SAPIENZA
UNIVERSITÀ DI ROMA

Generative Diffusion Models for Audio Inpainting

Facoltà di Ingegneria dell'informazione, informatica e statistica
Engineering in Computer Science

Andrea Rodriguez
ID number 1834937

Advisor
Prof. Danilo Comminiello

Academic Year 2022/2023

Generative Diffusion Models for Audio Inpainting
Master thesis. Sapienza University of Rome

© 2023 Andrea Rodriguez. All rights reserved

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: rodriguez.1834937@studenti.uniroma1.it

A nonna Vera

Contents

Chapter 1

Introduction

The field of audio restoration and inpainting has witnessed significant advancements in recent years, driven by the increasing demand for high-quality audio processing. One promising approach in this domain is the utilization of generative diffusion models. In this master's thesis, we focus on the utilization of techniques to extend the capabilities of generative diffusion models to the audio restoration domain, opening up new possibilities for audio inpainting and denoising.

Our research builds upon the foundation laid by previous studies on generative diffusion models and their successful applications in the image domain. However, we go a step further by adapting these techniques to the unique characteristics and challenges posed by audio signals. By harnessing the power of generative diffusion models and integrating it with state-of-the-art image inpainting techniques tailored for audio, we aim to address the specific intricacies of audio inpainting and offer robust and effective solutions for reconstructing missing or degraded audio segments.

Additionally we explore the integration of denoising techniques. While generative diffusion models excel in reconstructing missing audio segments, they also have the potential to effectively reduce the impact of noise on audio signals. Leveraging the advancements in denoising algorithms designed for images, we adapt and apply these techniques to audio data, aiming to improve the overall quality and clarity of the inpainted audio and providing a comprehensive solution for audio restoration and inpainting.

The following chapters will be structured in this way:

- **Chapter 2: Background.** This chapter provides a comprehensive overview of the fundamental concepts and theories necessary to understand the subsequent

work. It covers essential topics such as Diffusion Models, Spectrograms, and the tasks of Audio generation and Audio inpainting.

- **Chapter 3: State-of-the-Art generative models.** In this chapter, a thorough survey of state-of-the-art generative models for audio generation will be presented. Prominent approaches, including AudioLDM [1] and TANGO [2], will be explored, shedding light on their architectural designs and capabilities in generating realistic audio content. The chapter aims to provide insights into the cutting-edge techniques used in the field.

- **Chapter 4: Diffusion models for audio inverse problems.** This chapter focuses on selected techniques originally designed for image inpainting, which have been adapted to the audio domain. Techniques such as DDNM [3] and RePaint [4] will be discussed in detail, examining their effectiveness and suitability for audio inpainting applications. The chapter aims to provide a comprehensive analysis of these methods and their applicability to audio restoration.

- **Chapter 5: Evaluation.** This chapter presents the results of extensive experimentation and evaluation conducted on the proposed audio inpainting techniques. Relevant metrics and evaluation criteria used to assess the performance and fidelity of the inpainting methods will be introduced. A comparative analysis will be performed to highlight the strengths and limitations of each approach, providing valuable insights into their effectiveness in various scenarios.

- **Chapter 6: Audio Inpainting in Communication Scenarios.** This chapter explores the potential applications of audio inpainting techniques in communication systems. The integration of denoising and inpainting capabilities will be discussed, highlighting how it can enhance audio quality in scenarios where noise and signal degradation occur during transmission. The chapter also addresses the benefits and challenges associated with utilizing audio inpainting in communication contexts.

- **Chapter 7: Conclusions and Future works.** The final chapter summarizes the key findings and contributions of the thesis. It provides a concise overview of the research conducted, highlights the implications of the results, and offers suggestions for future directions in the field of audio inpainting.

Chapter 2

Background

2.1 Diffusion models

Diffusion models [5] have emerged as a powerful class of generative models that have gathered significant attention in recent years. These models operate on the principle of iteratively transforming an initial noise distribution into a target distribution, offering the ability to generate high-quality samples and perform a range of tasks such as generation, denoising, and inpainting. Diffusion models provide a unique approach to modeling complex data distributions by leveraging the sequential refinement of the noise distribution through a series of diffusion steps. In this paragraph, we delve into the fundamental concepts of diffusion models.

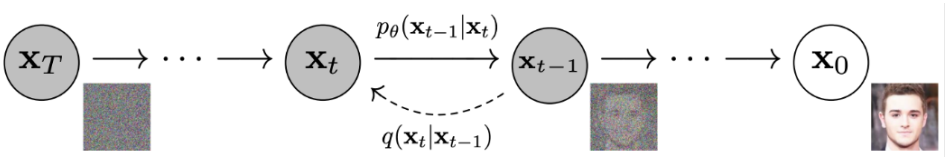


Figure 2.1. Diffusion models basic idea [5]

2.1.1 Forward diffusion process

The forward diffusion process lies at the core of diffusion models and is at the basis of their generative capabilities. In this process, the initial noise distribution is progressively transformed into the target distribution through a sequence of invertible transformations. The mathematical formulation of the forward diffusion process involves defining the transformation steps that update the noise distribution at each iteration. These transformations aim at gradually reducing the discrepancy between the generated samples and the target distribution by modeling the underlying data structure.

Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, we define a forward diffusion process in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$.

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

The data sample $\mathbf{x}_0$ gradually loses its distinguishable features as the step t becomes larger. A nice property of the above process is that we can sample $\mathbf{x}_t$ at any arbitrary time step t in a closed form using the reparameterization trick. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

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

2.1.2 Reverse diffusion process

The reverse diffusion process is a fundamental component of diffusion. In this process, the goal is to map a sample from the target distribution back to the initial noise distribution by inverting the forward diffusion steps. This inverse mapping enables tasks such as reconstruction, denoising and inpainting.

Conceptually, the reverse diffusion process involves iteratively applying the inverse transformations to the target sample, gradually recovering the initial noise distribution. By reversing the transformations, diffusion models aim at reconstructing the latent variables or clean versions of the observed data. The reverse diffusion process exploits the fact that the forward and inverse transformations are designed to be invertible, ensuring that the mapping from the target distribution to the initial noise distribution is well-defined.

One significant challenge in the reverse diffusion process is solving the inverse problem. Given a sample from the target distribution, the model needs to infer the corresponding initial noise distribution that could have generated that sample. If we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because it needs to use the entire dataset and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run

the reverse diffusion process.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

The reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

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

Using the Bayes' rule and following the standard Gaussian density function, the mean and variance can be parameterized as follows:

$$\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0$$

We can represent $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_t)$ and plug it into the above equation to obtain:

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_t\Big)$$

2.1.3 UNet

UNet [6] is a popular neural network architecture. It was originally proposed for biomedical image segmentation tasks but has since been widely adopted for various tasks. The UNet architecture is renowned for its ability to capture fine-grained details while maintaining a global context understanding.

The key characteristic of the UNet architecture is its U-shaped design, which consists of an encoder path and a corresponding decoder path. The encoder path resembles a traditional convolutional neural network (CNN) and gradually reduces the spatial dimensions of the input while extracting high-level features. This downsampling process is achieved through repeated application of convolutional layers with pooling operations, allowing the network to capture and encode contextual information effectively.

The decoder path of the UNet architecture performs the reverse operation of the encoder. It gradually upsamples the features by employing transposed convolutions or upsampling layers, allowing the network to recover the spatial resolution lost during the encoding stage. The decoder path also includes skip connections that concatenate the corresponding feature maps from the encoder path. These skip

connections serve as shortcuts, providing the decoder with detailed local information while maintaining a global context understanding. By integrating the skip connections, the UNet architecture facilitates the fusion of multi-scale information, enabling precise localization and segmentation of objects in the image.

The skip connections in the UNet architecture play a vital role in its success. They allow the network to combine both low-level and high-level features, enabling the precise localization of objects while capturing contextual information. This design choice ensures that the network can effectively handle both fine details and global structures. The skip connections also help address the problem of information loss during the downsampling and upsampling processes, enabling the network to reconstruct accurate and detailed output predictions.

The UNet architecture has found successful applications in the context of diffusion models. By integrating the UNet structure into the design of diffusion models, researchers have been able to enhance the generative capabilities and inference tasks of these models. The U-shaped encoder-decoder design of UNet complements the forward and reverse diffusion processes, enabling diffusion models to capture fine-grained details while maintaining a global context understanding. The skip connections in the UNet architecture, which facilitate the fusion of multi-scale information, prove to be especially beneficial in diffusion models, as they allow precise localization and reconstruction during the reverse diffusion process. By leveraging the UNet architecture within diffusion models, researchers have achieved improved performances in various tasks. The adaptability and effectiveness of the UNet architecture make it a valuable asset in advancing the capabilities and applications of diffusion models.

2.1.4 Training and inference

We need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process. We would like to train μ_θ to predict $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\Big)$. Because $\mathbf{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict ϵ_t from the input $\mathbf{x}_t$ at time step t:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\Big)$$

$$\mathbf{x}_{t-1} = \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\Big), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$

The loss term L_t is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$. Empirically, Ho et al. (2020) [5] found that training the diffusion model works better with a simplified objective that ignores the weighting term:

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[\|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2\right] \\ &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[\|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)\|^2\right] \end{aligned}$$

Generating samples from trained diffusion models involves running the forward diffusion process starting from the initial noise distribution and performing a fixed number of diffusion steps. Each transformation step updates the noise distribution, gradually refining it towards the target distribution. This procedure allows for the generation of diverse and high-quality samples from the learned target distribution. By controlling the sampling temperature or utilizing conditioning information, the generated samples can be tailored to exhibit specific characteristics or adhere to certain constraints.

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Figure 2.2. Training and Sampling algorithms [5]

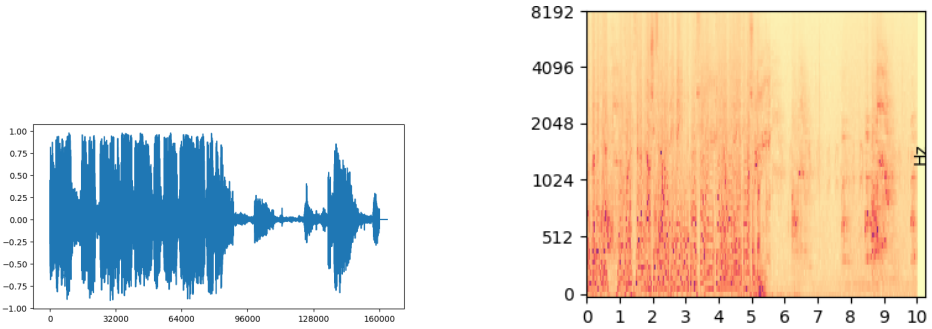
2.2 Spectrograms

Spectrograms offer a powerful and intuitive way to visualize and analyze the frequency content of audio over time. By converting the audio waveform into a spectrogram, we can gain insights into the spectral characteristics, temporal dynamics, and structural components of the audio.

