Imperial College London

Department of Computing

Bootstrapping Stable Diffusion for Audio Synthesis

Final Report

Author:
Ryan Perkins

Supervisors:
Professor Björn Schuller,
Rodrigo Schönburg Carrillo de Mira

Submitted in partial fulfillment of the requirements for the MSc degree in Computing of Imperial College London

September 2023

Abstract

Stable Diffusion (SD) is a popular open-source model pre-trained on billions of image-text pairs, designed to generate images from textual prompts. With its extensive ecosystem and supportive community, SD offers functionalities like inpainting and outpainting along with a multitude of other features enabling enhanced control over generation. Notably, SD can be fine-tuned on consumer-level hardware, enabling users to adapt the model to their specific needs. While diffusion models have been foundational in text-to-audio (TTA) systems, training them from scratch remains prohibitively expensive for many consumers. Moreover, these models may lack the functionality, support and tools that the SD ecosystem provides.

In *Bootstrapping Stable Diffusion for Audio Synthesis*, we pioneer the exploration of fine-tuning SD to generate non-music audio spectrograms. A primary aim is to democratise consumer access to fine-tuning their own text-to-audio systems and to harness the capabilities of the SD ecosystem for audio generation. Through extensive empirical experiments, we find that while SD showed proficiency in generating certain audio classes, it faced challenges in others, suggesting a potential need for a more comprehensive training dataset. We also highlight the versatility of the SD ecosystem for both researchers and users.

Although our model did not surpass the current state-of-the-art, the insights emphasise the potential of SD in audio generation, its prospective role as a consumer-accessible TTA system, and avenues for further research. We conclude by addressing ethical considerations, including potential misuse and copyright concerns, emphasising the importance of responsible advancement in this domain. Audio samples illustrating our model's capabilities can be accessed here.

Acknowledgements

I made the decision to leave my comfortable career as an economist to retrain into a field I was passionate about - a risk I could only take with the support of my friends and family, and a risk which was truly paid off by the brilliant minds I met throughout. For what it's worth, I would like to dedicate this space to acknowledging those who have helped make this year not only a transformative one, but a fulfilling one.

I would like to firstly thank Professor Björn Schuller for his role as my project supervisor, as well as providing me the opportunity to complete a dissertation within the field of Deep Learning. I also would like to thank Dr. Rodrigo Mira, who's invaluable guidance was paramount for not only the completion of this project, but also for equipping me with the tools and experience I will no doubt need for my future career. I would also like to thank Dr. Josiah Wang, not only for his valuable input but also as an integral figure in ensuring the high quality of the course overall.

I would also like to thank my partner Klaudija, who not only inspired me to make this move, but also supported me utmost throughout it. I would also like to thank the wonderful people I met from the department, especially Nihir and Alex, with whom our many late-night conversations about AI and ML was vital in expanding my curiosity and confidence with these topics. I would like to thank my friend Kane, who by leading by example, planted the seeds that this change could also be a possibility for me. Finally, I would like to thank my parents, who's limitless support has always encouraged me to strive for greater heights.

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

Generative AI is used - with increasing prevalence - to generate high-quality data points in language, image and audio domains. Latent diffusion models (LDMs) [1] - a class of generative model - (and the Stable Diffusion (SD) LDM specifically) have enabled consumer-level accessibility to state-of-the-art (SOTA) image generation capability with relatively accessible compute requirements. Alongside enabling accessible generation of high-quality images, SD also offers the capability to be fine-tuned to generate images of novel concepts.

Text-to-speech (TTS) and Text-to-audio (TTA) - or speech/audio synthesis - refers to using text-conditioning (i.e. prompts) to generate realistic samples of speech (e.g. speaking the words given by the prompt with a specific denoted style) or audio effects (e.g. natural/urban sounds such as bird song, or traffic). TTS and TTA have evolved with continual advent of new technologies and models - from early synthesis techniques [2, 3, 4], to initial networks (such as CNNs [5] and RNNs [6, 7, 8, 9]), and eventually to more recent advances such as Generative Adversarial Networks (GANs) [10, 11, 12, 13] and Transformer-based architectures [14, 15, 16]. Most recently, TTS/TTA models have found success using diffusion-based architectures to generate audio with promising results (SOTA in some cases) [17, 18, 19, 20, 21, 22].

Riffusion, a recent software release [23], has successfully fine-tuned SD to generate spectrograms of music, with positive and high-quality qualitative results. Inspired by Riffusion, this research aims to fine-tune an off-the-shelf SD model to evaluate its feasibility as a TTA system. Successful synthesis of audio in this manner would greatly increase the accessibility of high-quality audio synthesis - enabling easier implementation and adoption of TTA systems in future applications and use-cases. Successful research outcomes will involve the creation of a fine-tuned SD model capable of generating spectrograms of recognisable audio.

SD is a popular generative model, with a wealth of tools, software and tutorials created by its community. For example, Hugging Face - a platform for machine learning providing access to code and model checkpoints - maintain a package to support implementation and fine-tuning of popular diffusion models (including SD) named 'Diffusers' [24]. Diffusers provides easy to download SD model checkpoints, as well as training code for multiple different fine-tuning approaches - such as Low-Rank Adaptation (LoRA) [25] and Dreambooth [26] - which promise access to fine-tuning SD on consumer-level hardware (e.g. RTX 2080 TI and 3080 cards). One motivation for this research is to evaluate SD's performance at generating audio on consumer-grade hardware as if these fine-tuning approaches are effective, they would increase accessibility and proliferation of audio generative tools - each fine-tuned for a specific use-case or need.

The existence of the SD community also provides ample opportunity for further research and application of tools developed for SD. For example, SD provides the capability for techniques such as img2img (using a prompt as an image to generate another), inpainting (generating masks of new images over patches of existing images) and negative prompts (prompting elements to denote what the model should not generate); all of which can also be easily provided using front-end UIs such as Automatic1111 [27]. Trained SD model checkpoints can be easily inserted into these existing tools, allowing us to apply these techniques to the audio domain.

Finally, this research will also serve as an investigation into whether a latent diffusion model pre-trained on image data can serve as a parameter efficient audio generative tool. Recent research found that fine-tuning a vision transformer (pre-trained on images) on audio data, providing improved performance with the usage of fewer tunable parameters [28]. This research also provides a first look into whether SD can also provide parameter-efficient audio generation through this cross-domain transfer learning.

The three motivations of this research are therefore:

- Evaluate the feasibility of fine tuning a SD model on consumer-grade hardware to create a TTA system - thereby providing an accessible route for consumers to create their own fine-tuned systems.
- Investigate whether a fine-tuned LDM can provide parameter-efficient audio generation through cross-domain transfer learning.
- Demonstrate the potential for future research by leveraging the SD ecosystem for the audio domain.

1.2 Project overview

This project's primary aim is to perform fine-tuning on a frozen SD model with the goal of producing usable audio. This research consists of multiple components:

- A background literature review detailing the history of audio synthesis, including current SOTA approaches; the background of diffusion models, including Stable Diffusion; and project-specific domain details such as spectrogram conversion and fine-tuning methodologies. This will establish the theoretical framework for this research, and contextualise our study within the broader field of audio synthesis.
- An experimental exploration on the efficacy of LoRA fine-tuning in producing spectrograms and audio. This report includes details on each successive phase of our experimentation, the impact of different training hyper-parameters, along with results and conclusions from each phase. This is followed by a final assessment of the project output and success along with quantitative metrics.
- A brief exploration into applying SD techniques and tools to demonstrate potential further research avenues. This will involve a demonstration of the adaptability of our model for other use-cases, laying the groundwork for future research.

We will be using two audio datasets - AudioCaps [29] and AudioSet [30] - which have both been extensively used as training data for audio tasks, such as generation [31, 32, 33], classification [34, 35, 36], tagging [35] and source separation [37]. We primarily conduct our fine-tuning on multiple consumer-level GPUs (between NVIDIA GTX 1080 to RTX 3080 in terms of age) and conduct a total of 105 training runs. We also provide a companion web-page where example audios may be heard.[1]

[1]https://sites.google.com/view/diss-audio-outputs

Chapter 2

Background

Contents

2.1 Text-to-audio synthesis

2.1.1 Early synthesis

Before modern deep learning approaches, audio synthesis was conducted with traditional techniques focused on wave manipulation. Some early techniques include: Additive Synthesis (which involve combining sine waves of differing levels and frequencies [2]), Subtractive Synthesis (filtering parts of a frequency spectrum until a specific signal is obtained [3]), and Frequency Modulation (modulating the frequency of a waveform with another waveform [4]). These early forms of synthesis served as the foundation for further developments in audio synthesis. However, limitations include the requirement for manual programming and parameter adjustment, a lack of naturalness, and a limited level of control for more targeted adjustments of generated audio.

Formant synthesis, a specific method of speech synthesis, distinguishes itself from traditional techniques by focusing on modeling the resonant frequencies (formants) of the human vocal tract [38] - while it shares similarities with prior forms of synthesis in terms of involving manual signal manipulation, formant synthesis specifically targets speech synthesis by aiming to accurately reproduce the resonant properties of speech production. MITalk [39] and Klatalk [40] were examples of text-to-speech models utilising formant synthesis. This focus on formant modelling aimed to provide an audio-modelling approach solely targeting speech synthesis.

Following these methodologies, sample-based synthesis became prominent. Rather than use a fundamental waveform (such as sine/saw waves), sampled sounds were used as the base waveform to be adjusted. Sampled sounds in this context refer to recorded audio from a pre-existing source and can include: natural/urban recordings (e.g. traffic/bird song), instrument recordings (e.g. individual instrument notes) and speech. Samples can also be defined as a conversion of analogue sound to a digital representation [41]. Combining samples (along with additional manual adjustments) resulted in the synthesis of new audio waveforms, providing greater flexibility and sophistication over earlier synthesis techniques. Speech synthesis using sampling can also be referred to as 'concatenative synthesis', which is the concatenation of segments of speech samples.

2.1.2 Machine learning and initial networks

Hidden Markov Models (HMM) eventually followed sample-based synthesis as the most widely-used speech synthesis methodology [42]. These models were also one of the first machine learning (ML) generative models used for synthesis. HMMs are trained on pairs of speech samples and associated captions, with the aim of learning the statistical relationship between them. Input captions to this model are encoded as a sequence of 'hidden states', which each represent a particular acoustic/linguistic sound. By estimating state transition probabilities, HMMs generate speech by determining the most likely sequence of hidden states for a given caption. HMMs are able to capture temporal dynamics and acoustic variations, while encoding linguistic information to generate speech aligned with input captions. HMMs offer flexibility in generating speaker identities, emotions, and speaking styles, but have limitations affecting naturalness and quality [42].

Deep-learning networks - such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) - began to be used for audio and speech synthesis following HMMs. RNNs are a type of network designed to handle sequential (or temporal) data by providing connections within the network - these connections (which could form cycles) allowed for output from certain nodes to affect subsequent input to these same nodes. Long Short-Term Memory networks (LSTMs) are an RNN first proposed in 1997 [43], but first began to be used for speech and audio synthesis in the mid-2010s. LSTMs consist of a number of memory cells, which each contain gates to modify the information held in these memory cells. These gates facilitate the model's ability to selectively retain and forget information over long sequences, making LSTMs effective in capturing temporal dependencies in speech and audio data. An example of LSTMs for speech synthesis involve the network detailed in 2015 [6] (followed by further optimisations rivalling HMM performance in the following year [7] but with lower latency and resource overhead).

CNNs are a feed-forward network originally designed for processing structured grid-like data, such as images (or audio spectrograms) [44]. CNNs consist of multiple layers, in which multiple convolutions are applied using a sliding window (or kernel) moved across an image. Through the use of layers responsible for upsampling and downsampling input images, this sliding window allows for the identification of features and creation of subsequent feature maps. WaveNet is a deep neural network using a modified CNN in which dilated causal convolutions are used (a form of convolution which uses skips to capture long-range dependencies). These convolutions enable WaveNet to generate natural sounding speech [5] and - when combined with text-conditioning - state-of-the-art (SOTA) performance with text-to-speech generative quality for both English and Mandarin.

Tacotron is a text-to-speech (TTS) system which consists of an encoder, an attention-based decoder and a post-processing net [8] using the NLP (natural learning processing) model seq2seq as its architectural backbone [45]. Attention refers to a mechanism which allows a neural network to focus on specific parts of input data. Attention can be formulated as such:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q is the query vector (the elements which attention is being applied), K is the key vector (elements compared with queries to determine their relevance) and V is the value vector (content of the key element).

The encoder in Tacotron uses an RNN to process and encode input text embeddings. The decoder uses attention to determine the relevance of different parts of the encoded input sequence, generating a corresponding spectrogram - speech is reconstructed using the Griffin-Lim algorithm [46]. Tacotron outperformed the previous LSTM-focused architecture presented in 2016 [7] (in a mean opinion score test) without requiring more complex components needed for HMM-focused approaches. Tacotron 2 enhances on the first architecture by incorporating a WaveNet vocoder to directly generate raw audio samples while also incorporating training on increased amounts of data [47]. Another prominent RNN-based model is WaveRNN - released in 2018 [9].

2.1.3 Recent generative models

Following these initial networks, Generative Adversarial Networks (GANs) [48] were explored for speech and audio synthesis. GANs are composed of two core components: a generator network and a discriminator network. The generator is responsible for generating synthetic samples (resembling real data) whereas the discriminator is responsible for distinguishing input data as either real or a sample generated by the generator. During training, these two networks compete, the generator attempting to generate samples to 'fool' the discriminator, while the discriminator's parameters are updated based on its on-going performance with classifying the fake images correctly. This training process continues iteratively with the goal of reducing the discriminator's ability to distinguish between these two types of data-points (through the generation of increasingly convincing synthetic data) - after the training process completes, the learnt underlying data distribution is used to generate new samples.

Audio GAN-based synthesis models include WaveGAN (applying GANs to generate raw-waveform audio) and SpecGAN (using GANs to generate spectrograms) [10, 11], MelGAN (improved conditional waveform synthesis with reduced compute requirements and a focus on producing mel-spectrograms) [12] and Parallel WaveGAN (a model capable of producing multiple samples at once, resulting in improve generation speed and lower latency) [13].

Transformers - first detailed in 2017 [49] - are an architecture utilising multiple self-attention (an attention mechanism capable of weighing importance of different components of an input sequence) blocks in an encoder-decoder configuration. Transformers were initially designed for NLP tasks (such as machine translation) but were later adapted for audio and speech synthesis. These self-attention blocks allow transformers to model long-range dependencies, enabling increased audio generation quality. Transformers, with their ability to more effectively model linguistic and contextual information (through attention), began to be used for TTS systems - unlocking further increases in generative sample quality [14]. Prominent transformer-based TTS models include FastSpeech [15], RobustTrans [50], and FastSpeech 2 [16].

2.2 Diffusion

2.2.1 Foundations of Diffusion Models

One of the current classes of models achieving SOTA performance in image (and audio) generation are Diffusion Probabilistic Models - or Diffusion Models (DMs); the most prominent of these was first conceived in 2015 [51] and then expanded upon in 2020 (here referred to as Denoising Diffusion Probabilistic Models (DDPMs)) [52].

The core principle of DMs involves gradually adding Gaussian noise to an input image and then training a neural network to recover the original image. By learning how to reverse this process (i.e. learning an approximation of the reverse data distribution), the model can generate new images from the original image's data distribution.

The 'forward diffusion' pass of adding noise to an input image can be defined as a Markov chain that gradually adds Gaussian noise according to a variance schedule $\beta_1, ..., \beta_T$ [52]:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; (1 - \beta_t x_{t-1}), \beta_t I) \tag{2.2}$$

This equation describes the chain, where each successive x term depends on the value which came before it (x_{t-1}) and the variance schedule (β_t) - in DDPM, this schedule is set to scale linearly from 10^{-4} to 0.02 .[1]

The 'reverse diffusion' pass is performed in DDPM using a neural network trained to gradually denoise the sample obtained from the forward diffusion pass (via a series of 'denoising steps') generated from the forward diffusion step (depicted in Figure 2.1). This network uses a U-Net architecture with each step consisting of ResNet (utilising skip connections between network layers) [54] and self-attention blocks [49].

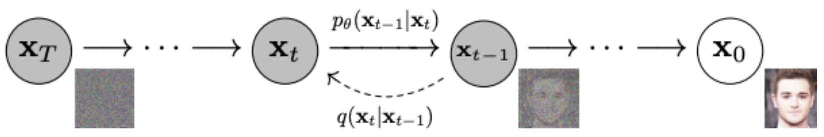

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 2.1: Reverse diffusion overview (DDPM). Reproduced from [1].

The U-Net architecture - as detailed in 2015 [55] - consists of an encoder and a decoder path consisting of successive encoder/decoder blocks. Through the encoder blocks, the feature map for an input image is obtained through successive unpadded convolutional layers and downsampled with pooling layers between each layer. After sufficient encoding blocks, the image is upsampled via transposed-convolutions. Each decoder block is connected to a corresponding encoding block with skip connections - concatenating the feature maps of these blocks together. DDPM utilised this architecture but modified this using wide ResNet blocks [56], group normalisation [57] and self-attention blocks [49].

At the end of the forward pass in DDPM (as $T \to \infty$), the sample x_T can be approximately considered an isotropic Gaussian distribution. DDPM attempts to approximate the reverse distribution $q(x_{t-1}|x_t)$ and therefore enable the sampling of x_T from a normal distribution. As this distribution is intractable, DDPM uses the (detailed)

[1] β can also be learned through the process of reparameterisation [53] (expressing this random variable with a fixed distribution, enabling faster sampling at each step of the forward diffusion process).

neural network for this approximation, learning to predict the mean and covariance matrices for each T:

$$p_\theta(x_{0:T}|y) = p_\theta(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)) \quad (2.3)$$

DDPM was assessed on dataset CIFAR10 [58] using three metrics: Inception Score (IS), Fréchet Inception Distance (FID) and Negative Log Likelihood loss. These are all distinct (but related) measures of generated image quality. IS evaluates the distribution of generated images (by evaluating how well a separate network can classify generated images) whereas FID instead compares distributions of a generated image set and a set of real images - lower scores in FID and higher scores in IS signify higher sample qualities. There is typically a trade-off between FID and IS when optimising generative models: IS can be considered as a measure of quality of image generation, whereas FID more specifically measures the similarity of generated data distribution to ground truth (or its 'realism') [59]. NLL determines the likelihood that a generative model can generate an image - with lower score signifying higher sample quality. Table 2.1 compares DDPM to existing generative models, showing SOTA performance using IS, and near SOTA performance using FID.

