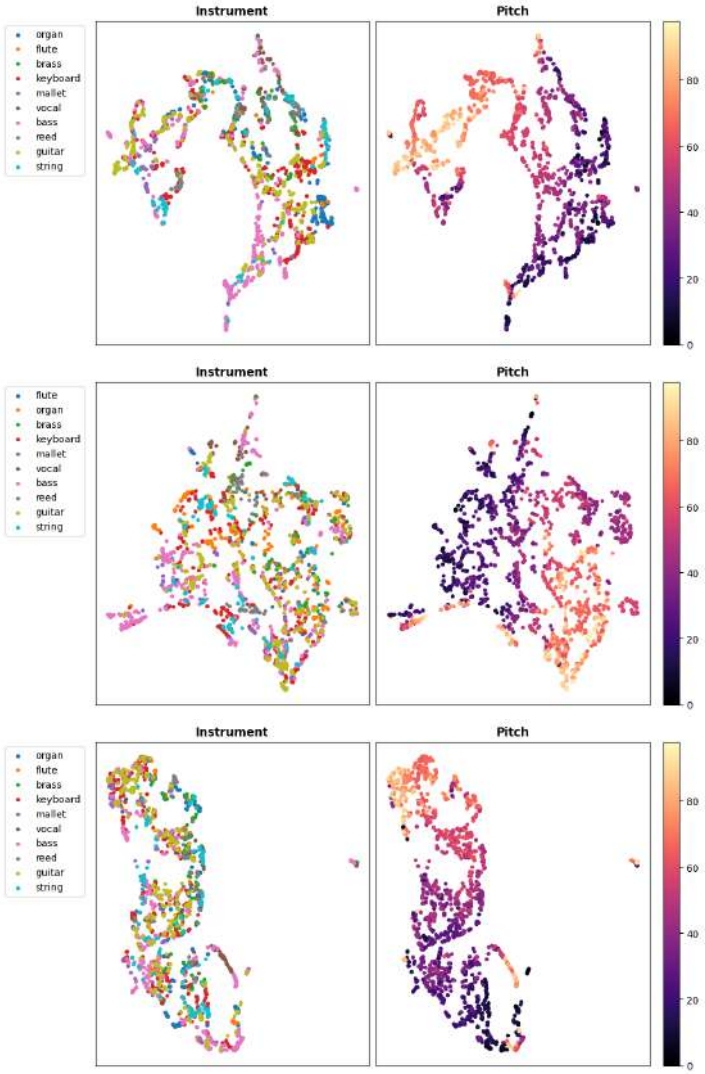

Figure 19: UMAPs for h_1, h_2 and h_3 in our model on Nsynth dataset

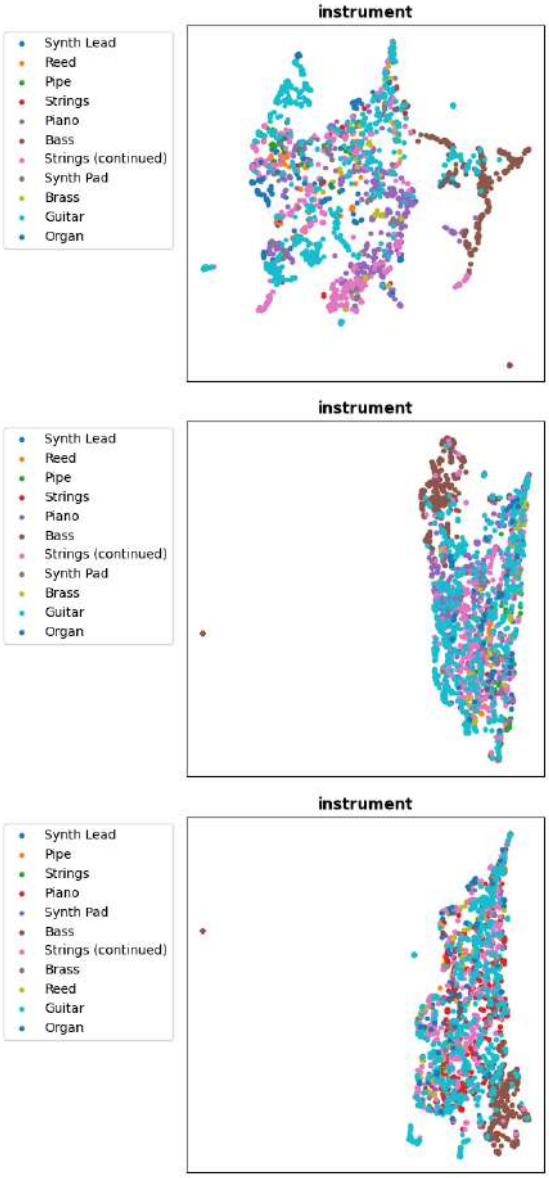