A spectrogram is a visual representation of the frequency content of an audio signal over time. It is obtained by applying a Fourier transform to short-time frames of the audio signal and mapping the resulting frequency spectrum onto a two-dimensional grid. The magnitude or power of each frequency component is typically represented by a color or intensity value, with time on the horizontal axis and frequency on the vertical axis.

The construction of spectrograms involves several steps. First, the audio signal is

divided into short overlapping frames, typically ranging from 10 to 100 milliseconds, to capture the temporal dynamics. Each frame is then transformed into the frequency domain using a windowed Fourier transform, such as the Short-Time Fourier Transform (STFT). The resulting frequency spectrum is often converted to a logarithmic scale to better represent the perceptual characteristics of human hearing. Finally, the magnitude or power spectrum is mapped to the color or intensity values to generate the spectrogram.



(a) Audio wave **(b)** Spectrogram

Figure 2.3. Audio wave and corresponding spectrogram

When working with audio data in deep learning applications, the choice of representation can significantly impact the efficiency and effectiveness of the models. Traditional deep learning models often operate on raw audio waveforms, which consist of a series of consecutive samples representing the air pressure variations over time. While waveforms provide a direct representation of the audio signal, they suffer from inherent sparsity. This means that a vast majority of the waveform samples are typically close to zero, resulting in a large amount of redundant and irrelevant information.

The sparsity of audio waveforms poses challenges for deep learning models, as they have to process and analyze a considerable number of consecutive samples, which can be computationally demanding and increase the risk of overfitting. Furthermore, capturing long-term dependencies and extracting meaningful features from sparse waveform data can be challenging.

In contrast, spectrograms offer a more compressed and structured representation of audio data. By transforming the audio waveform into the frequency-time domain, spectrograms provide a two-dimensional visual representation of the audio signal. This representation condenses the temporal information into discrete time frames and the frequency content into the vertical axis, resulting in a much denser and

more informative data representation.

The compression achieved by spectrograms enables deep learning models to focus on relevant spectral features and temporal patterns, rather than processing a large number of individual waveform samples. Spectrograms capture the frequency content at different time points and allow the models to extract meaningful patterns and relationships. This compressed representation facilitates more efficient training and inference processes, as well as the ability to capture long-term dependencies more effectively.

2.3 Audio generation

Audio generation, the process of creating new and original audio content, has long been a fascinating and challenging task in the field of digital signal processing. With the advent of deep learning audio generation has seen significant advancements, pushing the boundaries of what is possible in terms of creating realistic, diverse, and expressive audio signals. Deep learning models have revolutionized the field, enabling the generation of music, speech, sound effects, and other forms of audio with remarkable fidelity and creativity.

The goal of audio generation using deep learning is to train models that can autonomously produce audio that is indistinguishable from human-created or real-world audio. By leveraging the power of neural networks, these models learn from vast amounts of audio data and discover underlying patterns and relationships. This enables them to generate new audio samples that possess similar characteristics to the training data, while also showcasing novel and innovative attributes.

2.3.1 Text-to-Audio

Text-to-audio generation includes not only the synthesis of speech but also the generation of audio based on textual descriptions. While speech synthesis focuses on converting written text into spoken words, text-to-audio generation involves transforming textual descriptions into corresponding audio representations.

Text-to-audio generation has evolved, expanding into the generation of audio that represents specific sounds, environmental ambiance, musical compositions, or even abstract audio concepts. By leveraging deep learning techniques, researchers have developed models that can understand textual descriptions and translate them into coherent and expressive audio signals. This advancement opens up possibilities for

applications in sound design, music production, audiovisual media, virtual reality experiences, and more.

2.4 Audio Inpainting

Audio inpainting involves the reconstruction of missing or corrupted portions of audio signals, aiming to restore the original audio content. This task has practical applications in various domains, including audio restoration, speech denoising, music production, and audio synthesis.

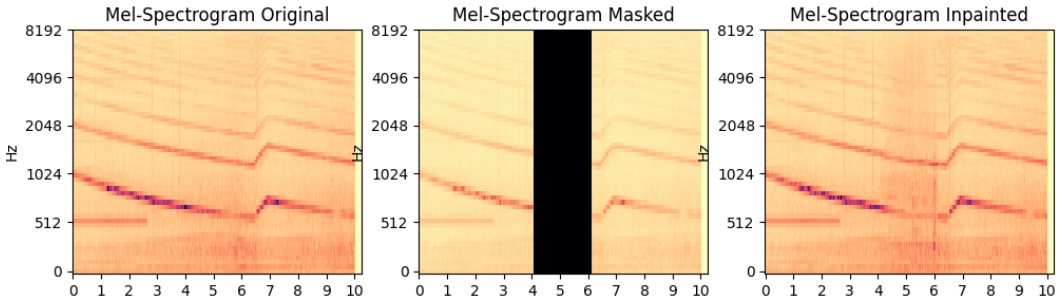


Figure 2.4. Inpainting task

The need for audio inpainting arises due to various reasons. During audio recording or transmission, audio signals can be subject to various forms of degradation, such as packet loss, transmission errors, or hardware malfunctions, leading to missing or corrupted sections in the audio. Additionally, in post-production and editing scenarios, it is often necessary to remove or replace certain portions of audio due to noise, artifacts, or undesired content. In such cases, audio inpainting techniques play a vital role in reconstructing the missing or altered parts, ensuring a seamless listening experience.

One of the key challenges in audio inpainting is effectively handling longer gaps in the audio signal, where the missing or corrupted sections span several seconds or more. Furthermore, ensuring coherence and continuity in the inpainted audio is crucial for a seamless listening experience. The reconstructed portions should seamlessly blend with the surrounding audio, matching the tonal characteristics, rhythm, and spatial attributes. To achieve these characteristics, it is required that the underlying temporal dependencies and complex audio patterns are captured, which can pose significant difficulties for traditional signal processing-based approaches.

Numerous approaches have been proposed to tackle the audio inpainting problem.

Traditional signal processing-based methods often rely on techniques such as linear prediction, spectral interpolation, and waveform interpolation. These methods employ statistical models and interpolation techniques to estimate the missing or corrupted audio segments based on the surrounding context. While these approaches have shown effectiveness in certain scenarios, they may struggle to capture long-term dependencies and intricate audio textures, resulting in perceptually suboptimal reconstructions.

More recently, deep learning-based methods have emerged as a powerful alternative for audio inpainting. They allow for the learning of complex audio patterns and the extraction of meaningful features, which can aid in reconstructing missing or corrupted audio sections. Deep neural networks can effectively model the contextual information from the surrounding audio, enabling the generation of plausible and perceptually coherent inpainted audio.

Understanding the limitations of existing audio inpainting techniques motivates the exploration of generative diffusion models as a promising alternative approach. By applying diffusion models to audio inpainting, it is possible to capture the intricate dynamics of audio signals and generate high-quality inpainted audio that is perceptually faithful to the original content.

Chapter 3

State-of-the-Art generative models

In this chapter, we focus on two existing diffusion models for audio generation: AudioLDM [1] and Tango [2]. These models serve as the baselines for our study and will be described based on their original proposed architectures in their respective papers. Both models stem from recent publications, representing state-of-the-art approaches in audio generation using diffusion models. Our aim is to provide a detailed description and analysis of these baseline models. By understanding the architectural choices, training strategies, and performance characteristics of AudioLDM and Tango, we establish a foundation for evaluating and comparing advanced techniques for audio inpainting in subsequent chapters.

3.1 AudioLDM

AudioLDM (Audio Latent Diffusion Model) is a novel Text-to-Audio (TTA) system that leverages a latent space to learn continuous audio representations through contrastive language-audio pretraining (CLAP) [7] latents. By utilizing pretrained CLAP models, they are able to train Latent Diffusion Models (LDMs) with audio embeddings while incorporating text embeddings as conditions during the sampling process. Furthermore, AudioLDM exhibits exceptional computational efficiency, making it a practical and scalable solution. By focusing on the latent space rather than explicitly modeling the cross-modal relationship, AudioLDM reduces the complexity of the generation process, resulting in faster inference times and lower resource requirements. AudioLDM was trained on the AudioCaps dataset [8] using a single GPU. The results showcase the state-of-the-art performance of AudioLDM in TTA tasks, as evidenced by both objective metrics and subjective evaluations.

3.1.1 Architecture

AudioLDM consists of three essential components: contrastive language-audio pretraining (CLAP), the latent diffusion model (LDM) [9], and the mel-spectrogram/audio variational autoencoder (VAE). The CLAP stage focuses on encoding the input audio descriptions into a shared audio-text-aligned embedding space. This facilitates the alignment of audio and textual information. Using reverse diffusion, the encoded textual representation is utilized to generate a latent representation of the audio or audio prior by incorporating standard Gaussian noise. The mel-spectrogram VAE's decoder then reconstructs a mel-spectrogram from the obtained latent audio representation. Finally, the mel-spectrogram is passed through a vocoder, resulting in the generation of the final audio waveform.

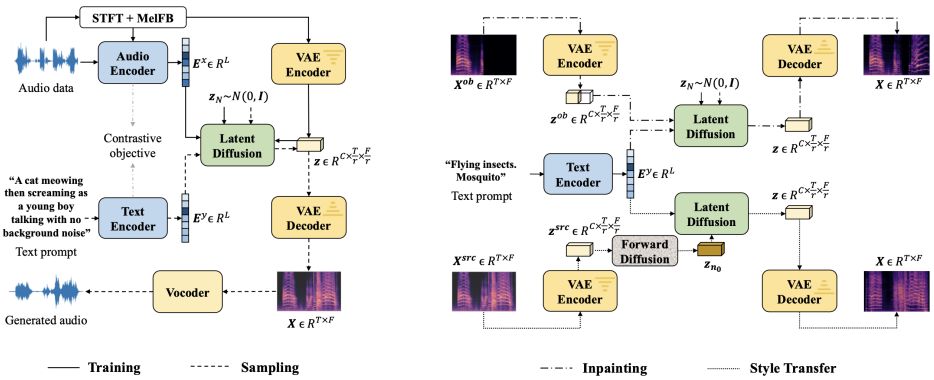


Figure 3.1. AudioLDM Architecture [1]

Conditioned generation: CLAP AudioLDM extends its generative capabilities through conditioned generation, a technique that allows the model to synthesize audio samples based on specific conditioning information. One such approach is the Conditional Latent Audio Prior (CLAP) [7]. CLAP enables AudioLDM to generate audio samples conditioned on additional input variables, such as text prompts or other audio samples. By incorporating these condition variables into the generation process, AudioLDM gains the ability to produce audio samples tailored to specific contexts or desired attributes.