Model	FID	IS	NLL
Diffusion (original) [51]	–	–	$\leq$ 5.40
Gated PixelCNN [60]	4.60	65.93	3.03 (2.90)
Sparse Transformer [61]	–	–	**2.80**
PixelIQN [62]	5.29	49.46	–
EBM [63]	6.78	38.2	–
NCSNv2 [64]	31.75	–	–
NCSN [65]	8.87$\pm$0.12	25.32	–
SNGAN [66]	8.22$\pm$0.05	21.7	–
SNGAN-DDLS [67]	9.09$\pm$0.10	15.42	–
StyleGAN2 + ADA (v1) [68]	**9.74** $\pm$ 0.05	3.26	–
DDPM	9.46$\pm$0.11	**3.17**	$\leq$ 3.75 (3.72)

Table 2.1: Unconditional image generation results. Table recreated from [52].

Diffusion models are also capable of generating new samples based on 'guidance' from information such as text (or other samples). This 'guided diffusion' works by conditioning the forward diffusion steps (seen here as y) by applying this conditioning information at each diffusion step:

$$p_\theta(x_{0:T}) := p_\theta(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, y) \quad (2.4)$$

2.2.2 Expansions and developments

From this foundation, there have been several developments expanding on DDPMs resulting in improved generative performance, capabilities and robustness.

DDPM utilised a linear variance schedule (β) for forward diffusion noising steps, however, subsequent research investigating adjustments to this schedule resulted in increased performance. This improvement came in the form of IDDPM, an improved model introduced in 2021 through experiments with the variance schedule [69]. Researchers found that using a 'cosine' variance schedule (over a linear schedule) resulted in increased performance, especially for lower resolution images (such as 64x64 and 32x32). Specifically, the end of the forward diffusion process using the linear noise schedule did not contribute much to sample quality (such that skipping 20% of the reverse diffusion process did not result in worse performance). This new cosine variance schedule resulted in the destruction of less information than necessary during forward diffusion ($\bar{\alpha}_t$ here represents the variance at time step t:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1 + s} * \frac{\pi}{2}\right)^2 \tag{2.5}$$

The other key improvement made to DDPMs enabled the learning of variances in the reverse diffusion process - which resulted in significantly fewer required forward diffusion steps - using reparameterisation. This new learning objective resulted in the required number of forward diffusion steps (to produce good samples) falling from hundreds to less than 100 - with an associated reduction in NLL to **2.94** (from 3.72).

In the following year, further improvements for DMs arose through continued ablation study, showcasing SOTA performance with a new model - referred to as 'Ablated Diffusion Models (ADMs)' [70]. This paper explored a number of architectural changes - each of these changes (except rescaling residual connections) resulted in improved performance (along with a compounding effect when combined):

- Increasing depth of the model
- Increasing the number of attention heads
- Applying attention at different resolutions
- Using the residual block used in BigGAN [71] (following prior research [72])
- Rescaling residual connections with $1/\sqrt{2}$ (following [72, 73, 74])

Alongside these architectural changes, this model also explores using a classifier for 'guidance' in conditional image generation. A classifier conditioned on a specific class is used to guide the forward diffusion by providing feedback at each step - leveraging the classifier results to guide ADM in producing higher quality images. Benchmarking on various ImageNet scales, ADM was found to beat IDDPM as well

Model	ImageNet 64x64		ImageNet 128x128		ImageNet 256x256	
	FID ↓	sFID ↓	FID ↓	sFID ↓	FID ↓	sFID ↓
BigGAN-deep [71]	4.06	3.96	6.02	7.18	6.95	7.36
IDDPM [69]	2.92	3.79	–	–	12.26	5.42
LOGAN [75]	–	–	3.36	–	–	–
ADM	2.61	**3.77**	5.91	**5.09**	10.94	6.02
ADM (dropout)	**2.07**	4.29	–	–	–	–
ADM-G (25 steps)	–	–	5.98	7.04	5.44	5.32
ADM-G	–	–	**2.97**	**5.09**	**4.59**	**5.25**

Table 2.2: Model performance comparison on ImageNet dataset at different resolutions. ADM-G denotes classifier guidance, sFID refers to FID scaled to account for differences in resolution between benchmarks. Table adapted from [70].

as SOTA GAN models (see Table 2.2). Further research in 2021 found that similar performance could be achieved without the use of a classifier [76]. Classifier guidance introduces complication with the necessity of needing to train an additional classifier. A model using 'classifier-free guidance' (hence referred as CDM) involved jointly training both an unconditional and conditional DM with a single network. With a fixed probability, the label y in the conditional DM $\hat{\epsilon_\theta}(x_t|y)$ is replaced with a null label $\emptyset$ [77]. During sampling, the model output is extrapolated further in direction of $\hat{\epsilon_\theta}(x_t|y)$ and away from $\hat{\epsilon_\theta}(x_t|\emptyset)$. s in equation 2.6 denotes the 'scale' of guidance, which is set at $s \geq 1$.

$$\hat{\epsilon_\theta}(x_t|y) = \hat{\epsilon_\theta}(x_t|\emptyset) + s \cdot (\hat{\epsilon_\theta}(x_t|y) - \hat{\epsilon_\theta}(x_t|\emptyset)) \tag{2.6}$$

Adjusting this guidance scale resulted in FID monotonically decreasing and IS monotonically increasing with increasing values of guidance. CDMs achieved SOTA FID performance in 128x128 ImageNet against ADMs (achieving **1.48** against ADMs **2.07**) and outperforming BigGAN-deep in FID and IS.

Shortly after the publication of CDMs, GLIDE was released [77], a model enabling near-SOTA image generation and editing with text conditioning.[2] GLIDE explored the use of classifier-free guidance, as well as CLIP guidance; CLIP is a neural network trained to learn joint representations of text and image [78]. CLIP models contain an image encoder and a caption encoder, and are trained on pairs of captions and images - CLIP generates a loss which denotes how close an image is to a caption. CLIP guidance operates similarly to classifier guidance, but using a CLIP model in place of a classifier. With this functionality, CLIP has previously been used to guide GAN-based generative models using a provided text-caption [79, 80, 81] which inspired its application to DMs.

[2]Similar to the concept of guidance, conditioning provides a vector to use additional information to affect the generation process. Conditioning is the process of providing explicit instructions (such as a text prompt) compared to the less explicit and more subtle influencing effect of guidance.

Quantitative results from these two guidance strategies show classifier-free guidance achieving favourable results over CLIP guidance. Classifier-free guidance achieve pareto optimality with the trade-off in FID and IS, as well as achieving improved 'Elo' human evaluation scores in photorealism and caption similarity. Using classifier-free guidance, GLIDE achieves zero-shot[3] generation quality quantitatively competitive with SOTA models (e.g. transformer-based DALL-E) and qualitatively outperforming with human evaluation.

2.2.3 Latent Diffusion Models and Stable Diffusion

Diffusion models require significant computational resources - resulting in restrictive resource constraints for training and inference. Training the most powerful DMs can require hundreds of GPU days [1] (e.g. 150-1000 V100 days for ADMs [70]) with inference requiring resources in the region of 5 days to produce 50,000 samples with ADM on a single A100 GPU [70]. Rombach et al. proposed the Latent Diffusion Model (LDM) in 2022 [1] to improve accessibility to DMs - a diffusion model which capitalises on encoding representations in latent space.

Models operating in pixel-space operate in a high-dimensional space. This space necessitates the need for significant compute, which incentivises the need to transform image data into either lower-dimensional feature space or latent space. Latent space refers to a low-dimensional abstract data representation which aims to compress data by reducing the data distribution to a minimal representation. Encoding data into latent space significantly lowers compute required to interact with this data and only decoding back into pixel space after necessary computations are performed.

LDMs expand on previous diffusion models by running the forward-diffusion and denoising processes in latent space, rather than pixel space. The encoder ε encodes x into a latent representation $z = \varepsilon(x)$ and the decoder D reconstructs the pixel representation/image from the latent ($\bar{x} = D(z) = D(\varepsilon(x))$). The architecture can be seen in Figure 2.2. Forward diffusion (Diffusion Process) as previously discussed occurs from Z to Z_T, and the reverse diffusion process is performed using a U-Net, with a Variational Autoencoder (VAE) used to encode and decode these representations between pixel and latent space [82].

LDMs offer a few differences to existing DMs, the first of these focuses on introducing a new model of image compression - which is used to encode images into their latent representations. This compression is separated into two stages: a "perceptual compression" stage which removes high-frequency details, and a second "semantic compression" stage in which the "generative model learns the semantic and conceptual composition of the data"[82]. This compression method results in a compression which allows for more "faithful and detailed reconstructions than previous work".

LDMs also introduce a new conditioning mechanism, offering flexible conditioning

[3]Zero-shot refers to the ability a model has to be generalised to unseen concepts. In this context, GLIDE can generate images/concepts without explicit training on these specific concepts.

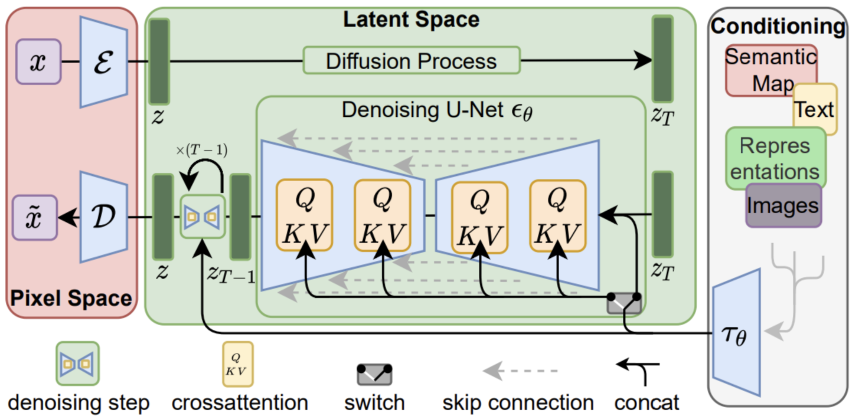


Figure 2.2: LDM architecture illustrating usage of latent space. Reproduced from [1].

using a domain specific encoder (denoted in Figure 2.2 as τ_θ). The representation generated by this encoder is passed through a cross-attention layer, and added to each layer of the denoising U-Net. This representation enters into the attention layer as such (where ϕ is a function mapping the latent representation to a query vector):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{2.7}$$

$$\text{where } Q = W_Q^{(i)} \cdot \phi_i(z_t), \quad K = W_K^{(i)} \cdot \tau_\theta(y), \quad V = W_V^{(i)} \cdot \tau_\theta(y) \tag{2.8}$$

LDMs were introduced with a CLIP-based text encoder to generate usable embeddings from text prompts. These embeddings are what are added to each cross-attention layer. The result of these improvements is an image generation model, which can be conditioned by a number of different modalities (e.g. text, image etc.), capable of producing SOTA-levels in image quality via unconditional, text-conditional and class-conditional generation.

Stable Diffusion (SD) is an open-source LDM trained on compute resources provided by AI-firm Stability AI [1]. Since its release, SD has enabled consumer-level access to SOTA conditioned image generation, as well as several other features, such as image conditioning, negative prompts, image in-painting, and super-resolution. Other prominent text-conditioning generative image models, including OpenAI's DALL-E [83] and Midjourney. Inc.'s Midjourney [84]. These models - due to their closed-source nature - are unable to be fine-tuned and therefore unsuitable for this research exploring image generative models for audio generation.

SD was originally trained on the LAION2B-en dataset, an English language subset of the LAION5B dataset [85] - a large-scale multi-modal dataset consisting of 5.85 billion image-text pairs (with 2.3 billion English pairs). For subsequent releases of SD, further training was conducted on different subsets of the data such as 'laion-improved-aesthetics' - a subset of LAION2B-en consisting of images greater than 512x512 and filtered by quality and presence of watermarks. The most recent 1.x model is 1.5, which was trained for 237,000 steps at 256x256 on LAION2Ben, 194,000 steps at 512x512 on a high-resolution subset, then further trained for 515,000 steps using the 'laion-improved-aesthetics' dataset before resuming training once more for 595,000 steps on a new 'laion-aesthetics v2.5+' subset.

SD 1.5 is therefore trained for almost 2 million steps, on over 2 billion data samples of varying classes. SD is in constant development, with SD 0.9 XL released in June 2023 offering 1024 x 1024 image generation and SD 2.0 (released in November 2022) providing 768 x 768 image generation at the cost of higher memory requirements. SD 2.0 in particular also found that using a different text encoder (OpenCLIP [86]) greatly improved the quality of generated images compared to earlier V1 releases.

SD - due to its open-source status - has been used prominently, resulting in a community of researchers and consumers supporting its continual development. Multiple consumer-targeted tools - such as Stability AI's DreamStudio[4] and the AUTOMATIC1111 WebUI [87] - have increased user accessibility to fine-tuning and running inference with SD models. These tools, as well as the work of Hugging Face in improving accessibility to diffusion models [24] - has resulted in large communities such as the Stable Diffusion subreddit[5] frequently discussing outputs, training details and new developments. Tools - such as ControlNet [88] - based on new research also offer additional flexibility and options for conditioning generative output. Researching the limits of SD for generating audio has the potential to enable new paradigms of generation by leveraging existing SD tools, research and communities.

2.2.4 Diffusion audio synthesis models

Despite their original purpose of image synthesis, recent research has also explored using DMs for audio synthesis [89]. Pioneering works involve Diff-TTS (the first application of applying DDPMs for spectrogram generation) [17], Grad-TTS (using DDPMs to generate mel-spectrograms) [18] and ProDiff (reducing required iterations through parameterisation) [19]. Other works include DiffWave (which conditioned the vocoder on mel-spectrograms) [20], and bilateral denoising diffusion models (BBDMs) (which proposed including an additional network to predict the noising schedule used in the forward diffusion steps) [21].

These results show DMs clear capability in synthesising audio and speech - achieving SOTA performance (in subjective assessment with expert listeners [89]). The low RTF values also indicate that diffusion-based TTS models can generate speech with low latency, potentially making them practical for real-world (live) applications where efficient synthesis is required.

DMs have also successfully been used for TTA generation in non-speech domains. Examples include AudioLDM [31], Make-An-Audio [90] and TANGO [32] - which

[4]https://dreamstudio.ai/

[5]Found at https://reddit.com/r/stablediffusion, this is a community ranked in the 99th percentile by size across all communities on popular forum Reddit

Methods	Dataset	MOS ↑	RTF ↓	
Diff-TTS [17]	LJSpeech	4.337	0.035	
Grad-TTS [18]	LJSpeech	4.44	0.012	
ProDiff [19]	LJSpeech	4.08	0.04	
NoreSpeech [22]	LibriTTS	4.11		
DiffWave [20]	LJSpeech	4.47		
BBDM [21]	LJSpeech	4.48	0.438	-

Table 2.3: Performance of Diffusion-based TTS models. (S)MOS (Subjective Mean Opinion Score) is a subjective measure used to evaluate quality and naturalness of speech, RTF (Real-Time Factor) is a measure of speed/efficiency. Table adapted from [89].

utilised a fine-tuned large language model to improve upon AudioLDM's previous SOTA performance. Using Fréchet Audio Distance (FAD) - FID adapted for the audio domain [91] - TANGO achieves a score of **1.59**, beating AudioLDM's score of **2.08** (and **1.96** when further fine-tuning AudioLDM on the AudioCaps dataset). This field is under continual development, with AudioLDM2's recent release in August 2023 exceeding TANGO as SOTA with a FAD of **1.42**.

2.3 Spectrograms and Riffusion

2.3.1 Spectrogram: audio representation

Waveforms and spectrograms are visual representations of audio. A waveform is a time-domain representation of a signal, illustrating changes in amplitude (loudness) over time. Waveforms do not provide frequency information at specific points in time so a spectrogram representation is needed to capture additional details of the signal. Audio spectrograms are visual representations of a particular audio's frequency - specifically, the magnitude and phase (the timing and position of this magnitude) of different frequency components. Spectrograms are a frequency-domain representation of an audio signal, providing information about frequency over time.

Waveforms can be converted to spectrograms using short-time Fourier transform (STFT), which approximates audio as a combination of sine waves of varying amplitudes and phases [23]. STFT is also invertible, which enables audio to be reconstructed from a spectrogram image.

As audio signals in a digital system are digital, applying the STFT to a digital waveform involves using the discrete Fourier transform (DFT):

$$X(\omega_k) \triangleq \sum_{n=0}^{N-1} x(t_n) e^{-j2\pi kn/N} \tag{2.9}$$

where $X(\omega_k)$ is the spectrum of the audio signal at a particular frequency, $x(t_n)$ is

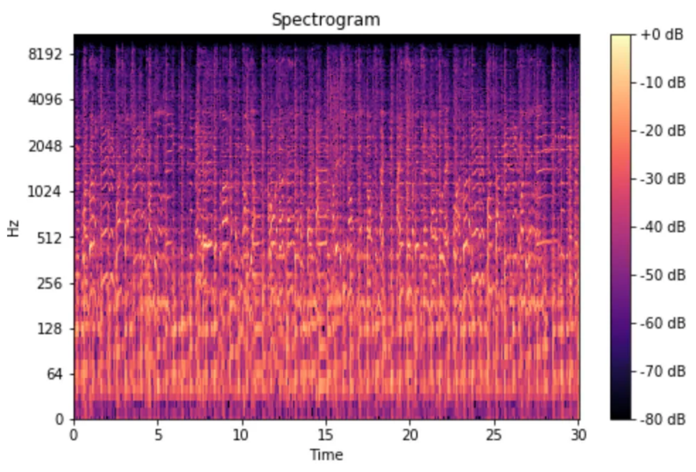


Figure 2.3: Example of a spectrogram - y-axis: frequency, x-axis time, colour: decibel. Reproduced from [92]

the time signal at a specific time t_n, $\sum_{n=0}^{N-1}$ is the summation of contribution of each sample from the original signal and $e^{-j2\pi kn/N}$ representing how a specific frequency components are shifted in phase compared to a reference point. DFT allows us to decompose a signal into separate frequency components, which can then be displayed visually on a spectrogram.