Figure 20: UMAPs for h_1, h_2 and h_3 in our model on Slakh dataset

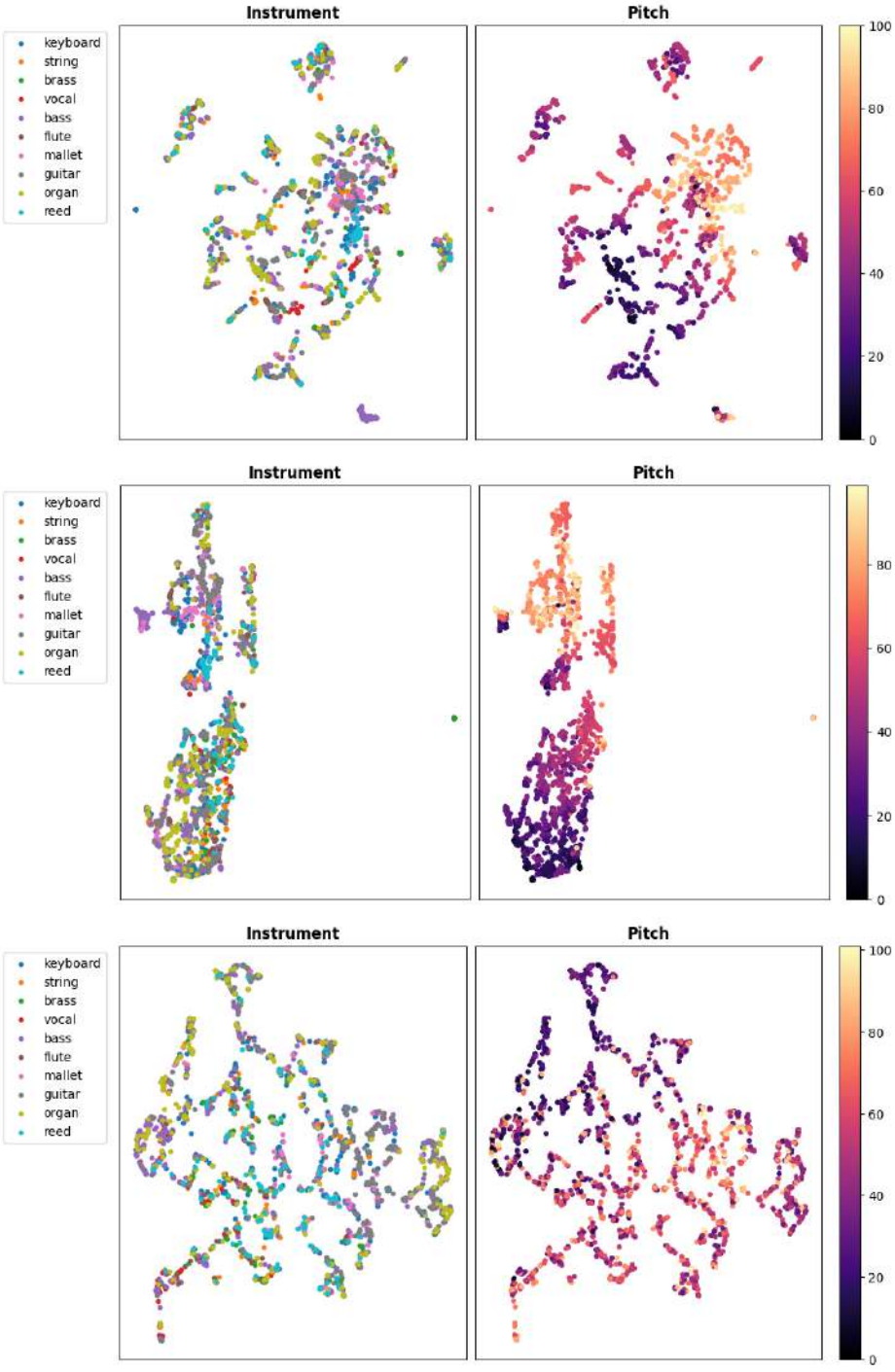

Figure 21: UMAPs for h_1, h_2 and h_3 in HDAE on Nsynth dataset