CLAP aims to capture the rich semantic and acoustic information present in audio signals and align it with textual context. By pretraining models using CLAP, we can effectively learn latent representations that facilitate various downstream tasks. The CLAP framework typically involves training a contrastive objective function that encourages audio and text embeddings to be close in the shared latent space, while simultaneously pushing away irrelevant audio and text pairs. This contrastive

learning setup allows the model to capture the semantic relationships between audio and text, enabling it to associate the acoustic patterns in the audio signals with their corresponding linguistic context.

They build the audio encoder on HTSAT [10] and the text encoder on RoBERTa [11]. Once the CLAP model is trained, we are able to extract the representation of an audio sample, which encompasses both audio and text information.

Latent space: VAE To compress the mel-spectrogram into a compact latent space, AudioLDM employs a Variational Autoencoder (VAE) [12] architecture. The VAE consists of two main components: an encoder and a decoder, both comprising stacked convolutional modules. Through the training process, AudioLDM utilizes various loss functions to optimize the VAE's performance. These include a reconstruction loss, which ensures the fidelity of the mel-spectrogram reconstruction; an adversarial loss, which promotes the synthesis of realistic samples; and a Gaussian constraint loss, which encourages the learned latent space to follow a Gaussian distribution.

During the sampling process, the decoder component of the VAE is leveraged to reconstruct the mel-spectrogram from the audio prior generated by the Latent Diffusion Model. The use of VAE-based compression allows us to achieve a small latent space size, which is crucial for the efficiency and effectiveness of the LDM without compromising the quality of the generated samples. In the default configuration, they set the compression level to 4, as it achieves a balance between computational efficiency and generation quality.

To further enhance the performance of the VAE and guarantee the reconstruction quality of the generated compositional samples, data augmentation techniques are applied. This augmentation process helps improving the robustness and generalization capability of the VAE by introducing variations in the input data, thereby enhancing the quality of the reconstructed mel-spectrograms.

The latent space in AudioLDM plays a crucial role in capturing the underlying structure and semantic information of audio signals. By leveraging the VAE, AudioLDM can generate audio samples by sampling from the latent space and feeding them through the decoder network, resulting in diverse and high-quality audio synthesis.

Diffusion The diffusion process, which was introduced and discussed in detail in the previous chapter, forms a crucial component of the AudioLDM framework.

In AudioLDM, the diffusion process is employed to model the generation of audio samples from a latent space. Here, we examine the specific details of how the diffusion process is applied within the context of AudioLDM.

To begin, AudioLDM leverages the UNet backbone architecture as its basic structure. The UNet backbone, with its encoder and decoder blocks, plays a pivotal role in capturing the hierarchical features of the audio data. By conditioning the UNet on both the time step and the CLAP embedding, AudioLDM incorporates temporal and semantic information into the generation process. They directly merge the conditioning information with the feature map of the UNet convolution block using the feature-wise linear modulation layer. This fusion process ensures that the conditioning information is integrated into the feature extraction process of the UNet backbone.

The UNet backbone has four encoder blocks, a middle block, and four decoder blocks. With a basic channel number of c_u, the channel dimensions of encoder blocks are $[c_u, 2c_u, 3c_u, 5c_u]$. The channel dimensions of decoder blocks are the reverse of encoder blocks, and the channel of the middle block has $5c_u$ dimensions. They add an attention block in the last three encoder blocks and the first three decoder blocks. Specifically, they add two multi-head self-attention layers with a fully-connected layer in the middle as the attention block. The number of heads is determined by dividing the embedding dimension of the attention block with a parameter c_h. They propose two version of the model: AudioLDM-S with $c_u = 128$, $c_h = 32$, and AudioLDM-L with $c_u = 256$, $c_h = 64$, respectively.

The diffusion steps are guided by a noise schedule, which determines the intensity of the added noise at each step. In AudioLDM, a linear noise schedule is employed, where the noise level increases linearly from the initial step to the final step. This noise schedule allows for a smooth and controlled transition from the latent space to the audio sample space. During the forward process of AudioLDM, they utilize $N = 1000$ steps and employ a linear noise schedule that ranges from $\beta_1 = 0.0015$ to $\beta_N = 0.0195$.

The diffusion process in AudioLDM is performed over a fixed number of steps. Each diffusion step involves updating the latent variables based on a noise source and conditioning information. The conditioning information, derived from the CLAP embedding and the time step embedding, provides essential context and guidance during the generation process.

For classifier-free guidance [13] in AudioLDM, they introduce a guidance scale of 2.0. This scale influences the balance between the guidance term and the noise term in the diffusion process, allowing them to control the impact of the guidance signal on the generated samples.

During the sampling process, they employ the DDIM (Denoising Diffusion Probabilistic Models) sampler, as proposed by Song et al. [14]. This sampler facilitates efficient and high-quality audio synthesis by performing 200 sampling steps and reducing inference time.

Vocoder: HiFi-GAN To generate high-quality audio samples from the reconstructed mel-spectrogram, AudioLDM employs the HiFi-GAN vocoder introduced by Kong et al. [15]. The HiFi-GAN architecture is specifically designed for high-fidelity audio synthesis, capable of producing realistic and natural-sounding audio waveforms. It employs two sets of discriminators, a multi-period discriminator, and a multi-scale discriminator, to enhance perceptual quality. HiFi-GAN processes input samples at a sampling rate of 16000Hz and extracts a 64-band mel-spectrogram. It follows the default settings of HiFi-GAN V1, with a window size, FFT size, and hop size of 1024, 1024, and 160 respectively. The frequency range is set from 0 to 8000 (f_{min} to f_{max}).

The process begins by taking the reconstructed mel-spectrogram as input to the HiFi-GAN vocoder. The vocoder leverages a sophisticated neural network architecture that incorporates components such as upsampling layers, residual blocks, and waveform upsampling modules. By employing the HiFi-GAN vocoder, we can effectively bridge the gap between the reconstructed mel-spectrogram and the final audio sample. The vocoder synthesizes the audio waveform, leveraging the learned knowledge from the training process to generate high-fidelity audio that closely aligns with the original audio content. By utilizing Hi-Fi GAN, AudioLDM ensures that the generated audio samples maintain the fine-grained details and natural characteristics of real-world audio.

3.1.2 Dataset and Training

The datasets utilized to train AudioLDM consist of AudioSet (AS) [16], AudioCaps (AC) [8], Freesound (FS), and the BBC Sound Effect library (SFX). AudioSet is the largest dataset, featuring 527 labels and over 5,000 hours of audio data. AC, on the other hand, is a smaller dataset comprising approximately 49,000 audio clips

with corresponding text descriptions. Both AudioSet and AudioCaps predominantly contain in-the-wild audio sourced from YouTube, implying that the audio quality is not guaranteed. To diversify the dataset and include higher-quality audio data, they crawled data from FreeSound and BBC SFX, which offer a broad range of categories such as music, speech, and sound effects.

In terms of the audio sample durations, AudioSet and AudioCaps consist of 10-second segments, while the FreeSound and BBC SFX datasets contain longer audio samples. To prevent overuse of the data from lengthy audio segments, which may contain repetitive sounds, only the initial thirty seconds of audio are used from both the FreeSound and BBC SFX datasets. Subsequently, they segment these audio samples into ten-second clips. Consequently, the complete dataset is composed of 3,302,553 ten-second audio samples for training the models. It is important to note that even though certain datasets, such as AudioCaps and BBC SFX, provide text captions for the audio, they were not employed during LDM training. All the audio clips were resampled to a 16kHz sampling rate and mono format, with all samples padded to a duration of 10 seconds.

For each LDM model, the default compression level employed is $r = 4$. AudioLDM-S and AudioLDM-L were trained for 0.6 million steps on a single NVIDIA RTX 3090 GPU, with batch sizes of 5 and 8, respectively. The learning rate was set to 3×10^{-5}. Additionally, AudioLDM-L-Full was trained for 1.5 million steps on an NVIDIA A100 GPU, with a batch size of 8 and a learning rate of 10^{-5}. To optimize performance specifically on AudioCaps, AudioLDM-L-Full was further fine-tuned on the AudioCaps dataset for 0.25 million steps before conducting evaluations.

3.1.3 Inpainting

In the context of audio inpainting, the objective is to generate the missing portion of an audio signal based on the observed part, denoted as x^{ob}. They address the task by incorporating the observed part, x^{ob}, into the latent representation, z.

During the reverse diffusion process, which starts from the initial latent representation $z_N \sim \mathcal{N}(0, \mathbf{I})$, the generated latent representation z_{n-1} are modified at each step using the following equation:

$$z'_{n-1} = (1 - m) \odot z_{n-1} + m \odot z^{ob}_{n-1}$$

Here, z' is the modified latent representation, m represents an observation mask in the latent space, and z_{n-1}^{ob} is obtained by adding noise to z^{ob} using the forward diffusion process.

The values of the observation mask m depend on the observed part of the mel-spectrogram. Since a convolutional structure is used in the VAE to learn the latent representation z, the spatial correspondency in the mel-spectrogram is roughly preserved. Specifically, if a time-frequency bin is observed, the corresponding observation mask is set in the latent space as 1. This allows to retain the ground-truth observation, z^{ob}, while generating the missing information conditioned on the text prompt using TTA models.

3.2 Tango

Similar to AudioLDM, TANGO is a Text-to-Audio (TTA) model. Its main novelty lies in the adoption of an instruction-tuned LLM architecture, specifically FLAN-T5 [17], as the text encoder for text-to-audio (TTA) generation. Previous approaches either utilized a jointly pre-trained text-audio encoder or employed non-instruction-tuned models like T5. In contrast, the proposed approach demonstrates superior performance compared to the state-of-the-art AudioLDM on various evaluation metrics, while maintaining comparable performance on the remaining metrics on the AudioCaps test dataset. Notably, the LDM in TANGO is trained on a 63 times smaller dataset and the text encoder remains frozen. This improvement in performance can be also attributed to the adoption of audio pressure level-based sound mixing during the training set augmentation, which differs from the random mixing employed in previous methods.

In this section, we will primarily highlight the key distinctions between the architecture of TANGO and that of AudioLDM, as they share many similarities.

3.2.1 Architecture

TANGO comprises three key components: the textual-prompt encoder, the latent diffusion model (LDM), and the mel-spectrogram/audio variational autoencoder (VAE). The textual-prompt encoder is responsible for encoding the input description of the audio. Using reverse diffusion, the encoded textual representation is used to build a latent representation of the audio or audio prior from standard Gaussian noise. Next, the decoder of the mel-spectrogram VAE reconstructs a mel-spectrogram from the latent audio representation. Finally, the mel-spectrogram is fed into a vocoder

to generate the final audio waveform.