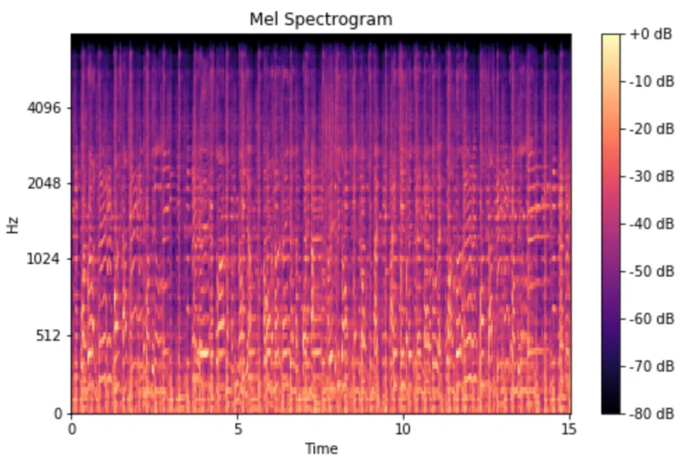


Figure 2.4: Example of a mel spectrogram - y-axis: frequency, x-axis time, colour: decibel. Note the change in frequency scale on the y-axis. Reproduced from [92]

As human perception of sound is concentrated within a narrow range of frequencies (e.g. higher sensitivity to lower frequencies compared to higher frequencies), the Mel Scale was developed to more accurately align frequency representation to human perception [93]. Converting a spectrogram to a mel spectrogram results in a spectrogram which emphasises perceptually relevant features of our target domain (speech) (an example can be seen in Figure 2.4). Conversion to the (log-scaled) mel spectrogram from a spectrogram can also be expressed mathematically [93]:

$$m = 2595 \cdot \log_{10}(1 + \frac{f}{700}) \tag{2.10}$$

Generating spectrograms through image generation essentially provides the ability to generate audio by converting these images through downstream audio processing. This leads us to Riffusion, an application which showcases the possibility of fine-tuning SD to generate these pictorial representations.

2.3.2 Riffusion: generating music spectrograms with Stable Diffusion

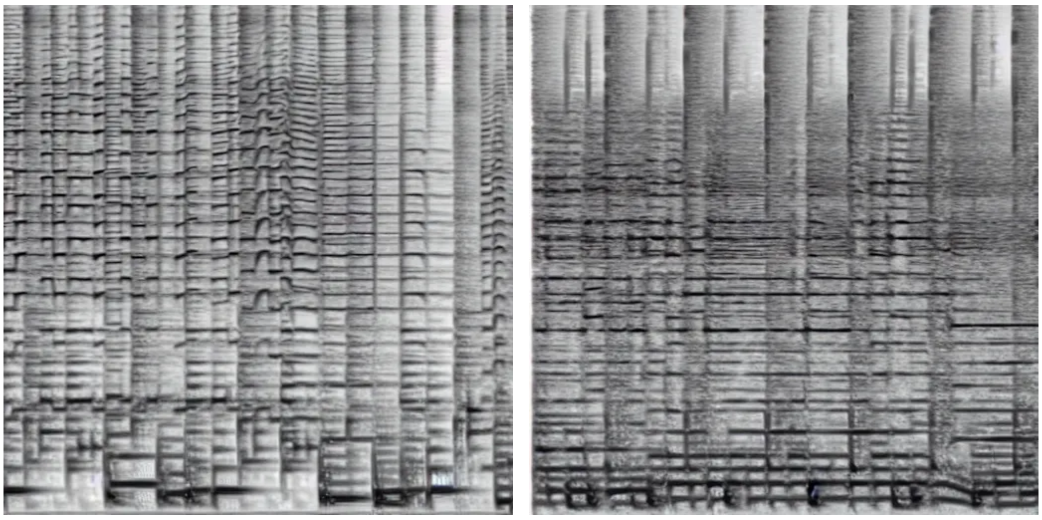


Figure 2.5: Spectrograms generated with Riffusion using text prompts. Left:'funk bassline with a jazzy saxophone solo', Right:'rock and roll electric guitar solo'[23]

Riffusion [23] uses a fine-tuned Stable Diffusion (SD) model to generate spectrograms of music (which are then converted to audio using STFT). This model is tuned on images of spectrograms paired with corresponding captions and is the first notable example of using SD to generate audio as a spectrogram and subsequently, using SD for text-guided audio generation. This provides evidence for the feasibility of using a diffusion model - which is pre-trained on images - as an effective TTA system.

Riffusion is compatible with SD features, with image conditioning, in-painting, negative prompts and interpolation fully functional. This allows the user to condition the generation of new sound with an existing spectrogram while preserving the existing audio's structure. An example of this is interpolating a 'rock and roll electric guitar solo' to an 'acoustic folk fiddle solo' resulting in a change of style, while preserving the melody of the original music clip. Successful implementation of SD for non-music audio domains should also provide working functionality of these features.

The spectrograms generated by Riffusion do not provide information about phase (only magnitude); the phase for these spectrogram images are approximated using an additional algorithm (Griffin-Lim) [46] which iteratively estimates phase information using a spectogram's magnitude. This implies any audio spectrograms generated by our research would also need to consider the need for phase reconstruction. Other explored methods of phase reconstruction range from ML-based approaches (e.g. using deep neural networks[94]) to expansions on classical techniques (e.g. improvements on Griffin-Lim and manipulations of STFT) [95, 96, 97]. Application of different phase reconstruction techniques may result in improved fidelity of converted generated audio spectrograms and could be considered as a potential hyperparameter to tune. Improper phase reconstruction can result in low-quality audios with distortion and artifacts.

Unfortunately, training code and information for Riffusion is unavailable, which results in a necessity to explore methods of fine-tuning stable diffusion.[6]

2.4 Fine-tuning methodologies

Since their inception, multiple techniques have emerged for fine-tuning LDMs to generate new concepts, some of which require as few as 3-5 images [98, 26].

Introduced in 2022, Textual Inversion is an approach [98] which allows SD to be fine-tuned with a minimal set of images (3-5) to generate novel images or subjects. This method fine-tunes the text encoder of an LDM to map a new text prompt to its most suitable embedding. This research also finds that a single word embedding is sufficient to learn unique and varied concepts. Examples of how learned concepts are composed in this fine-tuned model can be seen in Figure B.1.

DreamBooth [26] offers a similar benefit to Textual Inversion, enabling SD to generate new concepts - similarly using only 3-5 input images. DreamBooth differs from textual inversion by focusing on tuning the entire LDM - rather than the text encoder solely. DreamBooth boasts improved prompt accuracy and image fidelity over textual inversion in certain contexts. An overview of this fine-tuning approach can be seen in Figure B.2.

A third approach of fine-tuning is LoRA [25] - or Low-Rank Adaptation of Large Language Models (LLMs) - a methodology created to fine-tune LLMs with billions of parameters (e.g. ChatGPT-3). An adaptation of this methodology for LDMs [99] provides faster training than previous approaches (e.g. DreamBooth), lower compute requirements (enabling fine-tuning with a single 2080 Ti) and much smaller trained weights - enabling increased ease of model sharing. Visual depictions of these strategies can be found in Appendix B

[6]The authors of this software were contacted with requests for training clarifications but did not respond.

These three methodologies each increase the accessibility of SD fine-tuning, reducing training set size and compute requirements. Colossal-AI [100] is an open-source package of tools enabling further improvements in efficiency, reducing training memory consumption by 'up to 5.6x' and hardware cost by 'up to 46x' (from A100 to RTX 3060). Hugging Face also provide functionality and support for these training paradigms as part of their Diffusers library[7] [24]. These resources present good feasibility for this research to fine-tune SD on limited project resources - mirroring similar compute resources of average consumers.

2.5 Cross-domain transfer learning

Stable Diffusion models are primarily pre-trained on image data for the purpose of image generation. However, emerging evidence suggests that cross-domain transfer learning is effective. In this approach, models pre-trained in one domain, such as image processing, can be efficiently adapted for other domains, like audio, with comparatively minimal fine-tuning. This method circumvents the need for extensive and costly domain-specific training by leveraging frozen pre-trained models, even those initially trained for different domains.

Recent research [28] has explored the use of pre-trained Vision Transformers (ViTs) for audio-visual tasks. These studies inject trainable parameters into every layer of a frozen ViT network and have found that ViTs can generalize to audio-visual data without requiring any audio-specific pre-training. Previous work in cross-domain transfer learning has primarily focused on language models [101, 102, 103, 104] or on more closely related domains (such as image to video) [105, 106, 103].

The current landscape of cross-domain research sets the stage for this study, which will explore the potential for efficiently adapting image-centric Stable Diffusion models for audio generation. This would contribute novel insights into the adaptability of diffusion models across different domains and modalities.

[7]This is a library containing pre-trained diffusion models with tools to enable custom training. Training overview detailed here: `https://huggingface.co/docs/diffusers/training/overview`

Chapter 3

Stable Diffusion for Audio Spectrograms

Contents

This research focuses on exploring the efficacy of fine-tuning a pre-trained, open-source Latent Diffusion Model (LDM) to generate spectrograms that can be converted to usable audio. A successful demonstration of generating usable audio via this method achieves two main objectives:

- It showcases that Stable Diffusion (SD), even when pre-trained with vast amounts of non-spectrogram data, can be feasibly adapted to produce quality samples from entirely different domains.
- It paves the way for leveraging a variety of SD-specific tools and techniques for the audio domain, broadening the horizons of SD applications for audio research and development.

To this end, we present *Bootstrapping Stable Diffusion for Audio Synthesis*, a first investigation into whether a pre-trained diffusion model can be successfully applied cross-domains for non-music audio generation.

3.1 Stable Diffusion 1.5

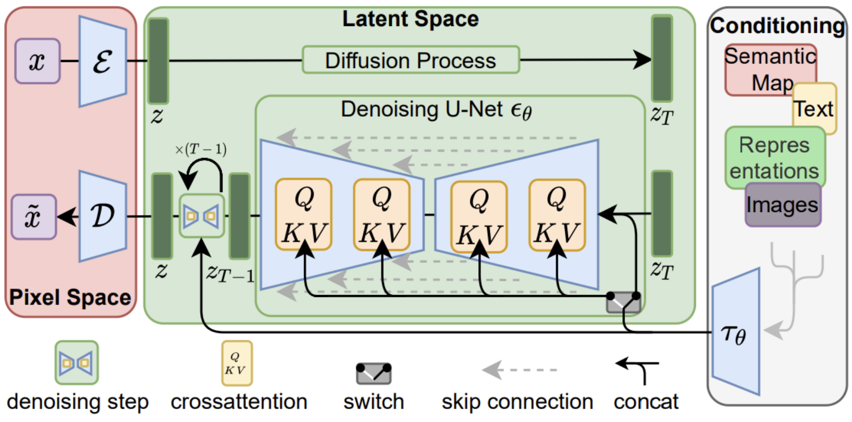


Figure 3.1: Stable Diffusion architecture [1]

SD is an LDM designed for image generation, equipped with functionality to utilise various prompt modalities for generation guidance. This model works by first taking multiple 'diffusion steps' in latent space, which involves iteratively adding noise to a sample z to obtain a noised sample z_T. This sample is then passed through a U-Net, while providing conditioning (such as text, or other images) at each level of the U-Net through cross-attention layers. Stable Diffusion v1.5 is a checkpoint of SD which is trained for almost 2 million steps on over 2 billion image-text pairs from the LAION-5B dataset [85]. A brief overview of the model components are as follows [1, 107]:

- A **Variational Autoencoder** [82] is used to encode (ε) and decode (D) images between pixel and latent space. A relative downsampling factor of 8 is used, and images of shape $HxWx3$ are mapped to latents of shape $H/fxW/fx4$ where f is this downsampling factor (8).

- A **CLIP model (ViT-L/14)** [78] is used as the text encoder (τ_θ) to convert input guidance (e.g. text, images) into injectable embeddings.

- A **U-Net** [55] is used as the model backbone ϵ_θ for the denoising process.

- Outputs of the text encoder are fed into the U-Net via crossattention (Q, K, V).

- The loss is a reconstructive objective between the noise added to the latent, and prediction made by the U-Net. We use the **mean-squared error (MSE)** of the difference between our denoised outputs and the ground truth. This can be interpreted as how well our model can denoise corruptions introduced during the diffusion process.

Although SD2.0 has been recently released, we elected to use SD1.5 for this research for a couple of reasons. The first of these reasons is to ensure parity with Riffusion [23]. Riffusion uses a stock SD1.5 model with no modifications, which was fine-tuned with images of spectrograms paired with text. As Riffusion successfully generates spectrograms of music, choosing SD1.5 for this research is a sensible starting point for investigating whether SD can be fine-tuned for non-music spectrograms. We also only similarly generate 512 x 512 spectrograms, removing the need for models capable of producing larger dimensions - such as SD0.9 and SD2.0

Secondly, memory and compute requirements are heavily considered for this research - as we had access to mainly consumer-level GPUs (e.g. 1080 Ti, 2080 Ti, 3080) with average VRAM amounts of 11GB. Components of this methodology were chosen to ensure effective fine-tuning with these restrictions, and SD1.5 was confirmed to be fine-tunable with our current VRAM restrictions using multiple existing training scripts (which provide methods to further reduce memory overhead [24]).

3.2 Dataset

The speech domain was initially targeted for this research, but focus quickly shifted to general (non-speech) audio effect generation. As the inference outputs of SD have a constant resolution (in this case 512 x 512 pixels), generating sentences of varying size would have introduced additional complication. Constraining the training data, as well as the inference output length, to a constant length (10 seconds) allows this research to focus on the effect of fine-tuning SD on spectrogram generation. Speech has much more variability in length in order to be legible - for example, a sentence of one word would require much less time (or ‘space’ in the spectrogram) compared to a longer sentence. Natural and urban audio effects can have variability within a predefined range of time and still be legible and useful. The data described in this section were therefore selected for the goal of non-speech audio generation.

3.2.1 Description

Text-to-audio (TTA) models such as AudioLDM [31] and TANGO [32] used a number of datasets for their training - including AudioSet (AS) [30] and AudioCaps (AC) [29]. AS is a dataset containing over 2M human-annotated 10-second video clips collected from YouTube and is one of the largest accessible audio datasets - containing 527 classes and almost 6,000 hours of audio. AC is a dataset containing human-written text pairs describing audio events in sound; this audio is sourced from AS and consists of around 46,000 audio clips. For both AS and AC - as this data is sourced from YouTube - the quality can also not be assured to be standardised across audio clips.

AS contains a number of urban and natural audio classes, such as animal sounds, instruments, vehicles & tools. The primary objective of our research dataset was to provide a set of distinct classes to convert to spectrogram for our SD model to be fine-tuned with. AS is extensively used for downstream tasks such as audio generation [31]m audio classification [34, 35, 36], audio tagging [35] and audio source separation [37].

AC was formed by using audio sourced from AS but providing each sample with a natural text-description. Examples of descriptions include "A motor vehicle engine is running, a decelerating buzzing occurs, mechanical whines occur, and the motor vehicle engine accelerates" & "A frog chirps and other frogs and crickets chirp in the background". Although AC is a much smaller dataset than AS, the higher fidelity and variety in text descriptions provide a greater text representation of audio, enabling our model to learn more granular descriptions of audio - which should provide greater capability for text-guided audio generation at inference time.

AS exhibits significant label bias, as evidenced by a difference of 660,282 between the average count of its top three most common classes and its three least common ones [29]. Data ontology for AC can be found in Appendix A.

3.2.2 Collection

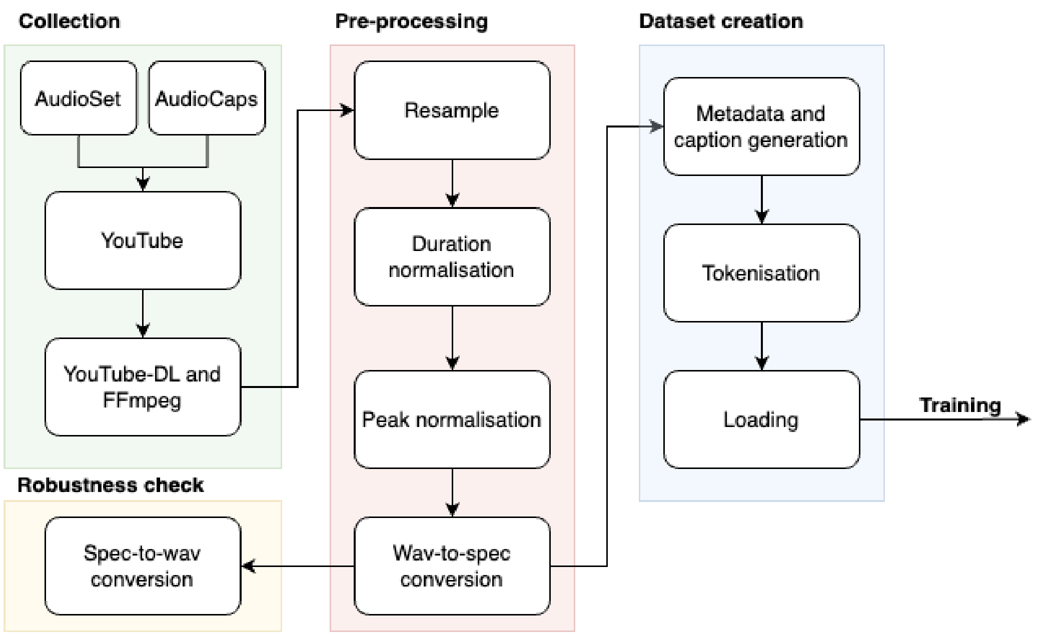


Figure 3.2: Pre-processing steps undertaken for dataset creation

The AS and AC datasets both provide metadata on where each audioclip is sourced from (via YouTube URL), as well as the duration over which this audioclip takes place - this allows users to download the datasets directly. With this metadata, these datasets were scraped using Youtube-DL [108] and FFmpeg [109] - the former a python package enabling direct download of YouTube videos, and the latter a framework converting 10-second video clips to audio. Using these two software packages in conjunction provides a dataset of .wav files corresponding to the information provided in these metadata files.