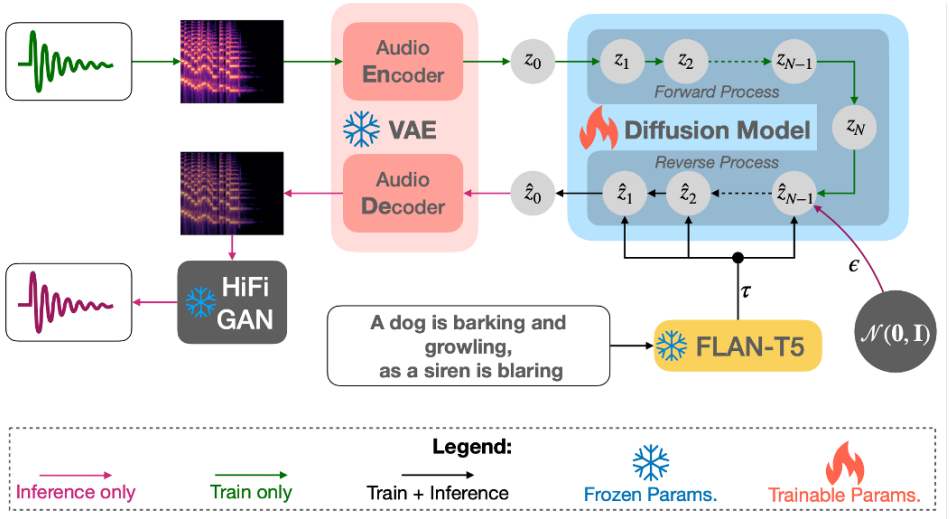


Figure 3.2. Tango Architecture [2]

Conditioned generation: FLAN-T5 The text encoder in TANGO is the pre-trained LLM FLAN-T5-LARGE [17] (780M) and it is used to produce text embeddings. The FLAN-T5 models, trained on a large-scale chain-of-thought (CoT) [18] and instruction-based dataset, possess the ability to learn new tasks effectively by mimicking gradient descent through attention weights. This property is advantageous compared to older large models like RoBERTa and T5, as it enables to learn the mapping between textual and acoustic concepts without fine-tuning the text encoder. The richer pre-training of FLAN-T5 allows the encoder to emphasize key details with reduced noise and enriched context, leading to improved transformation of textual concepts into their acoustic counterparts.

In TANGO the text encoder is kept frozen, assuming that the subsequent reverse diffusion process will effectively learn the inter-modality mapping for audio prior construction. Fine-tuning it may potentially degrade its in-context learning ability due to gradients from the audio modality, which is out of distribution to the pre-training dataset. This differs from a previous approach (AudioLDM), where the pre-trained text encoder is fine-tuned as part of the text-audio joint-representation learning (CLAP) for audio prior reconstruction from text.

Diffusion The latent diffusion model (LDM) [9] in TANGO is derived from AudioLDM, aiming to generate the audio prior z_0 while incorporating the guidance from text encoding. The noise estimation component is parametrized using a UNet

architecture, enhanced with a cross-attention component to incorporate text guidance. In contrast, AudioLDM utilizes audio guidance during training and text guidance during inference, leveraging the pre-trained joint text-audio embedding model (CLAP).

To guide the reconstruction of the audio prior z_0 during the reverse diffusion process, a classifier-free guidance [13] approach based on the input text is used. During inference, a guidance scale w is employed to regulate the influence of the text guidance on the noise estimation, in comparison to the unguided estimation, when an empty text is passed to the model.

Latent space (VAE) and Vocoder (HiFi-GAN) TANGO utilizes the same audio encoder and decoder as AudioLDM, leveraging the pre-trained audio VAE checkpoint provided by the authors of AudioLDM. Additionally, to convert the mel-spectrogram generated by the audio VAE decoder into an audio waveform, TANGO employs HiFi-GAN [15], just like AudioLDM.

3.2.2 Dataset and Training

The primary experiments for text-to-audio generation are conducted using the AudioCaps dataset [8]. This dataset consists of 45,438 audio clips, each paired with a corresponding human-written caption for training purposes. The validation set contains 2,240 instances. The audio clips are ten seconds in duration and were sourced from YouTube videos. Originally, these clips were part of the larger AudioSet dataset [16], which was created for the audio classification task.

In TANGO, they generate additional text-audio pairs by superimposing existing audio pairs and concatenating their captions to enhance the cross-modal concept-composition capabilities of diffusion networks. Unlike the approach of AudioLDM, which employs random combinations of audio pairs, they adopt a different strategy and consider human auditory perception for the fusion process [19]. Specifically, they take into account the audio pressure level to ensure that samples with high pressure levels do not overpower those with low pressure levels.

For training the latent diffusion model (LDM), only the paired (text, audio) examples from the AudioCaps dataset were used. The evaluation of the model was also performed on the AudioCaps test set. This test set includes five different human-written captions for each audio clip. To ensure consistent evaluation with the work of AudioLDM, one caption per clip was randomly selected as the text prompt

for generating the audio signal using the model.

The FLAN-T5-LARGE text encoder was frozen and only the parameters of the latent diffusion model were trained. The diffusion model was built upon the Stable Diffusion UNet architecture, which comprises a total of 866M parameters. In the UNet model, 8 channels were employed and the cross-attention dimension was set to 1024.

For training, the AdamW optimizer was employed with a learning rate of 3×10^{-5} and linear learning rate scheduler during the training process. The model was trained for 40 epochs on the AudioCaps dataset. Training TANGO involved using four A6000 GPUs, and it took approximately 52 hours to complete 40 epochs, with validation performed at the end of each epoch. During training, the per-GPU batch size was 3, which includes 2 original instances and 1 augmented instance, and 4 gradient accumulation steps were performed.

3.2.3 Inpainting

In the provided code of TANGO, inpainting functionality was not included. To incorporate inpainting, we adopted a common method used in diffusion models, which was also employed in AudioLDM. Specifically, we denote the ground truth as x, the unknown part as $m \odot x$, and the known part as $(1-m) \odot x$. Since each reverse step from x_t to x_{t-1} relies solely on x_t, we have the flexibility to modify the known regions $(1-m) \odot x_t$ while maintaining the correct properties of the corresponding distribution. Since the forward process follows a Markov Chain with added Gaussian noise, we can sample the intermediate x_t at any point in time using the equation:

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

This allows us to sample the know regions $m \odot x_t$ at any time step t. As a result, we obtain the following expression for one reverse step in our approach:

$$x_{t-1} = m \odot x_{t-1}^{known} + (1-m) \odot x_{t-1}^{unknown}$$

Here, x_{t-1}^{known} is sampled using the known part $m \odot x_0$, while $x_{t-1}^{unknown}$ is sampled from the model based on the previous iteration x_t. These two components are then combined to form the new sample x_{t-1} using the mask.

Chapter 4

Diffusion models for audio inverse problems

In this chapter, our focus will be on exploring two techniques (and the respective enhanced versions) that we experimented with for audio inpainting. These techniques are originally designed for image inpainting and we have adapted them to suit the audio domain. Both techniques are specifically designed to be applied to generative diffusion models without any fine-tuning or network modifications.

In this study, we extended the concepts introduced in the DDNM [3] and RePaint [4] papers, originally designed for image processing, to audio samples. Although these frameworks were initially developed for images, we explored their applicability to audio by leveraging the structure of spectrograms. Since the diffusion model employed in Tango operates on a latent space derived from spectrograms, which can be represented as tensors with dimensions similar to those of images, we investigated whether the similarities in data representation would allow us to apply the image-based approach to audio processing.

Each section will provide a detailed explanation of the method and how it has been implemented to fit our task.

4.1 DDNM

The Denoising Diffusion Null-Space Model (DDNM) [3] is a novel framework that addresses various linear image restoration problems, such as image super-resolution, colorization, inpainting, compressed sensing, and deblurring, without requiring additional training or network modifications. DDNM leverages a pre-trained off-the-shelf diffusion model as the generative prior. By focusing on refining the null-space

contents during the reverse diffusion process, DDNM produces diverse and high-quality results that satisfy both data consistency and realism. To handle noisy restoration and enhance the quality of challenging tasks, the authors also propose DDNM$^+$, an enhanced and robust version of the framework.

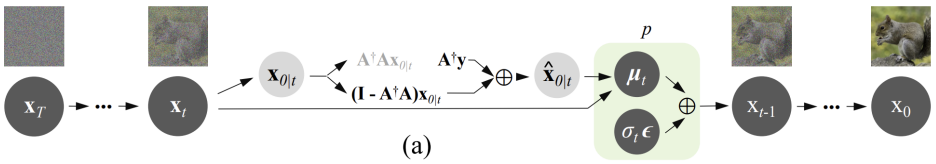


Figure 4.1. DDNM overview

The Range-Null space decomposition introduces a fresh viewpoint on the correlation between realness and data consistency. It reveals that data consistency is solely influenced by the contents of the range-space, which can be calculated analytically. As a result, the data term can be effectively ensured, shifting the focus to identifying suitable null-space contents that achieve the desired level of realness.

Consider an inpainting mask, denoted as A, and its pseudoinverse, denoted as $A^\dagger$, which satisfies $AA^\dagger A \equiv A$. A and $A^\dagger$ satisfy the following properties:

- $A^\dagger A$ can be interpreted as a projection operator that maps samples x to the range-space of A because $AA^\dagger Ax \equiv Ax$.
- $(I - A^\dagger A)$ can be interpreted as a projection operator that maps samples x to the null-space of A because $A(I - A^\dagger A)x \equiv 0$.

Considering the audio inpainting task as $y = Ax$, where x is the ground truth, A is the mask and y is the audio with a gap, we aim to reconstruct a $\hat{x}$ from the input y while satisfying two constraints:

$$Consistency : A\hat{x} \equiv y$$

$$Realness : \hat{x} \sim q(x)$$

where $q(x)$ is the distribution of the ground truth audio.

We can construct a solution $\hat{x}$ for an audio with a gap y that satisfies the consistency constraint as follows: $\hat{x} = A^\dagger y + (I - A^\dagger A)\bar{x}$ where $\bar{x}$ doesn't influence the result. The main objective is to find the appropriate $\bar{x}$ that fulfills the realness property.

Once we have computed the estimated x_0 at time-step t (denoted as $x_{0|t}$) using the standard diffusion model formula, we fix the range-space as $A^\dagger y$, while leaving the null-space unchanged. This results in the following expression:

$$\hat{x}_{0|t} = A^\dagger y + (I - A^\dagger A)x_{0|t}$$

Now we use $\hat{x}_{0|t}$ as an estimation for x_0 allowing only the null-space to be involved in the diffusion process. To compute x_{t-1} we sample from $p(x_{t-1}|x_t, \hat{x}_{0|t})$:

$$x_{t-1} = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{x}_{0|t} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \sigma_t\epsilon$$

The final result will satisfy the consistency criterion.