The total downloaded is less than the numbers quoted by the dataset creators, due to YouTube videos becoming unavailable over time - either due to copyright takedowns, privacy settings or removals otherwise. As AudioCaps is a much smaller dataset than AudioSet, we elected to download the entire dataset however - due to aforementioned availability problems - this resulted in a dataset of 25,063 samples out of the quoted approximate 46,000 value.

3.2.3 Pre-processing

A number of processing steps are then applied to the audio data to prepare them for our experiments. As the data is downloaded from YouTube, the characteristics of the audio cannot be assumed to be consistent across the dataset.

Resampling

The first stage is to resample the data to ensure this is consistent across samples. The sample rate determines the highest frequency which can be produced. Following the Nyquist-Shannon sampling theorem [110], we pick a sampling rate which is greater than twice the expected highest frequency in the audio signal to avoid distortion. A sample rate of 44.1kHz is a common sampling frequency used in (the music) industry, chosen mostly in part to be greater than twice the highest human-audible frequency of 20kHz [111]. Our data is initially resampled to 44.1kHz to ensure consistency across the dataset as well as ensure all audible frequencies are captured without distortion.

Duration normalisation

After resampling, the data length is then standardised to ensure each clip is 10 seconds in length. The download scripts specify 10 seconds to be downloaded so this stage serves as additional robustness for events where clips may have been downloaded incorrectly. Clips that are too long are trimmed to 10 seconds, while those that are too short are padded with zeroes (or silence) at the end.

Peak normalisation

To achieve further consistency across the training data, the volume/amplitudes of this data are normalised through peak-normalisation. This scraped data has inconsistent volumes between clips and we want to avoid the model becoming sensitive to volume differences, rather than inherent characteristics of the audio. Peak normalisation operates by dividing each audio signal by its maximum absolute value. Consequently, the loudest peak in any audio signal will be scaled to exactly 1 (or -1 for negative peaks), with all other samples in the clip being proportionally scaled based on this peak. This method of normalisation does not alter the audio's dynamics and instead adjusts the overall level of each file. This ensures a uniform amplitude across the dataset and helps prevent clipping — a type of distortion that occurs when the signal's amplitude surpasses its representable limit.

Waveform to spectrogram conversion

Once the audio data is processed, these are then converted to spectrograms using code provided by Riffusion [23]. The conversion to spectrogram uses short-time Fourier transform (STFT) within functions provided by PyTorch. STFT parameters can affect the final spectrogram image, especially its resolution - appropriate parameters must be selected to ensure the images generated are of a resolution which can be interpreted by the model, in this case 512x512.

Within STFT, the hop length determines the number of samples between successive frames. Adjusting this can result in adjustments to the width of the spectrogram - with larger spectrograms resulting from shorter hop lengths and subsequent better time resolution from more frequent samples. Hop length is calculated as such:

$$hoplength = stepsize(ms)/1000 * samplerate \tag{3.1}$$

Fixing the sample rate at 44.1kHz, the step size is adjusted to achieve a width of 512 pixels. These spectrograms are also scaled into mel-spectrograms. For this scaling, we set the number of frequencies to 512, with the min and maximum frequencies bounded between 20Hz and 20kHz - the range of human hearing. This scales the height dimension of our spectrograms to 512 - resulting in images of 512x512. An additional horizontal cropping stage is added to remove any width arising from rounding errors in the hop length calculation - this usually results in the removal of one column of pixels at most.

Spectrogram to waveform conversion

Converting spectrograms back into waveforms involves applying the Griffin-Lim algorithm. Waveforms are first inverse mel-scaled using the parameters used to generate the spectrograms before applying Griffin-Lim to generate a waveform. Both Griffin-Lim and mel scaling are lossy operations, resulting in a loss of fidelity and information when converting mel-spectrograms to waveforms and vice versa. Spectrograms are converted back as a further robustness check to evaluate the extent of this loss, with original training audio logged alongside converted audio to ensure this loss is acceptable. Lastly, this conversion operation is also employed to transform the model outputs into audible formats.

3.2.4 Dataset creation

After these preprocessing stages, we now have a collection of 10-second, 44.1kHz and peak normalised waveforms - along with a corresponding set of spectrograms created from these waveforms. This research used HuggingFace's datasets package, which enables the creation of datasets from folders containing corresponding metadata files.

Metadata and caption generation

AC and AS have pre-defined splits for training, validation and testing. To create the dataset, we separate this data into corresponding train/test/val folders and then create metadata files, which contain the spectrogram image, a caption and the audiofile (for logging purposes). For AS, the captions follow the pattern of "A spectrogram of a class" where class is the specific audio class (e.g. snare drum, bird song). AC provides captions, so a signifier of "A spectrogram of" is added to the start of every caption. This was done to ensure our model could generate spectrograms by fine-tuning with these signifiers, and by providing a signifier to be added before prompts at inference time to ensure we generate our target domain (spectrograms).

Tokenisation

After preparing the captions and the dataset, they are subjected to a tokenisation process. This step employs a CLIP tokeniser to transform the captions into text embeddings tailored for a CLIP text encoder used in subsequent stages. This transformation breaks the captions into individual tokens and maps them to an intermediate representation, which the model's text encoder can process more efficiently.

Loading

After these pre-processing steps, the training and validation data is loaded into the model for training. These are loaded in pre-specified batch sizes - with the order shuffled for training data and not shuffled for validation data. The training data is shuffled to avoid any bias inherent in the data collection (such as clumping of classes) and minimise any bias from the model memorising specific sequences or ordering of the data. The validation data is not shuffled to ensure predictability when evaluating the model's performance on this unseen data - this also enables ease of comparison between different models (or the same model at different stages of training).

3.3 Fine-tuning

Fine-tuning refers to performing additional training on a pre-trained model to accomplish a specific task. Using a task-focused dataset allows us to leverage the power of large-scale pre-trained models and adapt them to new domains and tasks. This section will detail our fine-tuning technique and methodology.

3.3.1 Low-Rank Adaptation (LoRA)

We chose to fine-tune our model using LoRA or Low-Rank Adaptation of Large Language Models (LLMS) [25]. This technique originally developed for LLMs was first adapted for SD in 2023 [99]. Fine-tuning a model with a large number of parameters (such as GPT-3 with 175B parameters) is prohibitively computationally expensive. LoRA significantly reduces computational cost by freezing pre-trained model weights, and then only fine-tuning on newly injected weight matrices added to existing weights. For SD, these are injected in the cross-attention layers; this mirrors LoRA's original focus on fine-tuning LLM Transformer attention blocks. This reduces the number of parameters needed to fine-tune the overall model, dramatically reducing memory overheads.

LoRA also has the benefit of avoiding catastrophic forgetting, a problem where existing pre-trained weights within a network may be overwritten, resulting in a loss of prior generative power. As LoRA preserves the original SD model weights, it should ensure that existing SD capability prior to fine-tuning is unaffected. Diagram illustrating LoRA can be found in Appendix B

Existing training scripts and packages [24] suggest that LoRA enables fine-tuning on GPUs with VRAM as low as 11GB - whereas other popular fine-tuning methodologies such as Dreambooth [26] potentially necessitate upwards of 24GB of VRAM [112].

LoRA's benefits are therefore:

- Prior existence of fine-tuning scripts
- Reduced memory requirements
- Compact fine-tuning weights compared to non-LoRA fine-tuned weights. This increases the ease of both sharing fine-tuned model weights and using existing SD tools.

These benefits - in light of project compute and timeline restrictions - resulted in LoRA being chosen as our fine-tuning methodology.

3.3.2 Fine-tuning procedure

We chose to adopt an existing fine-tuning script[1] which implements LoRA fine-tuning using a weighting strategy known as Min-SNR [113]. Min-SNR promises 3.4x faster convergence speeds over previous strategies by using signal-to-noise ratios to balance conflicting optimisation directions between timesteps (which was found to be responsible for previously slow convergence speeds for diffusion models).

We conduct fine-tuning and log progress using a number of quantitative and qualitative metrics to evaluate the efficacy of our training:

[1]This script can be found in the Diffusers library [24]

- Average mean-squared error (MSE) loss for training and validation sets.
- A randomly selected training and validation input sample is logged per epoch to spot-check correct data-loading. We log spectrograms, original audio as well as these spectrograms converted to audio to evaluate the robustness of our spec-to-wav conversion.
- For each epoch, we run inference to generate a number of spectrograms using pre-set validation prompts. We log generated spectrograms alongside their conversions to audio.

These outputs allow us to evaluate changes in loss curves, as well as qualitatively evaluate the quality of generated samples before further evaluation.

Inference outputs can provide an approximate indication of how our model is performing. For each inference step, we provide a number of validation prompts which consist of prompts present in the validation set. We also provide simpler prompts to evaluate how the model may generalise to prompts with similar textual embeddings without replicating the entire validation prompt exactly. For each prompt, we generate four images using a generator which is seeded with a consistent seed across experiments. Examples of prompts (each prefixed with "a spectrogram of" are:

- "A dog barking"
- "A man laughing"
- "A horn honking followed by a man laughing then talking and plastic clanking on a hard surface"
- "Music plays and someone speaks before gunfire and an explosion occurs"

Perceptual evaluation of inference outputs focuses on the quality of the audio, and its success at matching the inference prompts. These early evaluation stages allow us to perform quick initial tuning of hyperparameters before evaluating our best-performing models using a quantitative evaluation framework to produce FAD and IS scores - this is detailed in Section 5.1.

3.4 Hyperparameters

A primary objective of this fine-tuning work involves the selection of appropriate hyperparameter values to optimise our model's performance - optimisation of these parameters will be performed by evaluating the results of different experiment configurations. The parameters for this model are the following:

- **Learning rate** - Our model employs an Adam-based optimiser. This algorithm actively adjusts the effective learning rate during training based on its estimations of the mean and variance of each parameter [114]. Specifically, we use

AdamW, an extension of Adam with added weight decay. The learning rate we select serves as the base rate for the Adam optimiser. This rate influences the optimiser's estimations of the mean and variance of each parameter and sets the general scale for step sizes during gradient descent. It defines the intensity of these updates. An excessively high learning rate can lead to divergence (due to overly large gradient updates), while an overly low rate might cause the training to get trapped in local minima.

- **Adam parameters** - The optimiser behaviour can be controlled further by adjusting three parameters β_1 β_2 and the weight decay.
  - **β_1** represents the moving average of the gradients (the mean), adjusting this parameter determines the exponential decay rate of past gradients being considered by the optimiser when adjusting its steps. A β_1 closer to 1 results in a longer memory, whereas values closer to zero result in a shorter memory. Setting this too high might lead to slow convergence, whereas values too low might lead to unstable optimisation. The original Adam paper sets this value to 0.9 as default [114].
  - **β_2** represents the moving average of past squared gradients (the variance), adjusting this parameter determines the exponential decay rate of past squared gradients being considered by the optimiser when adjusting its steps. Similar to β_1, setting this too high might lead to slow convergence, whereas values too low might lead to unstable optimisation. The original Adam paper sets this value to 0.999 as default [114].
  - **Weight decay** is a form of regularisation which discourages large weights by applying a decay penalty to weights during optimisation. The form of weight decay used in our model involves adding this cost directly to the weights to avoid conflict with Adam's adaptive learning rate (rather than during gradient calculation). The purpose of this regularisation is to prevent overfitting by promoting simpler models with smaller weights - setting a weight decay too low may lead to overfitting, whereas a weight decay too high may lead to underfitting and degraded performance from overly reduced weights.
- **Warm-up steps** - Warm-up is a technique which involves gradually increasing the learning rate to the specified parameter value over a number of warm-up steps. These steps - along with our optimiser - help to stabilise our training, by avoiding aggressive weight updates at the beginning of training caused by immediately using our maximum learning rate. Starting with a smaller learning rate allows our model to settle into a more stable region of the loss landscape before larger weight updates. If our learning rate is too high at the start of training, weights could be updated too aggressively, leading to either divergence, or oscillations in our loss. For example, SD1.5 training used 10,000 warm-up steps but on a significantly larger dataset than AS or AC - the level of

warm-up steps will therefore be tuned over our experiments.

- **Epochs** - It is important to ensure training periods are long enough to ensure that models have converged. Using validation loss as a guide, we can halt training once the validation loss no longer shows signs of further decreasing (by halting training after a number of static periods, or periods of increasing loss). Early experiments here allows us to set an appropriate number of epochs for further hyperparameter tuning, or for full fine-tuning on our set of optimised parameters.

- **Batch size** - Batch size determines the number of data samples fed into our fine-tuning loop at each training step. Selecting the optimal batch size involves striking a balance between memory/time requirements and training stability/convergence. Smaller batches demand less memory but result in longer training durations. They might also lead to noisier gradient updates, which can act as implicit regularisation and help the model avoid local minima. Conversely, larger batches can yield more accurate gradient estimations and stable convergence, but with potential overfitting. The noise in smaller batches arises from the increased variability between them, causing potentially erratic gradient updates.

  Batching methods range from stochastic gradient descent (a single sample per iteration), to mini-batching (a subset of data per iteration), to batch gradient descent (using the entire dataset per iteration). Generally, smaller batches might reduce memory overhead while providing a regularisation effect but could lead to extended convergence times due to erratic updates. Larger batches might offer more stable convergence and improved gradient accuracy but could result in memory strain and miss out on the regularisation effects of smaller batches.

  - **Gradient accumulation** is a technique which can enable training on larger batch sizes with smaller memory requirements. This parameter sets the number of batches which undergo a forward and backward pass before weights are updated. This means that with a batch size of 1 and a gradient accumulation of 4, our model will process each sample and add these updates to a gradient buffer before updating the weights on the fourth iteration. This results in lower memory requirements, but less overall weight updates.

Chapter 4

Exploratory experiments

Contents

In this experimental research, our procedure for fine-tuning Stable Diffusion (SD) follows an iterative procedure where we add complexity to our experiments with the aim of achieving model configurations with usable audio generation. This chapter details our experiments, alongside intermediate results and conclusions.

Similar to existing literature, full evaluation will require the generation of metrics - such as Inception (IS) and FID (Fréchet inception distance) scores - but as these scores add additional computational overhead we elect to only use these metrics to evaluate our final tuned model configurations.

An overview of our experiment phases are as follows, this section will detail our empirical experiments in chronological order:

- **Testing the feasibility of LoRA (Low-rank Adaptation) fine-tuning with a generic dataset:** this phase has the goal of confirming that our LoRA training script and hardware are sufficient for fine-tuning SD.

- **Fine-tuning using chosen natural audio-classes from AudioSet:** we start with a single class before scaling up to include more classes to determine early feasibility of spectrogram generation.
- **Fine-tuning with subset of AudioCaps:** we then scale up our experiments to a subset of AudioCaps, a dataset with more descriptive prompts similar to prompts used for downstream generation.
- **Prompt and data adjustment:** we then test the necessity of needing to include a textual signifier in our data and prompts to evaluate the effectiveness of our fine-tuning with it.
- **Scaling experiments to include full AudioCaps dataset**: we then scale our experiments, training with our full dataset.

Unless otherwise stated, our experiment configuration uses an AdamW optimiser (with β_1 at 0.9, β_2 at 0.999, ϵ at $1e^{-8}$ and weight decay of 0.01), a cosine warm-up scheduler, a batch size of 1 and 4 gradient accumulation steps.

4.1 Testing LoRA with Pokémon BLIP caption dataset

The initial step of our experimental process focuses on using LoRA to fine-tune using a pre-existing dataset evidenced to work. The Pokémon BLIP captions dataset [115] is a dataset of 833 images (of fictional creatures from the popular media franchise Pokémon) with corresponding natural text descriptions (examples in . The objective of this fine-tuning is to quickly test our pre-existing LoRA fine-tuning script on our available hardware. This dataset is given as an example within the Hugging Face Diffusers package (where we acquired our fine-tuning script) due to its use as a SD fine-tuning tutorial dataset elsewhere [116]. Its prior use as an SD fine-tuning tutorial dataset emphasises its appropriateness for this preliminary test.

Following default parameters from Hugging Face's Diffusers documentation [24], we fine-tune SD1.5 for **18 epochs**, at learning rate **0.0001** ($1e^{-4}$). Using the prompt "a pokemon with blue eyes", we can see the effect of LoRA fine-tuning over the training process in Figure D.2. From this initial training run, we confirm that our LoRA fine-tuning script can be run on our available hardware, and that there is some (qualitative) evidence of our model learning features from our example dataset. With these confirmations, we moved on to our primary research objective of spectrograms. Generation outputs and data sample examples can be found in Appendix D.

4.2 Fine-tuning with selected AudioSet classes

4.2.1 Bird song

The next phase of experiments involves fine-tuning our model on a subset of AudioSet (AS). We start with a simple configuration of data and parameters, before scaling to include more classes and finer hyperparameter tuning in later phases.

The first set of experiments focuses on fine-tuning using the 'Bird song' class of samples from AS. This subset consisted of 97 samples of various bird song and vocalisations, passed through our pre-processing pipeline detailed in section 3.2.3, each with a corresponding caption of "a spectrogram of bird song". For these experiments, our validation set is formed by randomly separating 20% of samples (15 samples) from our training set. Figure 4.1 shows many spectrograms lacking higher-end frequencies, resulting in blank spaces at the top of the image.