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, ..., 1$ **do**
3: $\quad \mathbf{x}_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \mathcal{Z}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\sqrt{1-\bar{\alpha}_t}\right)$
4: $\quad \hat{\mathbf{x}}_{0|t} = \mathbf{A}^\dagger\mathbf{y} + (\mathbf{I} - \mathbf{A}^\dagger\mathbf{A})\mathbf{x}_{0|t}$
5: $\quad \mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{\mathbf{x}}_{0|t})$
6: **return** $\mathbf{x}_0$

Figure 4.2. DDNM algorithm

4.2 DDNM$^+$

In the original paper, the authors introduce an enhanced version of the algorithm called DDNM$^+$. This enhanced version incorporates two major extensions to the original DDNM algorithm. The first extension is designed to handle noisy situations, but in our case it is not particularly relevant since any noise in the latent space representation of the spectrogram does not necessarily correspond to noise in the audio. Therefore, we did not focus on this extension in our experiments. The second extension, known as time-travel trick, is introduced to improve the realness of the inpainted audio and yield better overall results. We tested this extension and observed its impact on the quality of the generated audio.

When dealing with inpainting tasks involving large masks, the range-space contents $A^\dagger y$ is too local to guide the reverse diffusion process toward yielding a global harmony result. To address this limitation, the authors propose a technique known

as time-travel trick to generate a better "past", which in turn leads to a better "future". Specifically, after computing x_t, they "time travel" it back to x_{t+l} and use the future estimation $\hat{x}_{0|t}$, which should be more accurate than $\hat{x}_{0|t+l}$, to generate the next state x_{t+l-1}. This approach is inspired by RePaint [4], which will be discussed in more detail later in this chapter.

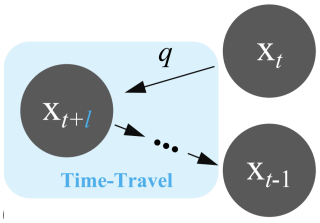


Figure 4.3. Time-travel trick

To incorporate the time-travel trick in the algorithm, for any time-step t we sample x_{t+l} from $q(x_{t+l}|x_t)$. Then we travel back to time-step $t+l$ and continue the normal sampling process of DDNM until we obtain x_{t-1}. The parameter l represents the travel length and directly affects the inference time in a linear manner. Setting l=0 corresponds to the classic DDNM algorithm without time travel.

4.3 RePaint

RePaint [4] is an image inpainting approach based on the Denoising Diffusion Probabilistic Model. It utilizes a pre-trained unconditional DDPM as the generative prior. Instead of modifying the original DDPM network, they condition the generation process by sampling the unmasked regions using the available image information during the reverse diffusion iterations. This technique allows the model to generate high-quality and diverse output images for various inpainting scenarios without modifying or conditioning the original DDPM network.

Inpainting approaches necessitate robust generative capabilities to ensure that the inpainted regions blend seamlessly with the surrounding data and maintain semantic coherence. To address the issue of semantically incorrect inpainting results with the standard DDPM sampling strategy, the authors propose an enhanced denoising strategy called RePaint. Their method involves moving both forward and backward in diffusion time. This unique approach allows for the generation of highly meaningful samples by effectively harmonizing the generated information with the available one throughout the entire inference process. As a result, the approach achieves more effective conditioning on the provided information, leading

to improved inpainting outcomes.

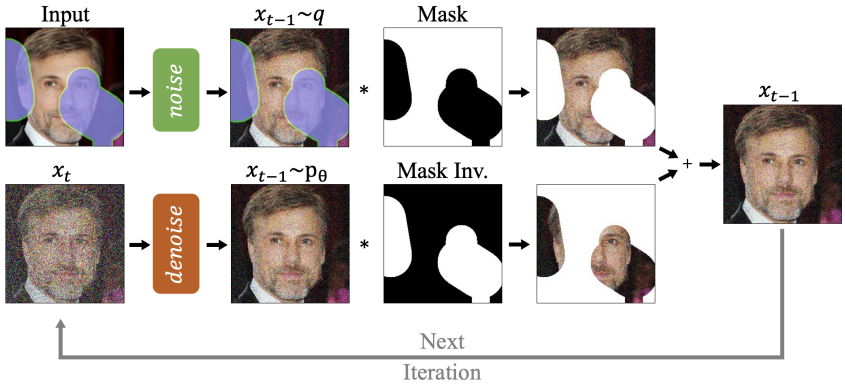


Figure 4.4. RePaint overview

Denote the ground truth as x, the unknown part as $m \odot x$ and the known part as $(1-m) \odot x$. In the reverse diffusion process, each step from x_t to x_{t-1} depends only on x_t. This means that we can modify the known regions $(1-m) \odot x_t$ while maintaining the correct properties of the corresponding distribution. Since the forward process is defined by a Markov Chain of added Gaussian noise, we can sample the intermediate x_t at any point in time. This allows us to sample the known regions $m \odot x_t$ at any time step t. Therefore, we achieve the following expression for one reverse step in this approach:

$$x_{t-1}^{known} \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$x_{t-1}^{unknown} \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$x_{t-1} = m \odot x_{t-1}^{known} + (1-m) \odot x_{t-1}^{unknown}$$

Thus, x_{t-1}^{known} is sampled using the known part of the given audio $m \odot x_0$, while $x_{t-1}^{unknown}$ is sampled from the model, given the previous iteration x_t. These are then combined to the new sample x_{t-1} using the mask.

Using the method just described, only the content type of the inpainted region matches the known regions. However, the inpainted region may not align semantically with the surrounding audio. In other words, although the DDPM takes into account the context of the known region, it does not effectively harmonize it with the remaining audio.

The model predicts x_{t-1} using x_t, which includes the output of the DDPM and the sample from the known region. However, the sampling of the known region

is performed independently of the generated parts of the audio, leading to disharmony. Although the model attempts to harmonize the audio in each step, it cannot fully converge because the same issue persists in the next step. Additionally, the maximum change to a sample decreases with each reverse step due to the variance schedule of β_t, limiting the model's ability to correct mistakes and address disharmonious boundaries in subsequent steps. Consequently, the model requires more time to effectively harmonize the conditional information x_{t-1}^{known} with the generated information $x_{t-1}^{unknown}$ within a single step before progressing to the next denoising step.

In the resampling approach, DDPM's ability to generate consistent structures within a data distribution is utilized to harmonize the input. To achieve this, the output x_{t-1} is diffused back to x_t by sampling $x_t \sim \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$. Despite scaling down the output and introducing noise, certain information from the generated region $x_{t-1}^{unknown}$ is still retained in $x_t^{unknown}$. This results in a new $x_t^{unknown}$ that is both more harmonized with x_t^{known} and incorporates conditional information from it.

To address the limitation of harmonizing only one step at a time, the authors introduce a parameter called "jump length" to extend the operation's time horizon (in the previous case it was set to 1). Increasing the jump length, more steps can be harmonized, allowing for the incorporation of semantic information throughout the entire denoising process. However, it's important to note that increasing the jump length also increases the runtime of the reverse diffusion. Empirical demonstrations showed that the benefits of resampling saturate at around 10 resampling steps.

$x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  for $u = 1, \ldots, U$ **do**
    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\epsilon = \mathbf{0}$
    $x_{t-1}^{\text{known}} = \sqrt{\bar{\alpha}_t} x_0 + (1 - \bar{\alpha}_t)\epsilon$
    $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
    $x_{t-1}^{\text{unknown}} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(x_t, t) \right) + \sigma_t z$
    $x_{t-1} = m \odot x_{t-1}^{\text{known}} + (1 - m) \odot x_{t-1}^{\text{unknown}}$
    if $u < U$ and $t > 1$ **then**
      $x_t \sim \mathcal{N}(\sqrt{1-\beta_{t-1}} x_{t-1}, \beta_{t-1}\mathbf{I})$
    end if
  end for
end for
return x_0

Figure 4.5. RePaint algorithm

4.4 RePaint$^+$

In addition to the resampling technique described before, we also incorporate jumps in diffusion time. In the figure we provide a pseudo-code representation to illustrate the generation of state transitions. It is important to note that each transition either increases or decreases the diffusion time t by one. For instance, when a jump length of j = 10 is chosen, we apply ten forward transitions followed by ten reverse transitions.

```
t_T = 250
jump_len = 10
jump_n_sample = 10

jumps = {}
for j in range(0, t_T - jump_len, jump_len):
    jumps[j] = jump_n_sample - 1

t = t_T
ts = []

while t >= 1:
    t = t-1
    ts.append(t)

    if jumps.get(t, 0) > 0:
        jumps[t] = jumps[t] - 1
        for _ in range(jump_len):
            t = t + 1
            ts.append(t)

ts.append(-1)
```

Figure 4.6. Pseudo code to generate a diffusion time schedule for RePaint$^+$

Chapter 5

Evaluation

In this chapter, we will provide a detailed explanation of our evaluation process for audio inpainting. We will discuss the selection criteria for the audio clips used in the evaluation and provide insights into the inference details, including the parameters employed and the inference times for each method. Additionally, we will explain the metrics chosen to assess the performance of the inpainting techniques. Finally, we will present the results of our evaluation. To facilitate access to the original audios, inpainted results, spectrograms, and metrics, we have made them available for listening and analysis at the following URL: `http://www.andrearodriguez.it/inpainting/`.

5.1 Audio samples

During the evaluation phase, we utilized 24 audio samples sourced from the test set of the AudioCaps dataset [8]. The AudioCaps dataset consists of links to YouTube videos, along with specified start times for 10-second audio clips. Each clip is accompanied by five captions, although we selected only one caption for our purposes. The chosen audio samples were representative of three distinct audio types: speech clips, audios exhibiting clear patterns in the spectrogram, and a mixture of different sounds.

The speech clips are associated with the following captions:

- A man speaks as birds chirp and dogs bark
- Female and male are having conversation
- Male speech with people speaking in the background
- A man is speaking with rumbling in the background
- An adult female is speaking in a quiet environment

- A man speaking as a crowd of people laugh and applaud
- Speech and then a pop and laughter
- A baby is crying, and a woman speaks

The audios exhibiting clear patterns in the spectrogram are associated with the following captions:

- An emergency siren wailing as a large truck engine accelerates growing louder
- A crowd murmurs as a siren blares and then stops at a distance
- An emergency vehicle has the siren on
- An emergency vehicles' siren with a brief male yell
- A man talking then whistling as guitar music plays
- A person whistles a tune with wind noise and people talking in the background
- A whistling noise with wind blowing in the background
- A telephone ringing

The last batch is composed by a mixture of different sounds associated with the following captions:

- Thunder and a gentle rain
- Duck quacking repeatedly
- An engine running and helicopter propellers spinning
- A whooshing noise followed by an explosion
- A person talking which later imitates a couple of meow sounds
- A woman talking followed by a young girl talking while an infant cries
- Instrumental music playing followed by heavy fabric being rustled then a man whistling
- A steady ringing with the tick took of a clock

The audio clips were obtained from YouTube and segmented based on the specified start and end times mentioned in the provided dataset's CSV file.