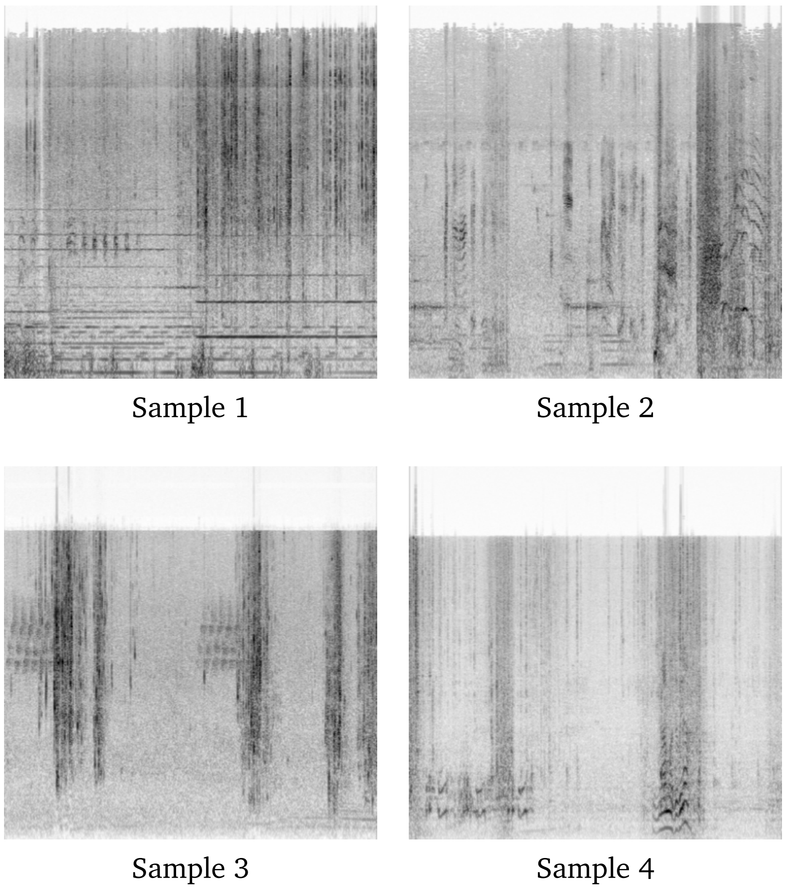


Figure 4.1: Four spectrogram samples from AS bird-song training dataset

We initially ran an experiment for **15,000 steps (155 epochs)** at learning rate $\mathbf{1e^{-4}}$, mirroring our configuration for our testing on the Pokémon BLIP dataset. During these experiments, we generated four inference images at every epoch of training, as well as logged training and validation loss per epoch.

As seen in Figure 4.2, our training and validation loss curves are erratic, potentially due to a combination of the complexity of our model, noise in the training data, fewer gradient updates from gradient accumulation and our adaptive learning rates from our AdamW optimiser. Additionally, the training process of diffusion models involve denoising samples which have had gaussian noise applied to them; this noise inherent in the training process may also be contributing to noise present in the loss curves. Due to these reasons, our curves will be smoothed to control for this noise - with exponentially moving average using a decay rate of 0.99 - to illustrate our loss trends.

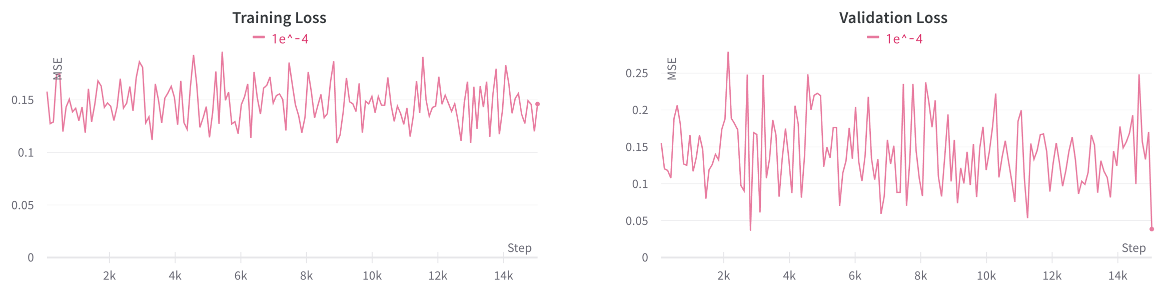


Figure 4.2: Unsmoothed loss curves from fine-tuning with AS bird-song - LR $1e^{-4}$ (97 training samples).

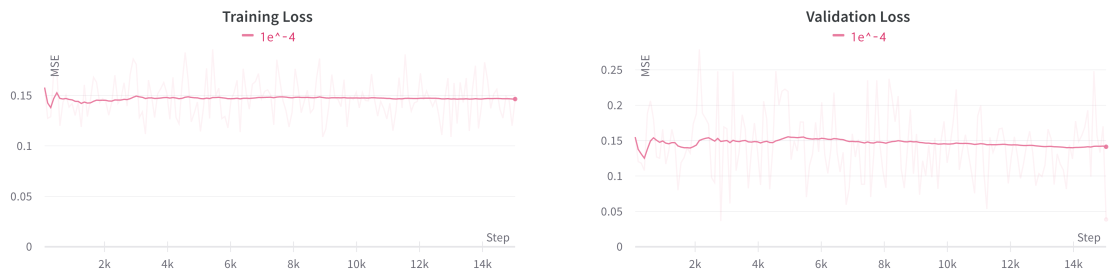


Figure 4.3: Smoothed loss curves from fine-tuning with AS bird-song - LR $1e^{-4}$ (97 training samples). A faint underlay of the unsmoothed curve can be seen in the background.

As seen from our smoothed curves, our training and validation losses fall to a local minimum at epoch 3, then increase before slowly decreasing over the rest of the training; looking at the unsmoothed curve, we also reach a global minimum in our validation loss at the end of training.

Our loss metric is MSE, which evaluates pixel differences from our inference data and our training/validation datasets. This metric can be used to evaluate how our model is learning but can not necessarily be assumed to have a direct link with the quality of our final audio output due to the nature of spectrograms. Spectrograms are highly sensitive to variations and two spectrograms with low MSE may sound different due to this. This is because MSE treats all differences equally, which contradicts with human non-linear perception of hearing, with certain frequencies being more perceptual than other frequencies. We include loss curves in a holistic evaluation of performance, with greater weight given to subjective assessment of audio quality. As previously mentioned, quantitative metrics will be generated after our empirical experiments.

Converting these spectrograms to waveform produces audio which contains some evidence of replicating bird song (such as non-rhythmic chirps) but is interspersed with loud sounds of distortion and a phaser effect (with fast shifts in phase over time creating a warbling sound). While this early experiment is promising, the results still contain significant levels of non-domain and non-natural sound - which makes this audio unsuitable to be used as natural sounding bird-song. Inference outputs can be found in Appendix E.

We also experiment with reducing the learning rate to $\mathbf{1e^{-5}}$, which results in similar quality of generated audio. Despite these results, these experiments illustrate SD's capability to learn spectrograms through fine-tuning - particular evidence of this can be seen from our generated spectrograms, showing a similar higher-frequency gap which was frequently present in our training data.

This early set of experiments demonstrates that:

- Loss curves can be used to evaluate how our model is learning to generate spectrograms over time, but not necessarily used to evaluate the quality of our final converted waveforms.
- Our model is capable of producing audio which contains some features of our target class.

4.2.2 Expanding fine-tuning to five classes

As seen in Section 4.1, the example dataset we use consisted of multiple images, each with a separate prompt. The structure of our AS dataset differs from the example dataset in that it contains multiple images sharing the prompt A spectrogram of bird song", as opposed to each image having its unique prompt. To remove the possibility of our limited prompt causing a failure to fully learn, we expand our experiments to five classes: Train horn, Snare drum, Cheering, Acoustic guitar, Dog bark. We selected five classes with distinct and recognisable characteristics to aid perceptual evaluation.

A potential issue with our bird-song experiments may arise from data quality - our small dataset consisted of vocalisations from different birds which may have resulted in our model failing to learn any common features across our data. The goal of this phase in our experiments was to expand our training data and eliminate potential biases associated with using a single class label. Data examples and inference outputs can be found in Appendix E.

Class	Samples	% of five-class dataset
Dog bark	55	18.1%
Cheering	59	19.4%
Acoustic guitar	51	16.8%
Snare drum	84	27.6%
Train horn	55	18.1%
Total	304	

Table 4.1: Sample size of classes in our five-class AS dataset.

This five-class dataset contained 304 samples, with a train/test/val split of 80/10/10. For this dataset, we ran 7 experiments, each with different configurations of learning rates and epochs. At the end of every epoch we generated an image using the prompt "a spectrogram of [class name]". For the first three experiments, we trained for **15,000 steps (61 epochs)**, with **1,000 warm-up steps** at three learning rates $\mathbf{1e^{-3}}$, $\mathbf{1e^{-4}}$, $\mathbf{1e^{-5}}$.

The outputs produced by our inference prompts vary in quality. Our model had some success at replicating dog barks and cheering, but struggled to replicate train horns, acoustic guitars and snare drums. For barks and cheering, we found learning rates $\mathbf{1e^{-3}}$ and $\mathbf{1e^{-4}}$ produce the highest quality audio which could be perceptually determined as the target class. Despite the model's failure to reproduce these latter three classes with these parameters, some features of these classes are present, such as rhythm within snare drum audio. Regardless, most inference output contained noticeable elements of distortion and warbling.

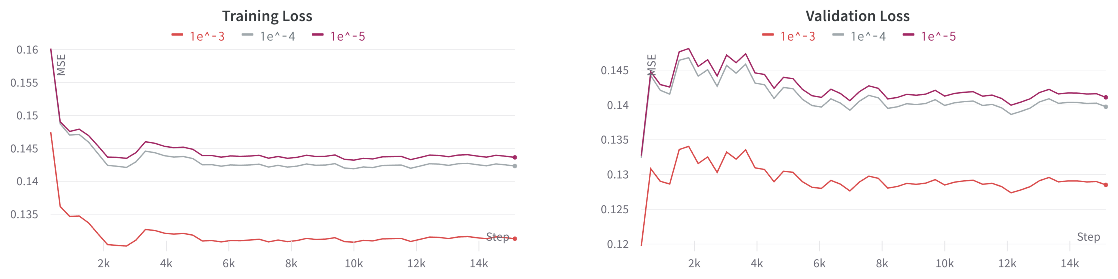


Figure 4.4: Training and validation smoothed MSE loss curves for AS 5-class dataset at 3 different learning rates.

Inspecting our (smoothed) loss curves for these experiments, we can see a fall in our training loss for all three learning rate experiments, followed by a slight increase before settling at steady levels from epoch 15 onwards. These training loss curves in isolation are promising, but our validation loss curves show a sudden rise from a global minimum at epoch 1. This discrepancy between our loss metrics and spectrogram outputs is evident, especially when, at our minimum validation loss, our outputs significantly deviate from our target of producing usable spectrograms.

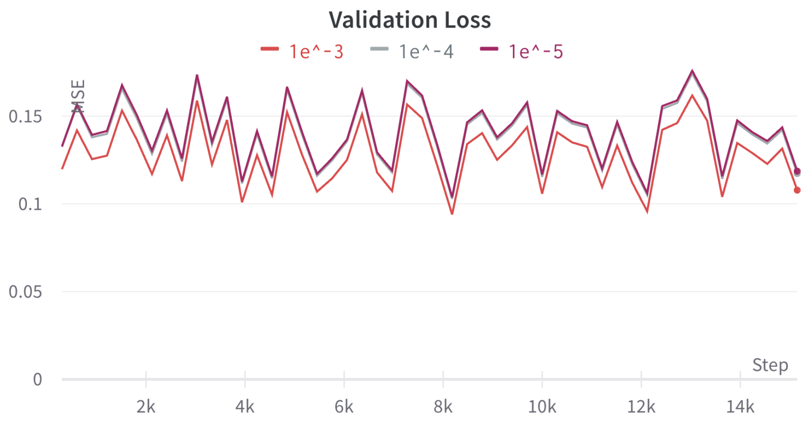


Figure 4.5: Unsmoothed validation loss for AS 5-class dataset at 3 different learning rates.

From these three experiments, learning rate $\mathbf{1e^{-3}}$ achieved the absolute lowest loss values between the three experiments. However, this behaviour may also be a result of a potentially unrepresentative validation set, with only 10 % of our limited subset used for validation. Alternatively, this validation curve behaviour may also arise from this smoothing - inspecting our unsmoothed curve shows increasingly volatile losses (with differences between epochs increasing); these experiments may therefore show potential signs of overfitting on our training set.

We also additionally tested learning rates $\mathbf{1e^{-5}}$, $\mathbf{1e^{-6}}$ for 33 % more epochs (**81 epochs**) to assess whether our experiments at these lower learning rates potentially did not have enough training time to converge (Figure 4.6). Our training losses for these two experiments are smoothed to reach similarly low levels to $\mathbf{1e^{-3}}$; the validation losses start high, before generally trending downwards to reach a similar level to $\mathbf{1e^{-3}}$ (although seemingly remaining above).

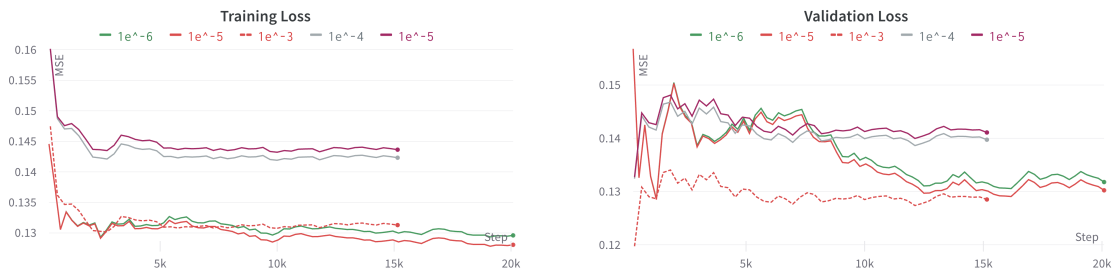


Figure 4.6: Training and validation smoothed MSE loss curves for AS 5-class dataset at 5 different learning rates.

When we examine the unsmoothed validation curve (Figure 4.7), we can see these smaller learning rates achieve lower absolute validation loss values - but with increased volatility across the curves (i.e. higher maximums and lower minimums). This shows that these lower learning rates may potentially result in more unstable training. This is further exemplified by our output audios being perceptually worse than our higher learning rate experiments - resulting in increased distortion and artifacts.

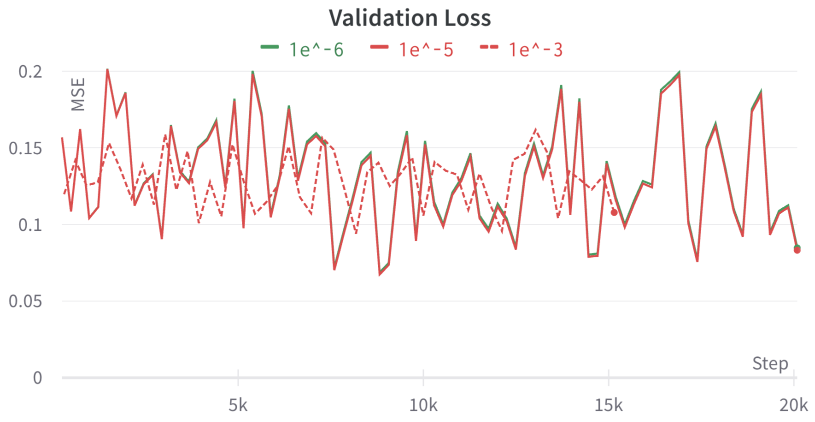


Figure 4.7: Unsmoothed validation loss for AS 5-class dataset at 3 different learning rates.

Our inference outputs for $\mathbf{1e^{-6}}$ show these artifacts, generating spectrograms which are quite distinct from our training data - with visual distortions such as colour and blurring (Figure 4.8). This may be a result of choosing a learning rate too low - and our training potentially becoming stuck in a local minimum. Although we achieved average lower validation losses, our generated audios have lower perceptual quality - this may imply that stability of training (and loss curves) may also be a useful metric to guide our further experiments.

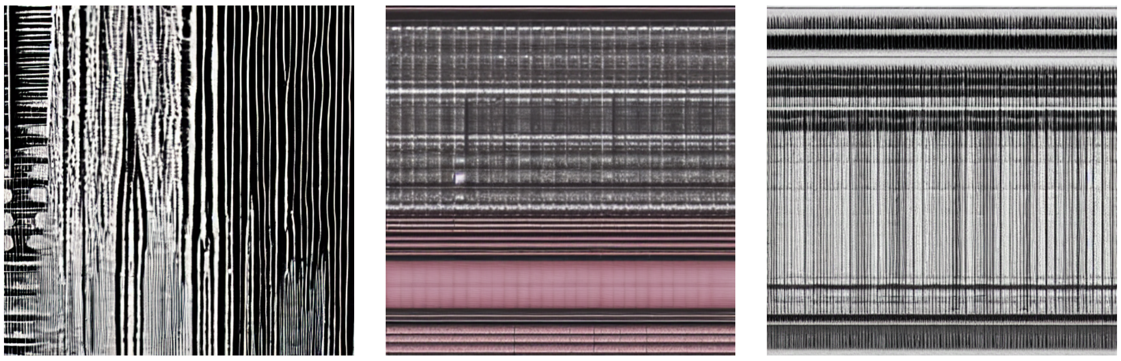


Figure 4.8: Selection of inference outputs from $\mathbf{1e^{-6}}$ demonstrating visual artifacts

Expanding our experiments to include these five classes shows us that:

- Our model is capable of producing some target sound classes after fine-tuning whilst failing to generate others. This could potentially be due to data quality issues across classes (due to AS sampling data of varying quality from YouTube).

- Learning rates of $\mathbf{1}e^{-\mathbf{3}}$ and $\mathbf{1}e^{-\mathbf{4}}$ are potentially promising starting points to explore further learning rates from.
- Absolute loss values continue to be a potentially confusing metric to guide our training - experiments with lowest achieved loss values do not generate higher quality audio samples. Stability of training may instead be a useful metric of how well our model can generate usable audio. However, our validation loss calculation may be affected by a potentially unrepresentative validation set.

4.3 AudioCaps

Fine-tuning on AudioSet involved attaching a simple prompt of "a spectrogram of [class]" to each sample in our dataset; we wanted to evaluate whether increasing the detail of our text prompt would result in improved output. Prompting SD often involves detailed natural language prompts which are tokenised using a CLIP text encoder. The usage of detailed prompts at inference, as well as the usage of detailed captions in our example dataset (used in Section 4.1) encouraged us to experiment with fine-tuning on AudioCaps (AC). AC uses the same underlying data source as AS, but instead provides detailed text descriptions of each sample. Previous audio generative research have also trained on AC (sometimes exclusively [32]), which motivated us further to begin experimenting with AC.

As an additional robustness check, we analyse the most common words across these captions (after removing stop-words such as 'and' & 'the')[1] - this reveals a weighting towards audio containing speech. Other natural descriptions include 'wind blow', 'birds chirp' and 'dog bark' (referred to as 'bow-wow' by AudioCaps). This bias aligns with AC's ontology (Figure A.1) which also presents speech as the most frequent classes; however, this bias may also result in our fine-tuned model failing to generalise to under-represented classes.

[1]List of stop-words used for filtering was obtained from Python's Natural Language Toolkit package [117]

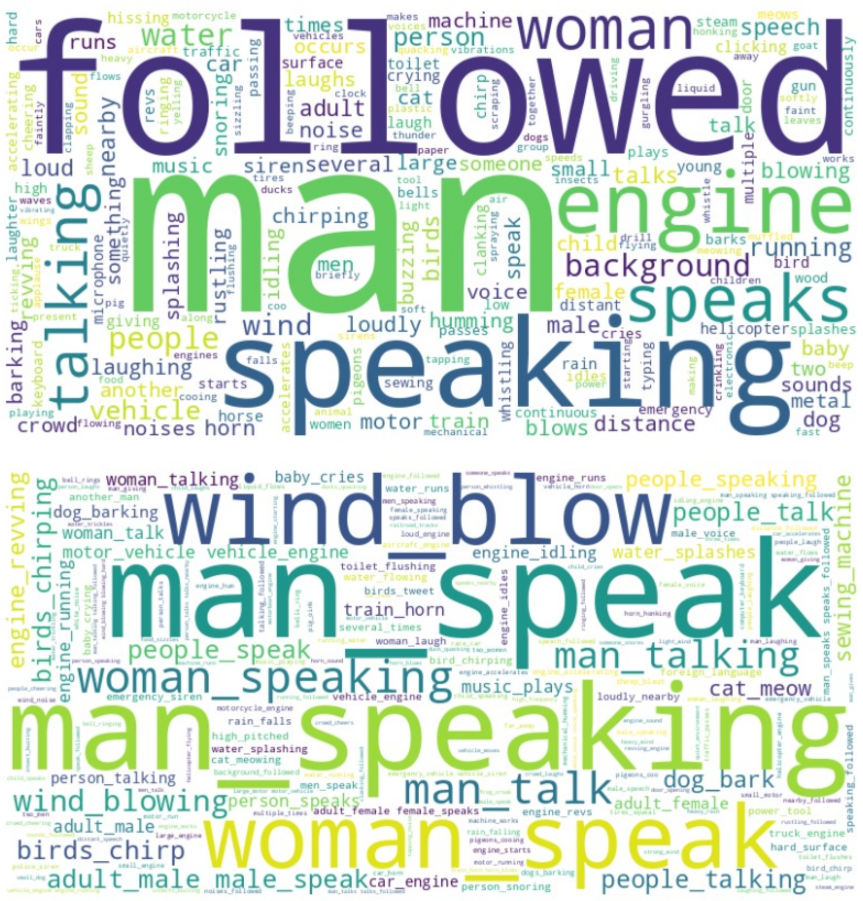


Figure 4.9: Word cloud of most common single words and bigrams (concurrent pair of words) in our AudioCaps dataset. Common stop-words are filtered from above.

4.3.1 Subset

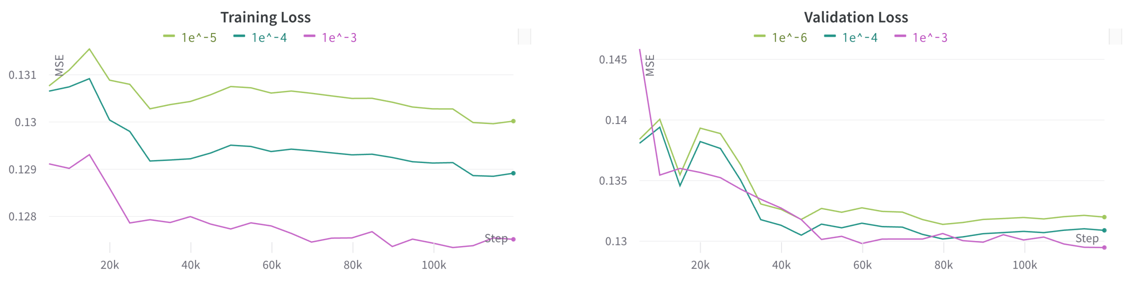
We begin by training on a subset of AC, randomly sampling 20 % of our total available AC data. We choose this subset for computational efficiency, aiming to run multiple shorter experiments before scaling up to our full dataset. Similar to our AS experiments, we processed our data by adding "a spectrogram of" before each caption. A total of **45** experiments were run on this subset with different parameter configurations, to evaluate the impact each of these parameters would have - with the aim of achieving high quality audio.

Throughout the experiments, we use a number of inference prompts randomly sampled from the validation set - initially using (each prefixed with "a spectrogram of"):

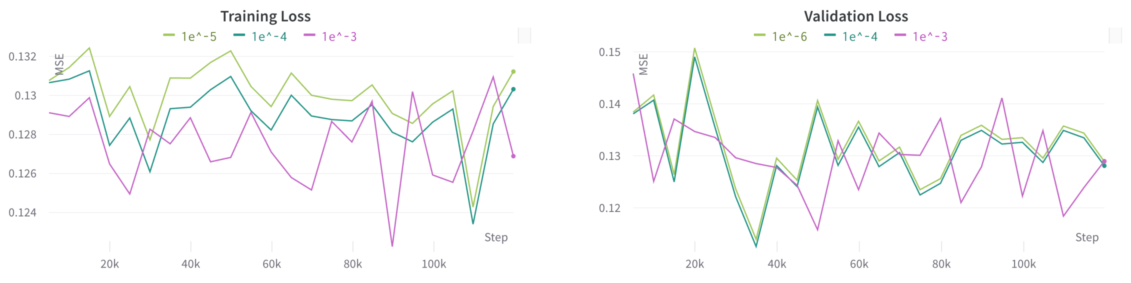
- "An animal hissing followed by a man mumbling then a pig oinking while birds chirp in the background"
- "Music plays and someone speaks before gunfire and an explosion occurs"
- "Someone is typing on a computer keyboard"

23 epochs

We initially run three experiments, testing our learning rates used with our AS experiments (excluding $1e^{-2}$ and $1e^{-6}$), training for 23 epochs with 20,000 warm-up steps. From our smoothed training curves, $1e^{-3}$ appears to constantly have a lower training loss than the other two rates while achieving a lower validation loss by the end of training. However, the unsmoothed training loss curve illustrates that these losses intersect, but with $1e^{-}3$ generally trending below the other two rates. Looking at the validation curves, $1e^{-4}$ and $1e^{-5}$ do achieve a lower absolute loss value than $1e^{-3}$ but has a higher absolute loss immediately before this drop - possibly alluding to increased volatility over training. Unfortunately, all generated audio is completely unrecognisable from any of the inference prompts - mostly producing distortions and noise.



Smoothed



Unsmoothed

Figure 4.10: Training and validation MSE loss curves for AC subset at 3 different learning rates for 23 epochs and 20,000 warm-up steps.

Following these three experiments, we test learning rates $3e^{-3}$ and $3e^{-4}$ (rates lying between $1e^{-3}$ and $1e^{-4}$). We adjust our warm-up values, lowering the former experiment to 16,000 rather than 20,000. For the former, the training loss falls before increasing slightly to a (potentially) stable level; this is accompanied with a validation loss with an almost inverse trend - implying potential overfitting.

For $3e^{-4}$, our validation loss falls initially, but then remains higher than initial loss for the rest of training; training loss increases to a maximum before steadily falling to meet the final loss value of $3e^{-3}$. This could potentially be a sign of overfitting - but as this model does eventually reach the training loss of $3e^{-3}$, it could instead signify that training at this learning rate requires additional time for our model to effectively learn. As with the previous three experiments, audio here is similarly unusable.

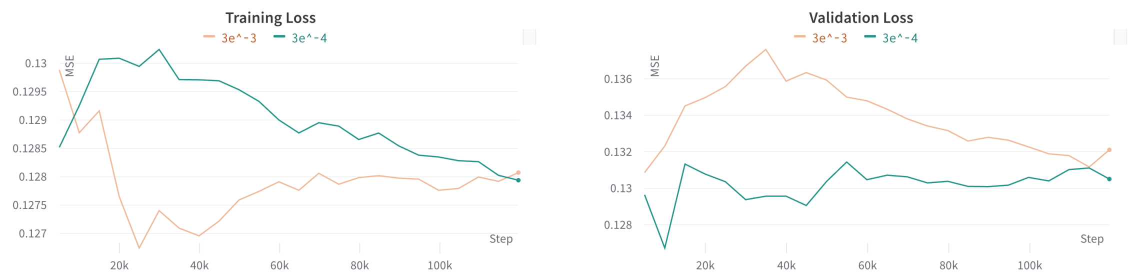


Figure 4.11: Training and validation MSE loss curves for AC subset at 2 different learning rates for 23 epochs.

From these experiments we learn:

- Our experiments for learning rate $3e^{-4}$ may have had insufficient time to converge, with training loss continuing to fall. However, the validation curve does show evidence of overfitting.
- We are unable to obtain audio anywhere close to usability - this is in contrast with our AS experiments, where our model was capable of generating audio matching the inference prompts when evaluated by perception. This may be due to the significantly fewer epochs used for this set of experiments.

To rule out the possibility of training times being too short - especially for smaller learning rates - we focused on increasing our number of epochs, whilst also further investigating the effect of warm-up.

50 epochs

After these 23-epoch experiments, we then focus on increasing training time - raising our number of epochs to 50, while adjusting our warm-up steps and learning rates. An overview of this next phase of experiments in tabular form can be found in Appendix C.

For our first three 50 epoch experiments, we tested $3e^{-4}$ and $3e^{-3}$ again, along with $1e^{-4}$ and $1e^{-3}$. Upon inspecting our loss curves, evidence of an exploding gradient becomes apparent for $3e^{-3}$. Inference outputs from this particular experiment also visually demonstrate this, with corrupted colourful spectrograms that are far from our target output (Figure 4.13).

From this set of experiments, we see it is possible for our fine-tuning to result in loss divergence. This also provides evidence of MSE's explanatory power in some situations - as the inference outputs align with our expectations that this rapid increase in training loss would cause. Our validation loss figures for $1e^{-4}$ and $3e^{-4}$

also trend below what we have previously seen - with our smoothed graphs mostly occupying loss values between 0.125 and 0.129 (in contrast to earlier experiments where smoothed validation loss rarely fell below 0.13).

Our inference outputs from these two experiments have also slightly improved, with audible elements of bird song present in our first inference prompt, as well as some features of keyboard typing in our third inference prompt. However, these models struggle with outputting audio matching all elements of these longer prompts - failing to generate some aspects of the prompt (e.g. a pig oinking) and only generating parts of the prompt in some cases (e.g. birds chirp). This output is still an improvement on our 23 epoch experiments however.

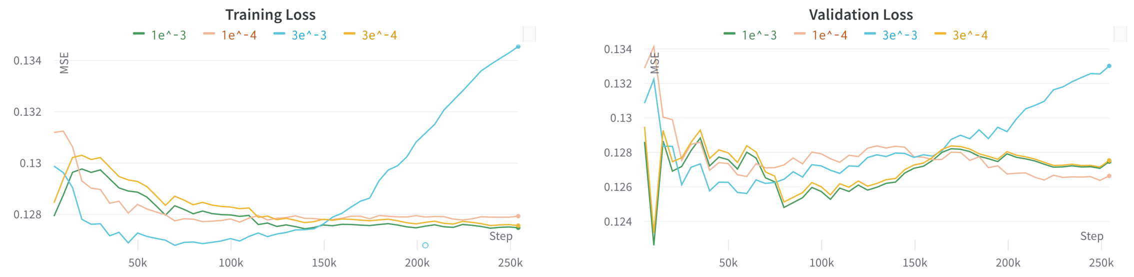


Figure 4.12: Training and validation smoothed MSE loss curves for AC subset at 4 different learning rates for 50 epochs and 16,000 warm-up steps.

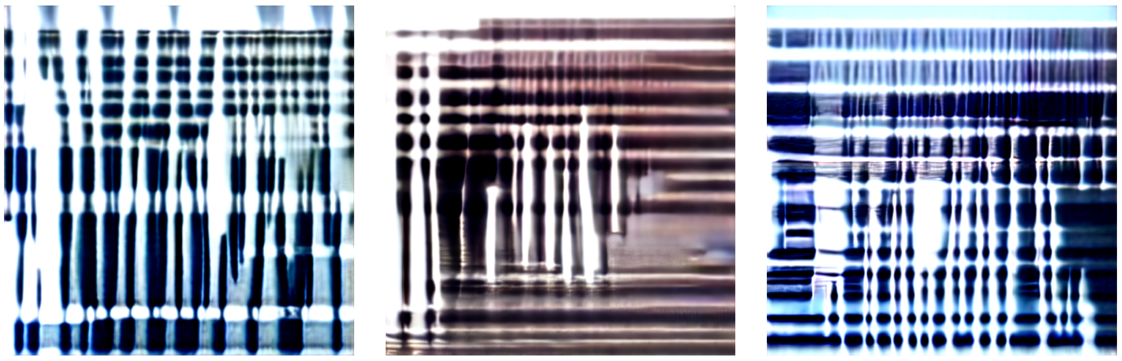


Figure 4.13: Inference outputs from $3e^{-3}$, 50 epochs and 16,000 warm-up steps demonstrating evidence of exploding gradient.

To compare performance on our training data, we also add an additional inference prompt: “a spectrogram of A horn honking followed by a man laughing then talking and plastic clanking on a hard surface” - this prompt was randomly sampled from our training set. This prompt enables us to identify how well our model is learning on our training data, as well as how well it can be applied to our validation set in comparison.

We then focus on tuning the number of warm-up steps across a number of learning rates - testing a range of warm-up steps (30,000, 60,000, 32,000 and 28,000). Running three experiments at $1e^{-3}$ with different warm-up steps show a difference in our loss curves. Our experiment with 16,000 warm-up steps shows a spike in training loss at the start of training, accompanied by a fall in our validation curve which then immediately increases to our initial validation loss; we also see a similar spike in validation loss with 30,000 warm-up steps. Our experiment with 32,000 warm-up steps results in a less volatile validation curve, but loss values on both curves is consistently higher than our other two experiments. This may suggest that the number of warm-up steps affects the stability of early training for the first few epochs.

he next set of experiments was designed to investigate the impact that warm-up would have for $3e^{-3}$ due to our previous experiment with this rate resulting in exploding gradient. Increasing our warm-up steps should result in increased stability of training, allowing our network more time to adapt its weights and biases with a gradual (and less aggressive) learning rate. This allows for a more controlled exploration of the loss landscape, and potentially avoiding regions where the gradient might explode.

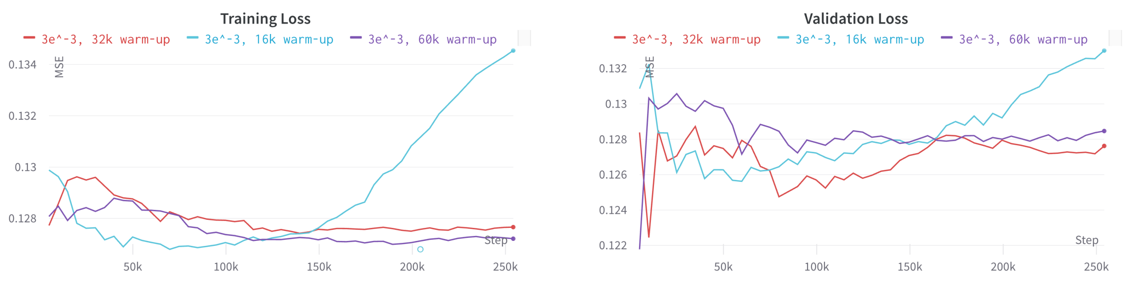


Figure 4.14: Training and validation MSE loss curves for AC subset at $3e^{-3}$ for 50 epochs and varying warm-up steps.

Looking at our loss curves for these three warm-up steps; we see that increasing the number of warm-up steps has avoided the exploding gradient problem previously encountered. This exploding gradient may therefore be a result of insufficient stabilisation caused by a shorter warm-up period. 60,000 warm-up steps may also be presenting signs of overfitting compared to 32,000 steps - with higher validation loss and lower training loss values. From these results, we elected to focus on 32,000 as an anchor for future experiments.

Testing these warm-up steps with $1e^{-4}$ resulted in strictly higher training and validation loss curves and no increase in audio fidelity.

Our next set of experiments explored learning rates $5e^{-4}$ & $7e^{-4}$ at two warm-up steps: 32,000, and a slightly lower value of 28,000. We chose 32,000 due to the results of our $3e^{-3}$ experiments, as well as this lower value to test the sensitivity of tuning our warm-up by a small amount. From our loss curves, our experiments with 28,000 warm-up steps achieved lower validation and training loss compared to 32,000. Additionally, inspection of our audio clips also demonstrated increased fidelity in audio for these learning rates over previous experiments.

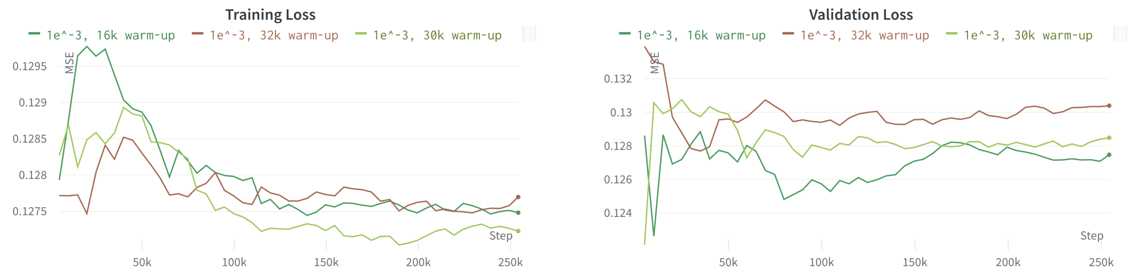


Figure 4.15: Training and validation MSE loss curves for AC subset at $1e^{-4}$ for 50 epochs and varying warm-up steps.