5.2 Inference details

In our evaluation, we examined two baseline methods described in Chapter 3 and four proposed methods outlined in Chapter 4. We conducted the evaluation by introducing gaps of 1 second and 2 seconds in the selected audio clips. These gaps were positioned in the middle of the clips, specifically from 4.5 seconds to 5.5 seconds and from 4 seconds to 6 seconds, respectively. Once the gapped audio was generated from the original version, it was provided as input to the models to generate the inpainted audio together with the corresponding caption.

We applied the four proposed inpainting methods (DDNM, DDNM$^+$, RePaint, RePaint$^+$) on the generative diffusion model Tango, specifically on the pretrained model checkpoint Tango-Full-FT-Audiocaps. Tango was chosen over AudioLDM due to its superior performance during generation and the potential to achieve better results in the inpainting task. For each of the four methods, the inference process was conducted using DDPM [5] with 1000 steps.

During the inference process we utilized the following parameters:

- For the AudioLDM method, we employed the pretrained model checkpoint audioldm-s-full-v2. As for the inference process, we utilized DDIM [14] with 200 steps;
- For the Tango method, we employed the pretrained model checkpoint Tango-Full-FT-Audiocaps. As for the inference process, we utilized DDPM [5] with 1000 steps;
- DDNM did not require any specific parameters to be set;
- For DDNM$^+$, we utilized an estimated noise level (σ_y) set to 0 and a travel length of 10;
- For the RePaint method, we employed 10 resampling steps;
- For the RePaint$^+$ method, we employed a jump length set to 10 and performed 10 resampling steps.

We conducted the model testing using Google Colab with one NVIDIA Tesla T4 GPU and Kaggle with one NVIDIA Tesla P100 GPU. The audio clips were divided into three batches, with each batch containing eight samples. The inference times for the specified parameters, with 10-second clips sampled at 16000Hz, are detailed in Table 5.1.

Table 5.1. Inference times in minutes using one NVIDIA Tesla T4.

	1 Clip	Batch of 8 Clips
AudioLDM	1	4
Tango	10	40
Tango + DDNM	10	40
Tango + DDNM$^+$	120	480
Tango + RePaint	100	400
Tango + RePaint$^+$	100	400

It is important to consider that when comparing the metrics computed on the results, certain methods may require significantly more time (10x or 100x) to generate the inpainted audio clips compared to others. While a method may achieve better results, the trade-off with longer inference times may not be favorable in certain scenarios in which low latency is crucial. The choice of the method depends on the specific context in which audio inpainting is required, and a balance must be struck between the desired quality of the inpainted audio and the associated inference time.

5.3 Chosen metrics

To assess the results obtained from the two baselines and the four proposed methods, we employed four distinct metrics to evaluate the quality of the inpainted audio signal in comparison to the original signal. Drawing from the evaluation practices conducted in relevant literature for similar tasks, we opted to compute two metrics directly on the audio signals and two metrics on the spectrograms generated from these audio signals. This approach allowed us to comprehensively evaluate various aspects of the inpainted audio quality and draw meaningful comparisons between the different methods.

Before computing each metric, we ensured that the inpainted signal, both in terms of audio and spectrogram, was scaled to the same range as the original signal. This step was taken to facilitate a fair and unbiased comparison between the two, allowing for meaningful evaluations of the inpainted audio quality across the different metrics.

SDR on audio The Signal-to-Distortion Ratio [20] [21] quantifies the ratio between the desired audio signal (signal) and the distortion introduced during the inpainting process (distortion). The SDR is computed by comparing the original audio signal with the inpainted audio signal and analyzing the differences between them. The

distortion refers to any deviation or error introduced during the inpainting process, which may include artifacts, noise, or inaccuracies in the reconstructed audio. A higher SDR value indicates a better preservation of the desired audio signal and a lower amount of distortion in the inpainted audio. It suggests that the inpainting technique successfully maintains the original content and minimizes any unwanted changes or errors.

SNR on audio The Signal-to-Noise Ratio [22] is a metric used to evaluate the quality of the inpainted audio signal by comparing the desired signal (signal) to the background noise introduced during the inpainting process (noise). The SNR is calculated by considering the ratio between the power of the desired audio signal and the power of the background noise. The desired signal represents the original audio content that should be preserved during the inpainting, while the noise refers to any additional unwanted sound or artifacts introduced during the inpainting process. A higher SNR value indicates a better preservation of the desired audio signal relative to the background noise. It suggests that the inpainting technique effectively reduces the amount of noise or unwanted artifacts in the inpainted audio, resulting in a cleaner and more accurate reconstruction.

PSNR on spectrograms The Peak Signal-to-Noise Ratio can be computed to evaluate the quality of the inpainted spectrogram compared to the original spectrogram. It provides a measure of the similarity between the two spectrograms by quantifying the ratio between the maximum possible power of the signal and the power of the distortion introduced during the inpainting process. A higher PSNR value indicates a higher similarity or preservation of the original spectrogram in the inpainted spectrogram. It suggests that the inpainting technique has successfully reconstructed the spectrogram with minimal distortion or error.

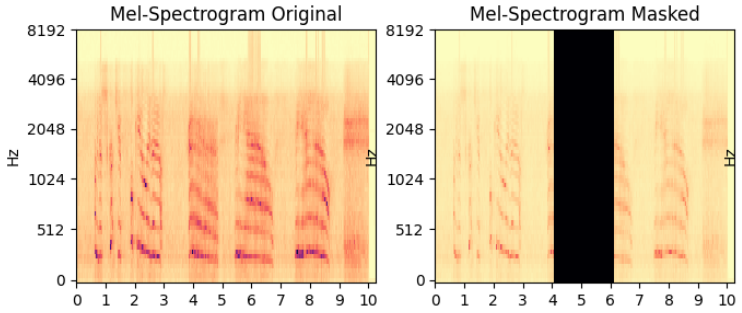
SSIM on spectrograms The Structural Similarity Index Measure is used to evaluate the similarity between two signals. In the context of audio inpainting, SSIM can be computed on spectrograms to assess the structural similarity between the original spectrogram and the inpainted spectrogram. SSIM takes into account three main components: luminance, contrast, and structure. When applied to spectrograms, SSIM evaluates the similarity of the frequency and time structures present in the original and inpainted audio signals. The resulting SSIM value ranges between 0 and 1, where 1 indicates perfect structural similarity and 0 indicates no structural similarity. Higher SSIM values indicate a higher degree of similarity between the original and inpainted spectrograms, suggesting that the inpainting technique successfully preserves the structural information of the audio signal.

5.4 Results

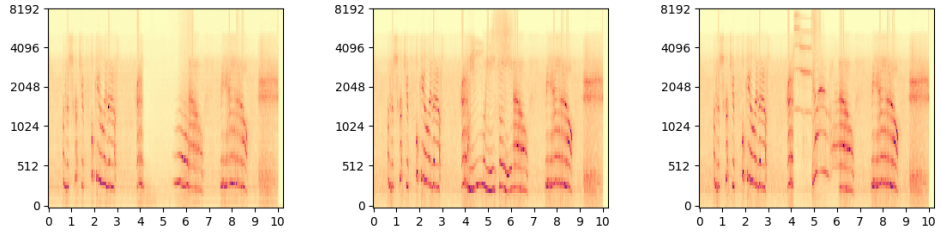
In this section, we present the results obtained using the different approaches and the previously mentioned metrics computed on the inpainted audio clips. In order to compare the results of each method, we will examine three selected clips. For the purpose of this comparison, we will display the spectrograms of the clips, while the corresponding audio can be accessed and listened to via the link provided before.

5.4.1 Clip 1

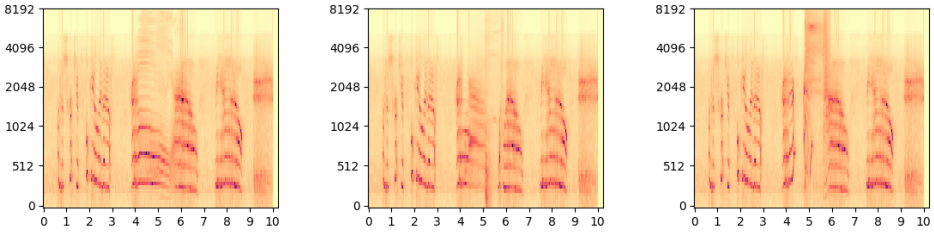
Let's begin by examining the first clip, which is described as "A person talking which later imitates a couple of meow sounds". In this clip, a 2-second segment from 4s to 6s was masked. Figure 5.1 showcases the outcomes of the inpainting process for this particular clip.



(a) Original and Masked



(b) AudioLDM **(c)** Tango **(d)** DDNM



(e) DDNM$^+$ **(f)** RePaint **(g)** RePaint$^+$

Figure 5.1. Results for audio clip with caption: "A person talking which later imitates a couple of meow sounds". Gap of 2 seconds from 4s to 6s.

Upon examining the spectrograms, it is evident that the two baselines and the four proposed methods exhibit significant differences in their performance. It is important to note that the usage of Gaussian noise during the inference process can also contribute to variations in the outcomes. A preliminary analysis reveals that RePaint appears to be the most effective method for this specific audio clip, as it successfully preserves both the signals before and after the gap, including the period of silence in between.

However, when considering the metrics presented in table 5.2, a different pattern emerges. The metrics computed directly on the audio signal favor the DDNM method, while those computed on the spectrograms favor RePaint (PSNR) and AudioLDM (SSIM), although SSIM values are nearly identical for RePaint and AudioLDM. This observation highlights that spectrograms may not always capture the full complexity of the audio, leading to differing results when compared to metrics computed on the actual audio waveform.