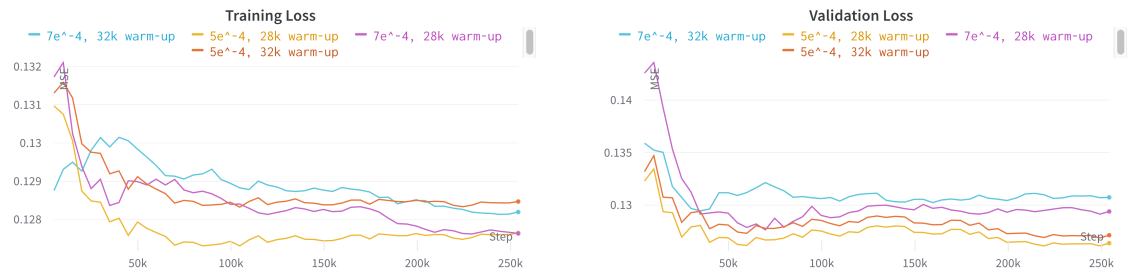


Figure 4.16: Training and validation MSE loss curves for AC subset at $5e^{-4}$ and $7e^{-4}$ for 50 epochs and at 28,000 and 32,000 warm-up steps.

From our set of 50 epoch experiments, we found:

- It is possible to experience exploding gradient when fine-tuning SD. Choosing a learning rate too high with too few warm-up steps can lead to our model generating increasingly unstable spectrograms filled with noise.
- Warm-up steps can affect our loss trends, sometimes resulting in trends with one warm-up configuration strictly greater than another warm-up configuration. Choosing too high of warm-up steps could potentially result in overfitting, or worse validation loss.
- Increasing the number of epochs can lead to lower validation loss values - especially compared to our 23 epoch experiments. This could suggest that training further may still lead to further decreases in loss - despite seemingly stabilising on our smoothed curves.
- From our continued experiments, the most promising set of parameters to pursue further - based on loss curves and perceptual evaluation - is learning rates $5e^{-4}$ & $7e^{-4}$, with warm-up steps near 32,000. These will be the starting point for one more phase of increasing our training time before expanding to the total dataset.

75 epochs

For this set of experiments, we increase our training time further - this time to 75 epochs. We also add additional validation prompts extracted from our training and validation sets as we found that our models were more capable of generating audio from simple prompts, rather than the descriptive prompts used in AC. To test this theory, we added in additional simple validation prompts to our training pipeline. Our validation prompts can be found in Appendix F

We choose our learning rates and warm-up steps following the results of our previous phase of experiments. A summary of our experiment parameters can be found in Appendix C.

Our loss curves (Figure 4.17 show that 28,000 warm-up steps are an improvement over 32,000 warm-up steps in terms of minimising both training and validation losses. The larger of the two learning rates $7e^{-4}$ also results in lower losses compared to $5e^{-4}$. However, these loss values are not wildly dissimilar to values obtained in previous experiment phases.

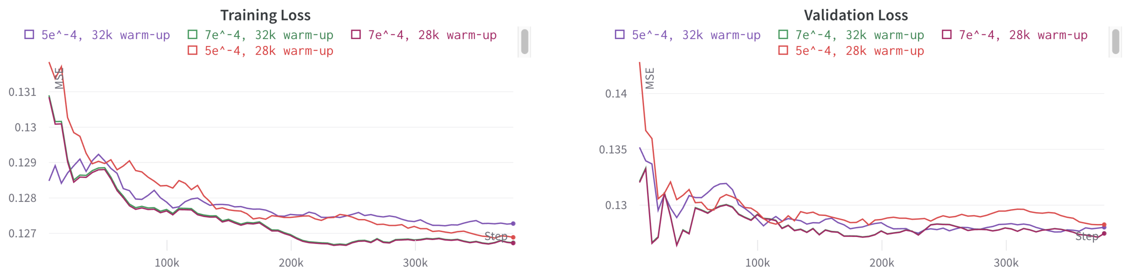


Figure 4.17: Training and validation MSE loss curves for AC subset at $5e^{-4}$ & $7e^{-4}$ for 75 epochs at 28,000 and 32,000 warm-p steps

The generated audio outputs yielded more promising results compared to previous experiments:

- For our simpler inference prompts, such as "a spectrogram of [someone typing on a keyboard/train horn/a dog barking]", our model generates recognisable audio matching the prompt.

- For more complex prompts, our model struggles to generate audio matching the prompt - generating noise and distortion in some cases. However, inference for prompts containing simple words and concepts (such as "birds chirp" and "a horn honking"), was capable of generating audio which contained features related to these prompts.

  For example, clear and recognisable bird chirping could be found in inference for "a spectrogram of An animal hissing followed by a man mumbling then a pig oinking while birds chirp in the background" whereas a train horn can be heard in "a spectrogram of A horn honking followed by a man laughing then talking and plastic clanking on a hard surface" at the start of our generated audios. This confirms that our model is better suited for generating simpler prompts but also that our model is capable of understanding temporal dimensions in our prompts (e.g. generating a horn honking before other sounds - per the prompt).

Increasing our training length did result in increasingly usable audio clips - however, our model still fails to produce audio matching longer, more complex prompts. As we have focused on learning rates, epochs and warm-up steps to varying levels of success, we then decided to evaluate whether adjusting our AdamW optimiser parameters could unlock this functionality.

Adam tuning

We continue our experiments by examining how changes in our optimiser parameter values affect our training and results. We had previously used the default Adam parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$ [114]) with a weight decay value of 0.01 (given as default in the Diffusers package [24]). These parameter values have been found to work well for many ML problems [114], so the focus of this optimisation is to perform a local search over values not too dissimilar to these defaults.

We chose to focus on β_2 to stabilise our gradient updates, aiming to make our optimiser more sensitive to recent gradients by reducing this value; our training loss often has high variance, and there is evidence (from our earlier phases) that there may be a causal link between stability of training and quality of outputs. For weight decay, we chose to test values on either side of the default, to explore the effects of increasing or decreasing this regularisation.

For these experiments, we evaluated the effect of the following for 25 epochs for learning rates $5e^{-4}$ & $7e^{-4}$:

- Lowering β_2 to 0.999 from 0.99.
- Increasing weight decay from $1e^{-2}$ to $3e^{-2}$.
- Lowering weight decay from $1e^{-2}$ to $3e^{-3}$.
- Combining a β_2 of 0.99 with both of these weight decay values.

Loss curves from our $5e^{-4}$ learning rate experiments demonstrate that our lowest validation loss is attained with a weight decay of $3e^{-3}$ and a β_2 of 0.99. However, this is paired with a smoothed training loss which never decreases past the original value. This may be a sign of the increased regularisation effect from the increased decay value, as well as faster adaptation to new gradients from the lowered β_2 value. However, keeping our weight decay at default while lowering β_2 resulted in a final lower validation loss, as well as lower training loss throughout training. However, we do not experience any noticeable increases in quality of our generated audio.

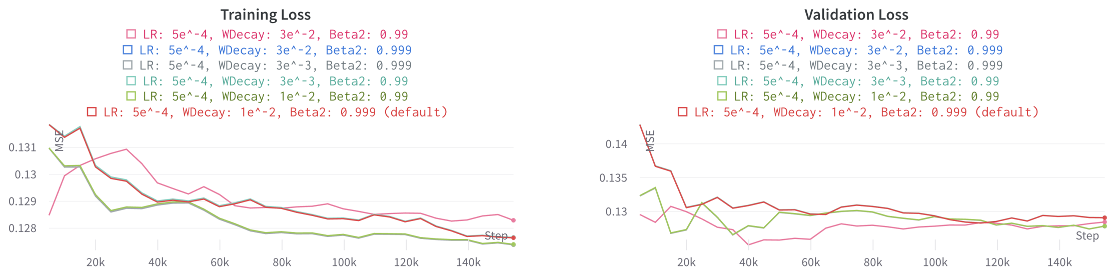


Figure 4.18: Training and validation MSE loss curves for AC subset at $5e^{-4}$ for 25 epochs, 28,000 with varying optimiser configurations

In our experiments with a learning rate of $7e^{-4}$, these parameter adjustments did not outperform our default parameters; this is coupled with no noticeable increases in quality of our generated audio. For both sets of these experiments, we do reach our lowest validation losses across all previous experiments, but inference at this epoch produces warbled and distorted audio - far from the promising results from our 75 epoch experiments.

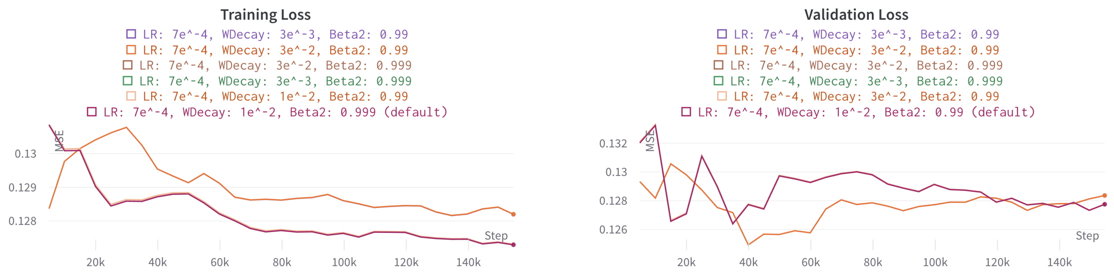


Figure 4.19: Training and validation MSE loss curves for AC subset at $7e^{-4}$ for 25 epochs, 28,000 with varying optimiser configurations

These experiments fail to produce evidence that changing our optimiser parameters improves on our default parameter values. Our lowest validation loss achieved is also not coupled with an improvement on generative output. This implies that default values for our parameters may be most suited for this task (without performing further search over the wider parameter space).

Subset observations

This empirical exploration of our subset yielded the following observations::

- Insufficient training time can negatively impact the quality of generated audio. Longer training times result in improved output, even if our loss curves appear to stabilise early in training.
- Selecting and optimising for appropriate learning rates can have a positive effect on generated audio quality. This parameter had the greatest impact on

improving our generated audio quality (after increasing training time). Further experiments could focus on granular fine-tuning for learning rate.

- Too many warm-up steps may possibly contribute to overfitting, and too few may result in unstable gradient updates (possibly leading to exploding gradient).
- Adjusting weight decay and β_2 optimiser parameter values seems to have no positive effect on generated audio quality.
- Our model struggles to generate audios of complex prompts and performs better when prompted with simpler and shorter prompts. This may be a result of missing classes in our dataset.

From these experiments, the most successful configuration for achieving prompt-aligned audio fidelity included learning rates of $5e^{-4}$ and $7e^{-4}$, warm-up steps of 28,000 (20% of the total training steps in our 75-epoch experiments), and the default Adam optimiser values combined with extended training durations.

4.3.2 Prompt and data adjustment

The final set of adjustments to our experiment methodology involves removing the phrase “a spectrogram of” from both the data samples and the inference prompts. Initially, each sample in our training dataset included this three-word prefix to ensure we could prompt our model with a signifier to enforce spectrogram generation. However, we hypothesised that this consistent prefix across all samples might hinder the model’s ability to generalise, potentially causing it to overfit to this repeated signifier. We aim to rule out such limitations and observe if our model could still generate spectrograms without this prefix.

For these experiments, we update our list of validation prompts to include another simpler prompt, while removing this signifier from all prompts (list can be found in Appendix F).

Upon removing this signifier, we immediately observed clear visual evidence that our model continues to generate spectrograms, underscoring the effectiveness of our fine-tuning (Figure 4.20). This indicates that the inclusion of this signifier might be unnecessary, given that our fine-tuning procedure alone seems sufficient for the model to generate spectrograms.

For our subsequent experiments, we opted to retain these adjustments to our data and prompts.

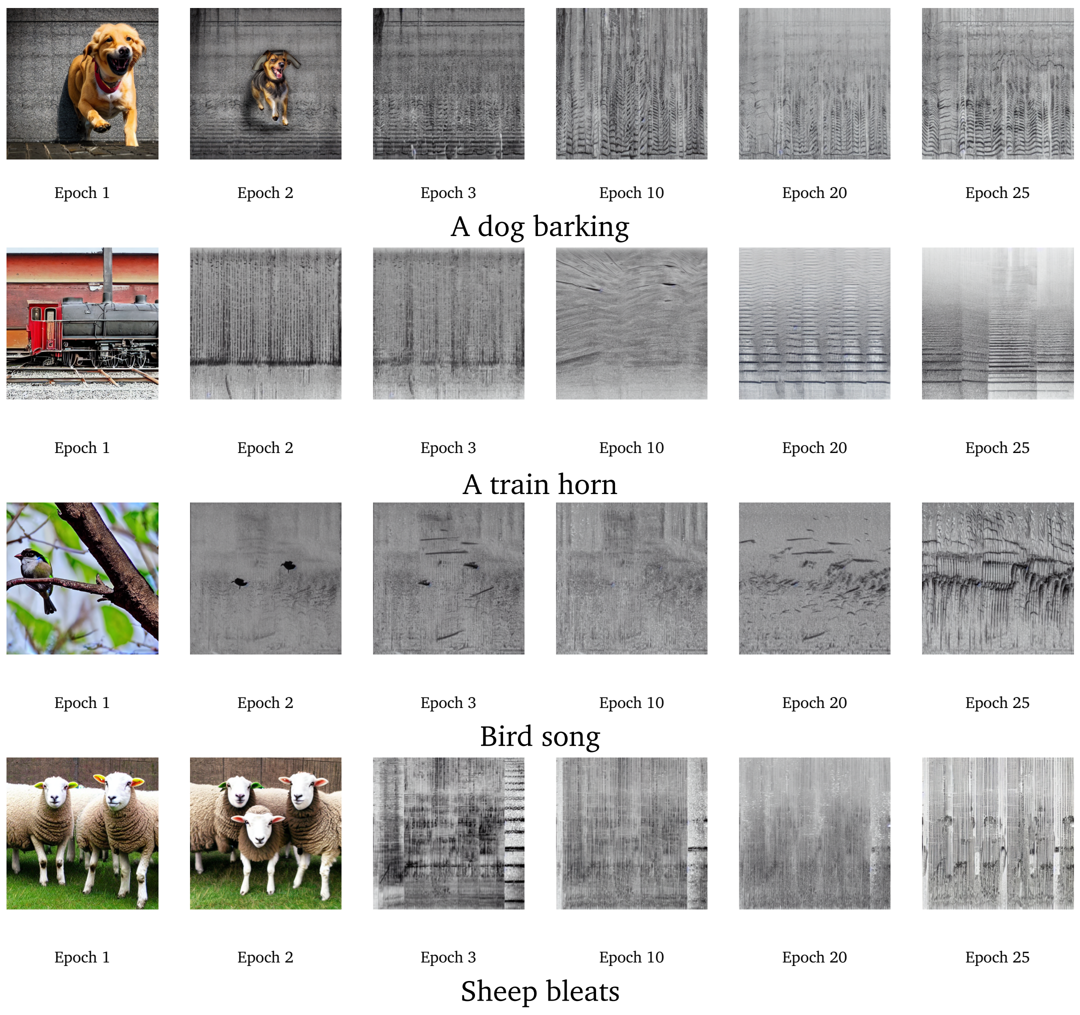


Figure 4.20: Inference outputs from 25 epochs of fine-tuning on signifier-adjusted dataset for different inference prompts, at different epochs through training - demonstrating evidence of spectrogram output.

4.3.3 Full dataset

After these experiments, we scale up experiments to our full dataset using learning rates $5e^{-4}$ & $7e^{-4}$, warm-up steps of 28,000 and our default AdamW parameters. These configurations were chosen based on promising results from our earlier experiments.

From our loss curves (Figure 4.21), we can see steadily decreasing training loss over time. However, our validation loss exhibits more variability: it decreases early in training, then increases, and eventually stabilises at a value higher than our initial loss. However, looking at our unsmoothed curve (Figure 4.22), we can see these volatile loss updates (including evidence of our loss successfully decreasing at multiple points throughout training). These validation loss fluctuations may arise due to gaps in our dataset causing a mismatch in data distribution between our training and validation sets - possibly resulting in our validation set failing to be fully representative of our training data. Comparing between these two learning rates, our loss curves are very similar for both experiments.

We elect to use these two models for our final quantitative evaluation, as detailed in Section 5.2.

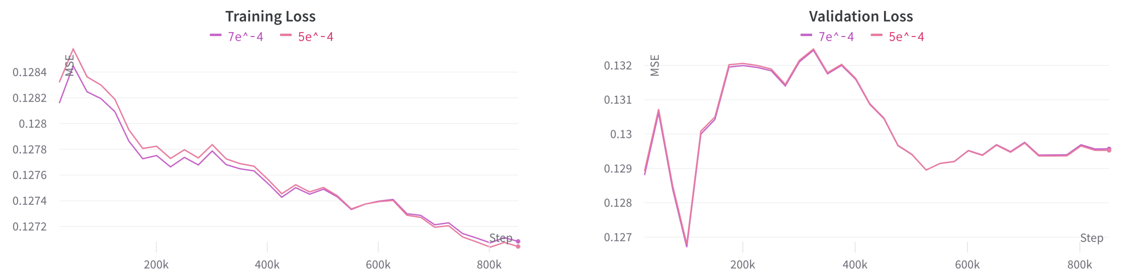


Figure 4.21: Training and validation MSE loss curves for AC dataset at learning rates $5e^{-4}$ and $7e^{-4}$ for 75 epochs and 28,000 warm-up steps.