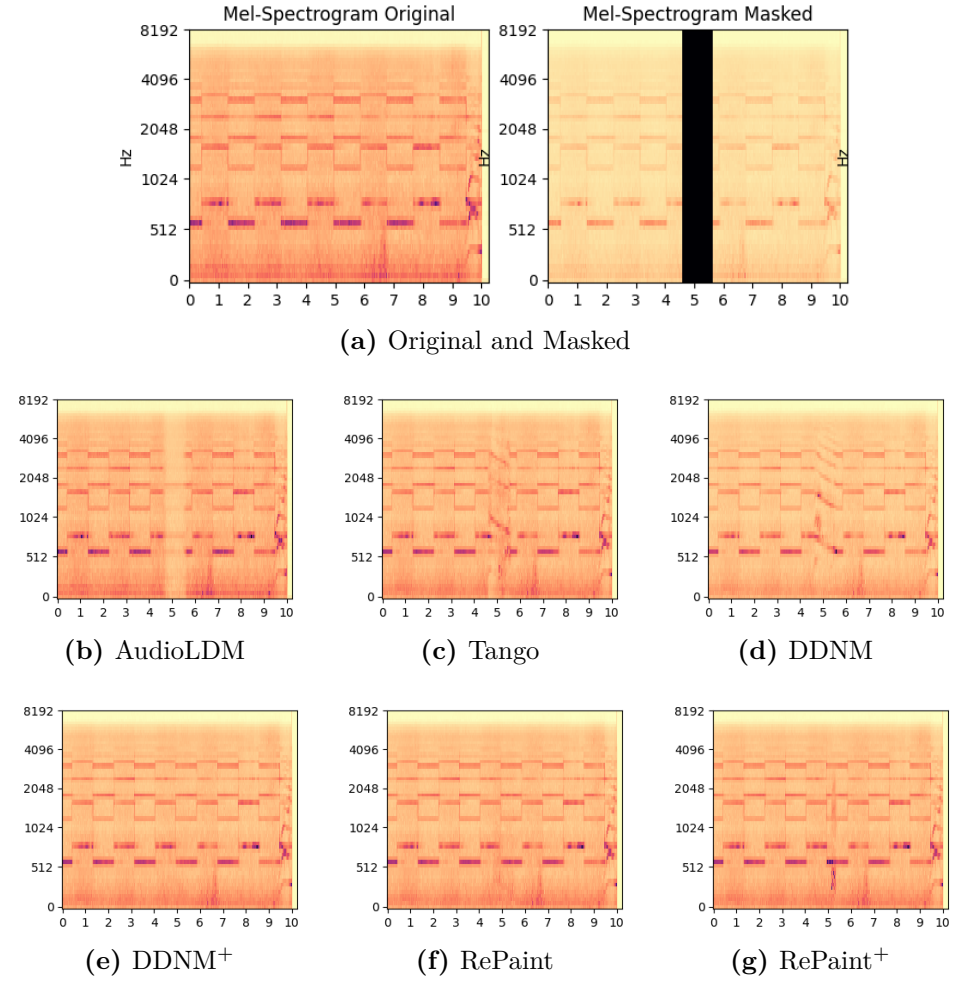
Table 5.2. Metrics for audio clip with caption: "A person talking which later imitates a couple of meow sounds". Gap of 2 seconds from 4s to 6s.

	AudioLDM	**Tango**	**Tango DDNM**	**Tango DDNM$^+$**	**Tango RePaint**	**Tango RePaint$^+$**
SDR	-2.10	0.81	**3.32**	0.48	1.49	0.92
SNR	1.31	2.52	**4.52**	1.51	2.60	1.82
PSNR	34.43	40.71	41.76	40.95	**42.30**	40.69
SSIM	**98.86**	98.52	98.49	98.43	98.84	98.35

5.4.2 Clip 2

Now let's shift our focus to the second audio clip, which is described as "An emergency vehicles' siren with a brief male yell". In this clip, a 1-second segment from 4.5s to 5.5s was masked. Figure 5.2 showcases the outcomes of the inpainting process for this particular clip.

In this instance, we observe that the three simpler approaches (AudioLDM, Tango, and DDNM) were unable to accurately reconstruct the original audio waveform. It appears that these methods attempted to inpaint the waveform of a siren without considering how it would blend with the surrounding portions of the original audio. Conversely, DDNM$^+$ and RePaint exhibited more favorable outcomes. These



(a) Original and Masked

(b) AudioLDM **(c)** Tango **(d)** DDNM

(e) DDNM$^+$ **(f)** RePaint **(g)** RePaint$^+$

Figure 5.2. Results for audio clip with caption: "An emergency vehicles' siren with a brief male yell". Gap of 1 second from 4.5s to 5.5s.

methods achieved better results by successfully reconstructing the desired waveform, while ensuring a smoother transition between the inpainted segment and the surrounding audio. It is worth noting that RePaint$^+$ introduced some unintended noise artifacts into the reconstructed waveform, which could potentially affect the overall listening experience.

The metrics provided in table 5.3 validate the previous observations. DDNM$^+$ exhibits higher values for SDR and SNR metrics, indicating superior performance in terms of signal distortion and noise reduction. On the other hand, RePaint achieves higher values for PSNR and SSIM metrics, indicating better preservation of signal quality and structural similarity. It is noteworthy that both DDNM$^+$ and RePaint demonstrate comparable metric values, while the other approaches display significantly lower metrics.

Table 5.3. Metrics for audio clip with caption: "An emergency vehicles' siren with a brief male yell". Gap of 1 second from 4.5s to 5.5s.

	AudioLDM	Tango	Tango DDNM	Tango DDNM$^+$	Tango RePaint	Tango RePaint$^+$
SDR	-3.74	6.55	6.77	**9.36**	8.47	6.27
SNR	-1.00	7.16	7.35	**9.64**	8.82	6.99
PSNR	36.43	47.52	48.69	53.82	**54.33**	48.37
SSIM	98.86	99.05	99.12	99.71	**99.72**	99.48

5.4.3 Clip 3

Finally, let's redirect our attention to the third audio clip, which is described as "A whistling noise with wind blowing in the background". In this clip, a 1-second segment from 4.5s to 5.5s was masked. Unlike the previous two audio clips, we want to draw attention to a notable behavior observed during the experiments. Specifically, consider the inpainted audio generated using the DDNM method, as depicted in Figure 5.3. While it may not precisely match the original waveform upon comparison, it reconstructs a plausible waveform for the masked section. In certain use cases, this behavior could be deemed satisfactory and considered a successful inpainted clip.

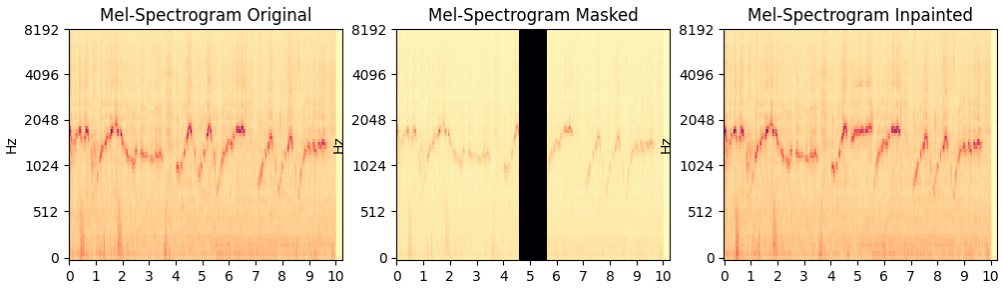


Figure 5.3. Results for audio clip with caption: "A whistling noise with wind blowing in the background". Gap of 1 second from 4.5s to 5.5s. Method used: DDNM

5.4.4 Average over all 24 clips

After computing the metrics for each audio clip, we calculated the average value of each metric for each model. These values provide a summary of the overall performance of each model. The averaged metrics are reported in table 5.4. It is important to note that when reporting SSIM values in this work, they were multiplied by 100 to provide more accurate and easily interpretable values.

Table 5.4. Metrics averages. Higher values indicate better performance.

4.5-5.5	AudioLDM	Tango	Tango DDNM	Tango DDNM$^+$	Tango RePaint	Tango RePaint$^+$
SDR	-3.27	5.48	5.03	5.47	**5.96**	4.97
SNR	-0.25	5.73	5.39	5.71	**6.17**	5.28
PSNR	39.46	43.35	42.44	43.22	**44.08**	42.38
SSIM	98.40	99.25	99.21	**99.28**	99.18	99.14

4-6	AudioLDM	Tango	Tango DDNM	Tango DDNM$^+$	Tango RePaint	Tango RePaint$^+$
SDR	-4.97	1.48	1.53	**1.99**	1.64	1.48
SNR	-1.45	2.82	**2.90**	2.21	2.71	2.02
PSNR	35.44	39.74	39.85	38.61	**39.92**	38.56
SSIM	97.84	98.34	98.46	98.45	**98.59**	98.56

These results provide valuable insights into the performance of the various methods. It is evident that the two baselines fall short in comparison to the other approaches across all metrics, regardless of the gap duration. RePaint consistently achieves superior results compared to the other methodologies. However, it is crucial to consider the trade-off with inference time. RePaint requires approximately 10 times longer than Tango and DDNM, and a 100-fold increase over AudioLDM. Therefore, when selecting an appropriate method, both the performance metrics and the associated inference time should be taken into account.

Chapter 6

A more complex use case: Audio inpainting in communication scenarios

The methods explored so far have demonstrated their effectiveness in audio inpainting. However, we also considered an additional scenario: audio transmission through a communication channel where noise can corrupt the signal, resulting in a degraded audio for the receiver. Modern techniques address this issue by transmitting additional data to accurately reconstruct the missing or noisy parts of the audio. However, in certain cases, it may not be necessary to have a complete and exact replica of the original transmitted audio. Preserving the semantic information could suffice, even if some of the received message bits are corrupted due to adverse channel conditions.

In our approach, we aim at predicting the pre-noise audio by leveraging the generative diffusion models mentioned earlier and employing an adapted version of DDNM [3]. This modified approach incorporates denoising capabilities and can be valuable also when opting to transmit less data and reconstruct the missing segments at the destination.

6.1 Scenario

In the context of communication systems, every method must confront the physical obstacles presented by real-world environments. In our scenario, we consider a situation where the signal z traverses a noisy channel. We model the noise as additive white Gaussian noise, where the noise component ϵ follows a Gaussian distribution

given by $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, with σ^2 representing the noise variance. Consequently, the received signal can be expressed as $\tilde{z} = z + \epsilon$. To measure the noise level added to the signal, we employ the Peak Signal-to-Noise Ratio (PSNR), which is defined as:

$$\text{PSNR} = 10 \log \frac{P}{\sigma^2} \text{ (dB)}$$

During the testing phase of our method, we explored two different scenarios by varying the Peak Signal-to-Noise Ratio (PSNR) values. We considered a PSNR of 30, where the noise only had a slight impact on the signal, and a PSNR of 20, where the noise significantly affected the signal quality. These choices allowed us to evaluate the performance of the method under varying levels of noise interference. We also attempted to reconstruct audio with added noise with PSNR of 10. However, in this case, the noise completely overwhelmed the signal and the denoising phase was unable to successfully reconstruct the original data.

We investigated a scenario where the latent representation of the audio is transmitted through the noisy channel. Consequently, the noise becomes incorporated into the latent representation and needs to be removed from it. To introduce noise, we followed a three-step process. First, we normalized the input data. Then, we added the generated noise based on the desired PSNR. Finally, we restored the data to its original range. The code snippet used to add the noise is provided in listing 6.1.

Listing 6.1. Code to add noise to the signal

```
min = torch.min(original_latents)
max = torch.max(original_latents)

original_latents = (original_latents - min) / (max - min)

# SNR_DICT is a dictionary containing the values to multiply
# the generated noise to reach the desired PSNR.
noise = torch.randn(original_latents.shape) * SNR_DICT[snr]

noisy_latents = original_latents + noise
noisy_latents = (noisy_latents * (max - min)) + min
```

6.2 Method

Let's now explore the methodology behind our approach. The core remains largely consistent with the description provided in section 4.1, with a few notable modifications as suggested in the original paper [3]. Specifically, they introduce a new hyperparameter called σ_y, which is set prior to utilizing the model for denoising purposes. The purpose of this parameter is to represent the estimated noise level that we aim to eliminate from the audio signal.

Let us define the following variables:

$$a_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}$$

$$\Sigma_t = \lambda_t \mathbf{I} \qquad \text{where} \qquad \lambda_t = \begin{cases} 1, & \sigma_t \geq a_t \sigma_y \\ \sigma_t / a_t \sigma_y, & \sigma_t < a_t \sigma_y \end{cases}$$

$$\Phi_t = \gamma_t \mathbf{I} \qquad \text{where} \qquad \gamma_t = \sigma_t^2 - (a_t \lambda_t \sigma_y)^2$$

The key differences from the classic DDNM algorithm appear when computing $\hat{x}_{0|t}$ and x_{t-1} which become:

$$\hat{x}_{0|t} = \Sigma_t A^\dagger y + (I - \Sigma_t A^\dagger A) x_{0|t}$$

$$x_{t-1} = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{x}_{0|t} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \sqrt{\Phi_t}\epsilon$$

Both the remaining components of the method and the inference process remain unchanged.

We also explored the combination of the denoising and inpainting capabilities of the model by cascading the two versions of DDNM. The transmitted audio's latent space representation, which contains both added noise and a gap in the middle (as described in section 5.2), undergoes denoising using the modified version of DDNM discussed earlier. The denoised result then undergoes inpainting using the original version of DDNM described in section 4.1. This cascading approach ensures that the inpainting phase can leverage the benefits of the denoised audio, allowing for a more accurate and context-aware inpainting process.

We conducted empirical experiments to determine the optimal levels of σ_y that yield improved results for each level of noise added to the signal. The findings are in table 6.1. We observed that when using a too low σ_y, the noise is not completely removed, resulting in residual artifacts. On the other hand, when using a too high σ_y, the output becomes overly smooth, leading to a loss of the original audio characteristics.

Table 6.1. Levels of σ_y corresponding to different amounts of added noise.

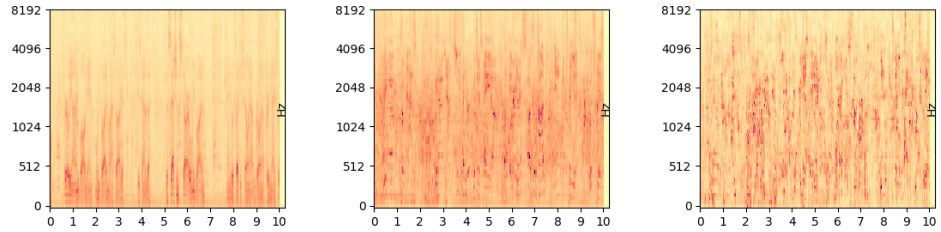
PSNR	σ_y
30	15.0
20	35.0
10	100.0

6.3 Results

We evaluated the method described above by testing it on four different audio clips containing speech. Each test was configured as follows:

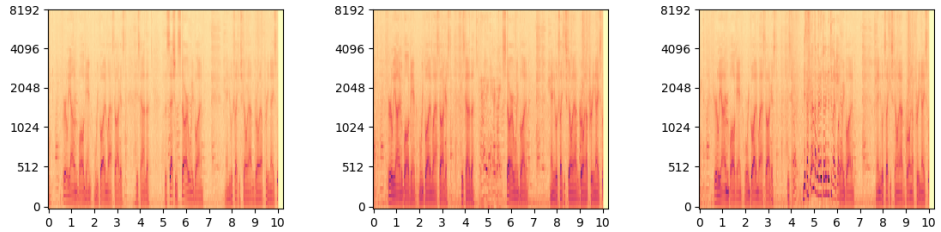
- PSNR = 30 and no gap
- PSNR = 20 and no gap
- PSNR = 30 and gap from 4.5s to 5.5s
- PSNR = 20 and gap from 4.5s to 5.5s
- PSNR = 30 and gap from 4s to 6s
- PSNR = 20 and gap from 4s to 6s

We will present the results for one of the four audio clips used in our testing, specifically the one captioned “Male speech with people speaking in the background”. In figure 6.1, we display the spectrograms of the original clip along with the ones containing added noise (with PSNR values of 30 and 20). Figures 6.2 and 6.3 showcase the denoising results for the PSNR 30 and 20 cases respectively, as well as the combined denoising and inpainting outcomes. It is evident that the method successfully reconstructs the original signal from both noisy inputs. To listen to the audio clips visit the following URL: `http://www.andrearodriguez.it/inpainting/`.



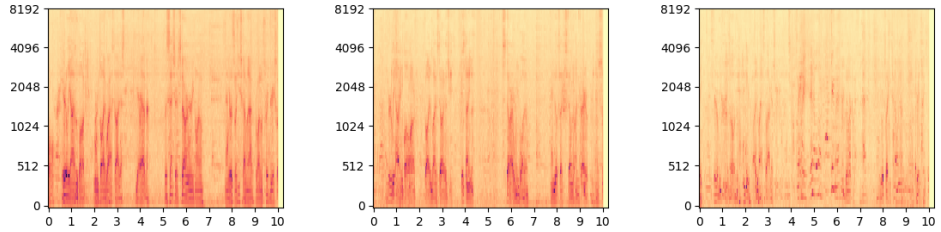
(a) Original **(b)** Noisy PSNR 30 **(c)** Noisy PSNR 20

Figure 6.1. Original audio spectrogram, noisy audio spectrogram with PSNR 30 and 20 for audio clip with caption: "Male speech with people speaking in the background".



(a) Denoise **(b)** Denoise + Inpaint 1s **(c)** Denoise + Inpaint 2s

Figure 6.2. Denoised from PSNR 30 and inpainted results for audio clip with caption: "Male speech with people speaking in the background".



(a) Denoise **(b)** Denoise + Inpaint 1s **(c)** Denoise + Inpaint 2s

Figure 6.3. Denoised from PSNR 20 and inpainted results for audio clip with caption: "Male speech with people speaking in the background".

The validity of the results is further confirmed by evaluating the Signal-to-Noise Ratio (SNR) both before and after the denoising process. Comparing the SNR values between the original audio clip and its corresponding noisy version, as well as between the original audio clip and the respective denoised version, we observe an increase in SNR for all four tested clips, regardless of whether the PSNR value is 30 or 20. These findings are presented in Table 6.2 and 6.3 respectively.

Table 6.2. SNR values computed between the original clip and the noisy audio (PSNR 30) and SNR values computed between the original clip and the denoised version.

PSNR 30	Clip 1	Clip 2	Clip 3	Clip 4
Noisy	-3.53	-2.45	-3.54	-3.61
Denoised	-1.82	-1.14	-1.32	-2.20

Table 6.3. SNR values computed between the original clip and the noisy audio (PSNR 20) and SNR values computed between the original clip and the denoised version.

PSNR 20	Clip 1	Clip 2	Clip 3	Clip 4
Noisy	-9.80	-9.11	-8.61	-10.10
Denoised	-2.47	-3.80	-2.64	-2.58

Chapter 7

Conclusions and Future works

Finally, drawing upon the findings and insights gathered throughout this thesis, several key conclusions can be derived. The proposed techniques have demonstrated significant improvements in comparison to the two baselines. Among the selected methods, RePaint consistently outperforms the other methodologies, showcasing its effectiveness in audio inpainting tasks. It is important, however, to carefully consider the trade-off between performance and inference time. While RePaint delivers superior results, it also requires a significantly longer inference time compared to DDNM. Thus, when selecting an appropriate method for audio inpainting, both the performance metrics and the associated inference time should be carefully weighed. By bridging the gap between generative diffusion models and audio inpainting, this research contributed to the growing field of audio restoration.

Furthermore, this research has unveiled two possibilities for future exploration. Firstly, the development of an automated communication system capable of identifying problematic segments after denoising and subsequently applying inpainting techniques. By integrating denoising and inpainting capabilities within a communication system, it would be possible to improve the overall audio quality by automatically addressing identified issues. Secondly, the exploration of inpainting techniques without relying on a text description, instead utilizing only the known portion of the audio as conditioning. By leveraging the available information and employing advanced inpainting methodologies, it may be possible to reconstruct missing or corrupted audio segments accurately and efficiently. This approach opens up new possibilities for inpainting in scenarios where a complete text description is not available or necessary.

Bibliography

[1] Haohe Liu et al. *AudioLDM: Text-to-Audio Generation with Latent Diffusion Models*. 2023. arXiv: 2301.12503 [cs.SD].

[2] Deepanway Ghosal et al. *Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model*. 2023. arXiv: 2304.13731 [eess.AS].

[3] Yinhuai Wang, Jiwen Yu, and Jian Zhang. *Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model*. 2022. arXiv: 2212.00490 [cs.CV].

[4] Andreas Lugmayr et al. *RePaint: Inpainting using Denoising Diffusion Probabilistic Models*. 2022. arXiv: 2201.09865 [cs.CV].

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[7] Benjamin Elizalde et al. *CLAP: Learning Audio Concepts From Natural Language Supervision*. 2022. arXiv: 2206.04769 [cs.SD].

[8] Chris Dongjoo Kim et al. "AudioCaps: Generating Captions for Audios in The Wild". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 119–132. DOI: 10.18653/v1/N19-1011. URL: https://aclanthology.org/N19-1011.

[9] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].

[10] Ke Chen et al. *HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection*. 2022. arXiv: 2202.00874 [cs.SD].

[11] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[12] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[13] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG].

[14] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG].

[15] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. *HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis*. 2020. arXiv: 2010.05646 [cs.SD].

[16] Jort F. Gemmeke et al. "Audio Set: An ontology and human-labeled dataset for audio events". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 776–780. DOI: 10.1109/ICASSP.2017.7952261.

[17] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 2022. arXiv: 2210.11416 [cs.LG].

[18] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL].

[19] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. *Learning from Between-class Examples for Deep Sound Recognition*. 2018. arXiv: 1711.10282 [cs.LG].

[20] Robin Scheibler. *SDR – Medium Rare with Fast Computations*. 2021. arXiv: 2110.06440 [eess.AS].

[21] E. Vincent, R. Gribonval, and C. Fevotte. "Performance measurement in blind audio source separation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.4 (2006), pp. 1462–1469. DOI: 10.1109/TSA.2005.858005.

[22] Jonathan Le Roux et al. *SDR - half-baked or well done?* 2018. arXiv: 1811.02508 [cs.SD].

[23] Lilian Weng. "What are diffusion models?" In: *lilianweng.github.io* (July 2021). URL: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.