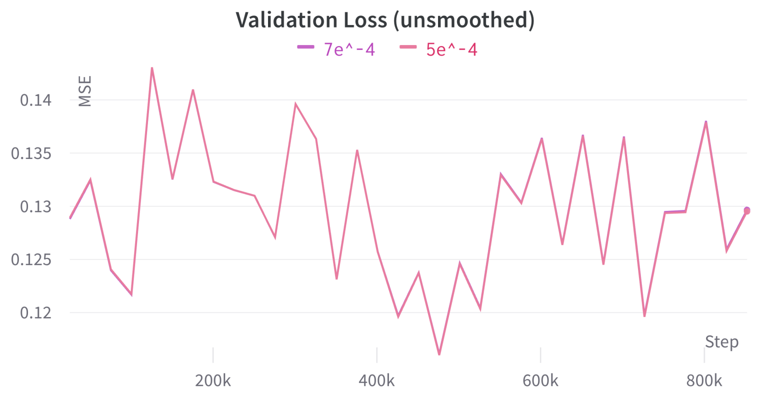


Figure 4.22: Unsmoothed validation MSE loss curves for AC dataset at learning rates $5e^{-4}$ and $7e^{-4}$ for 75 epochs and 28,000 warm-up steps.

4.4 Summary of experiments

In this chapter, we embarked on a comprehensive exploration of various experimental configurations to optimise the performance of our model in generating audio spectrograms. Here is a synthesis of our findings:

- Our model was successful at generating audio for certain classes, but unsuccessful with others. This may be solved from a further scaling up of our training dataset to improve generalisability.

- Our model also had more success at generating audio following simpler prompts - possibly due to low generalisability. However, our model did show some capability of discerning temporal dimensions in prompts - generating sounds in the order they were mentioned.

- Our model does not require a signifier - or keyword - added in training data or as part of inference prompts. Our fine-tuning alone is sufficient for our model to generate spectrograms without being specifically prompted to.
- We discovered that our loss curves - while indicative of the model's learning progression - are not necessarily fully indicative of the final audio quality. This can be seen from comparing inference outputs at validation loss minimums, to the end of training - epochs with lower validation losses, do not necessarily correspond to greater audio generation. However, this validation loss instability may have arisen from gaps in our dataset leading to a mismatch in distributions between our training and validation sets.
- Absolute loss values were not always indicative of audio quality. Instead, training stability might be a more reliable metric.
- Increasing training time has a direct correlation with improved generative power, with our 75 epoch experiments outperforming our 25 and 50 epoch experiments when evaluated by perceptual audio quality.
- Fine-tuning learning rate led to improvements in audio quality, however selection of appropriate learning rate is still necessary due to evidence of training experiencing exploding gradient.
- Choice of amount of warm-up steps influenced our training - with too few resulting in unstable gradient updates, and too many potentially contributing to overfitting.
- Slight adjustments to our AdamW optimiser parameters did not lead to any improvements. This suggests that the default values are ideal for this fine-tuning, but further research could potentially explore this parameter space in more depth.

This chapter details a novel exploration into fine-tuning SD for generating non-music audio spectrograms. Through rigorous experimentation, we have identified both strengths and limitations of our model, shedding light on its capabilities and areas requiring further refinement. The findings underscore the importance of dataset quality, the nuanced relationship between loss metrics and perceptual outcomes, and the sensitivity of the model to training parameters. These insights not only provide a foundation for future research but also emphasises the need for meticulous parameter tuning and dataset curation to achieve optimal performance. The insights from this chapter are pivotal for future research, emphasising the importance of parameter tuning and dataset curation in enhancing the generative capabilities of SD for audio.

Chapter 5

Results and metrics

Contents

The evaluation of experimental success also depends on the generation of objective metrics. These metrics can provide an understanding of how successful our model is at producing high-quality prompt-aligned audio and its level of generalisability to varied prompts. Generating these metrics also allows us to compare to current state-of-the-art (SOTA) approaches and how well our best-performing Stable Diffusion (SD) configurations can be used as a text-to-audio (TTA) system. In this chapter, we detail which metrics we use and subsequent results.

5.1 Metric selection

A number of metrics have been used to evaluate the performance and quality of generated images, and it is important to choose appropriate metrics for our domain (spectrograms).

The first of these is CLIPScore [118]; CLIPScore uses CLIP (a model trained on image caption pairs) to generate captions of image input [78]. CLIPScore is a metric which evaluates the correlation between a generated caption for an image, and the actual caption (or prompt) of the image. In CLIP's embedding space, images and their associated captions are nearby, whereas unrelated images and captions are far apart - this means that we can evaluate generated images by comparing how similar the image and their corresponding captions are in this embedding space. However, our model is fine-tuned to generate spectrograms of differing audio content. A CLIP model (trained on images) will be unable to appropriately discern the distance between our captions (e.g. bird song) from our spectrogram image (e.g. audio of bird song). This makes CLIPScore an inappropriate metric to use for our evaluation.

Other metrics include Inception Score (IS) [119] and Fréchet Inception Distance (FID) [59] which are extensively used to evaluate generated image quality [52, 70, 60, 61, 62, 63, 64, 65, 66, 67, 74]. IS evaluates the distribution of generated images (by evaluating how well a separate network can classify generated images), and FID compares distributions of our generated image set, to a set of real images. IS is usually implemented using a Inception v3 model which is a CNN image classifier pre-trained on ImageNet [120], but recent research [31] also uses an audio classifier (PANNs) in place of Inception [121].

FID similarly uses an Inception model, but instead uses this to extract features from generated and real images and then calculates the difference in distribution between both sets of images. The FID score is a measure of how different these two sets are - with a lower score indicating that generated images are more similar to real images. This metric was also adapted for the audio domain by using a different model for its classifier (Fréchet Audio Distance (FAD) [91]); this has since been used to evaluate generated audio quality [31, 32]. To compare with current SOTA TTA generative models, we elect to use FID, IS and FAD for our evaluation metrics.

To evaluate, we iterate over captions in our test set to generate a set of test spectrograms using these captions as prompts[1]. We then convert these spectrograms to audio and use this as our generated dataset - with the test set acting as our ground truth. We then apply a PANN model to extract features from both of these sets to generate IS, FID & FAD scores.[2]

5.2 Results

We generate metrics of our final set of models using parameters identified as promising during our exploratory experimentation detailed in Section 4. These are models fine-tuned with learning rates $5e^{-4}$ and $7e^{-4}$, using 28,000 warm-up steps and default AdamW optimiser values for 75 epochs on our full AudioCaps dataset.

[1]The AudioCaps test set contains 5 captions for each sample, we randomly sample one of these 5 text captions following the methodology detailed by current SOTA TTA model AudioLDM [31].

[2]Evaluation code is provided by Liu et al. (2023) as part of the release of AudioLDM [31].

Model	FID ↓	IS ↑	FAD ↓
AudioLDM-S [31]	29.48	6.90	2.43
AudioLDM-L-Full [31]	**23.31**	**8.13**	1.96
Make-an-Audio [122]	-	-	2.66
Make-an-Audio 2 [90]	-	-	2.05
TANGO [32]	-	-	1.73
AudioLDM 2-AC [33]	-	-	1.67
AudioLDM 2-AC-Large [33]	-	-	**1.42**
SD for Audio Spectrograms with Learning Rate: $5e^{-4}$	66.89	3.84	14.26
SD for Audio Spectrograms with Learning Rate: $7e^{-4}$	68.38	3.66	9.65

Table 5.1: Quantitative metrics comparing our models to current SOTA TTA systems. Our models are trained for 75 epochs, with 28,000 warm-up steps at two different learning rates. SOTA metrics reproduced from [31, 33].

As seen in Table 5.1, our current model does not perform as well in our metrics compared to current SOTA. Despite these results, our research offers the following insights:

- SD is capable of producing usable prompt-guided audio, but may be significantly affected by the quality and completeness of its training dataset. From our experiments, our model was capable of producing audio for certain concepts (e.g., bird song) but incapable of replicating other concepts (e.g. drums, or chatter). This problem could be attributed to our incomplete AudioCaps dataset (due to many samples being removed from YouTube). Our quantitative results reflect the poor generalisability revealed in our experiments.

- While usable audio was achieved, a more complete hyperparameter search could lead to improved performance. Due to memory constraints, we were unable to conduct a full parameter sweep nor experiment with larger batch sizes. Current SOTA models are trained on up to eight NVIDIA A100 80GB GPUs - significantly more than our consumer-level hardware. Further research focusing on performance could leverage additional compute to further search the parameter space.

- Our primary goal for this research was to conduct the first exploration into the feasibility of using SD as a TTA system - potentially offering a paradigm shift for providing increased consumer accessibility to building their own TTA systems. SD is particularly suitable for consumer accessibility due to its open-source nature and large community.

Due to the limited scope of this research, we believe these metrics do not reflect optimal performance which may be achieved through further research and experimentation. We present these metrics as an initial benchmark for which further developments can build upon.

Chapter 6

Stable Diffusion Tools: Exploring Inpainting and Outpainting Techniques

Contents

A benefit of the Stable Diffusion (SD) ecosystem is the availability of existing tools and functionality either provided natively as part of SD, or by the community. This chapter will demonstrate how additional SD functionality could be used to enable further research, provide additional control over generated output, and expand consumer accessibility to developing their own text-to-audio (TTA) systems.

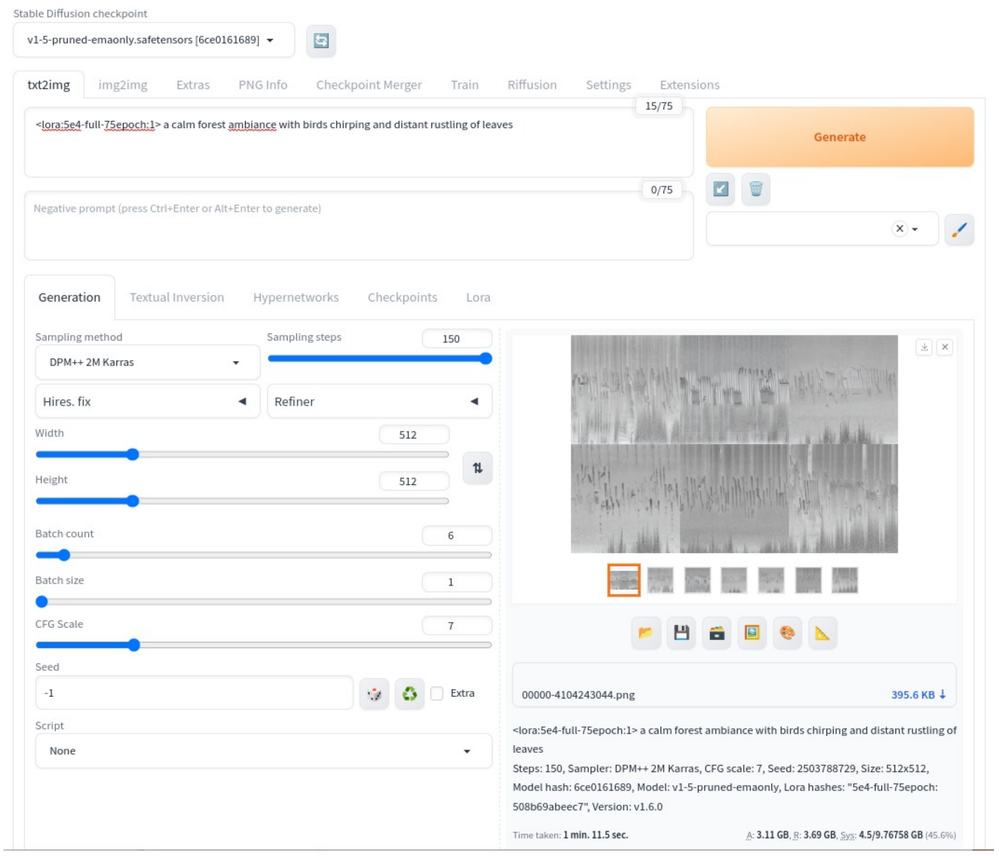
6.1 Automatic1111 Stable Diffusion WebUI

Automatic1111's Stable Diffusion WebUI is a browser interface developed to improve accessibility to training and inference using SD [87]. This software is very popular, with over 101,000 stars and 20,000 forks on its GitHub repository. This UI allows users to employ a model checkpoint (either provided by `AUTOMATIC1111` or supplied themselves) to easily generate images using original SD modes such as text-to-image, image-to-image as well as several additions such as inpainting (generating new images to fill in a specified mask), outpainting (extending the original image and inpainting the empty space), negative prompting (specifying what our model should avoid during generation) and upscaling (increasing the resolution of a specified image).

We can load our fine-tuned model weights (which are small due to our fine-tuning strategy LoRA) into this UI, and unlock this additional capability - not only for researchers, but also for consumers (without ML expertise) looking to generate audio themselves.

6.2 Demonstration

With minimal effort, we can use our trained model weights with the SD WebUI by first converting our checkpoints from .bin files (which our training script generates) to .safetensor files. These .safetensor files can be dragged into our WebUI installation directory - enabling easy usage of a number of SD tools and functionalities.



<lora:5e4-full-75epoch:1> a calm forest ambiance with birds chirping and distant rustling of leaves

<lora:5e4-full-75epoch:1> a calm forest ambiance with birds chirping and distant rustling of leaves
Steps: 150, Sampler: DPM++ 2M Karras, CFG scale: 7, Seed: 2503788729, Size: 512x512, Model hash: 6ce0161689, Model: v1-5-pruned-emaonly, Lora hashes: "5e4-full-75epoch: 508b69abeec7", Version: v1.6.0

Time taken: **1 min. 11.5 sec.** A: **3.11 GB**, R: **3.69 GB**, Sys: **4.5/9.76758 GB** (45.6%)

Figure 6.1: Demonstration of SD WebUI generating spectrograms using our fine-tuned model

This interface provides an easy way for a user to perform inference with our trained model - simply by entering a prompt and then pressing *Generate*. This interface also provides a number of options, such as batch count, batch size and the level of classifier-free guidance (i.e. how much the image generation follows the text prompt). The low size of our model checkpoints (3.3MB) and the usability of this UI, already lowers the barrier to entry for those looking to generate their own text-to-audio inferences.

Apart from its ease of use, this UI also allows us to apply SD-specific tools to our spectrogram generation. For example, we generate a 10-second audio clip following the prompt “a calm forest ambiance with birds chirping and distant rustling leaves”. We can then use this generated spectrogram and WebUI’s inpainting functionality to draw masks on our spectrogram to be inpainted over.

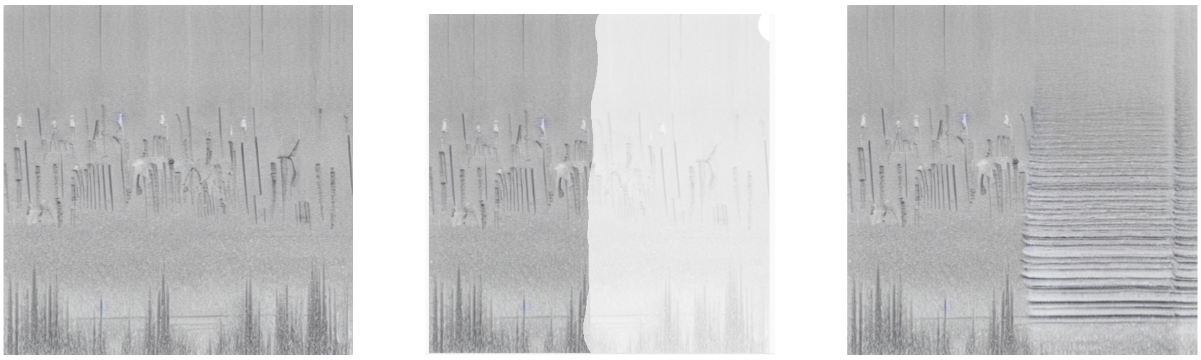

(a) Original image (b) Drawn mask (in light grey) (c) Inpainting result

Figure 6.2: Demonstration of inpainting ‘train horn’ over half of a spectrogram generated with prompt ‘a calm forest ambiance with birds chirping and distant rustling leaves’.

Inpainting half of our spectrogram (Figure 6.2) produces audio blending our original prompt with the inpainted ‘train horn’. Different mask placements modify temporal and frequency components, leading to varied audio blends. For instance, center inpainting merges train horn with forest ambiance, while bottom-right corner inpainting retains bird chirps. Irregular masks yield unique audio blends. Further images of this inpainting can be found in Appendix G.

WebUI also offers outpainting functionality, extending our audio length with additional generation. We extend our original spectrogram in both horizontal directions, expanding the length of our audio clips with generated audio. This allows us to extend the length of our audio clips with additional generation, offering additional flexibility to the audio lengths our model can provide.

This is a brief demonstration of the additional functionality that SD provides. This functionality can increase the flexibility and use-cases that our model is capable of. Inpainting offers a way for users to edit their audio with newly generated audio - creating blends of different audio classes or trimming entire durations from their audio to replace with new audio. Outpainting offers flexibility by extending our audio along the temporal dimension, allowing our model to generate audio longer than ten seconds. This control over the temporal dimension also represents a route for developing this research into a text-to-speech system - potentially using this outpainting to piece together audio clips of sentences of varying length.

Other functionalities not tested here include negative prompting, attention (allowing for certain elements of the prompt to be (de)emphasised), and prompt editing (adjusting the prompt at later stages in the generation process). Further research could explore different in-built functionalities within SD, as well as networks built to be used in conjunction with SD, such as ControlNet [88].

Chapter 7

Conclusions and future work

Contents

7.1 Overview of project

This project offers the first exploration into the feasibility of fine-tuning Stable Diffusion (SD) for generating natural and urban (non-music) audio. This novel research, conducted on consumer-level hardware, also tests the suitability of creating an accessible TTA paradigm for consumers looking to create their own systems.

To this end, we first conducted a literature review detailing the history of audio generation, diffusion models and the SD ecosystem. This showcased the efficacy of diffusion models in generating audio, evidence of SD in particular being used to generate audio through fine-tuning and highlighted the SD ecosystem - and how it could be leveraged to provide accessible and unique generation paradigms.

Subsequently, we undertook extensive empirical experiments, fine-tuning SD across 105 runs, amounting to a total compute time of 136 days. From these experiments, we generate initial evidence of the impact of various hyperparameters on fine-tuning - and subsequent generation.

Furthermore, we offer quantitative metrics for comparison with the current state-of-the-art (SOTA) and demonstrate the practical applications of the SD ecosystem for both consumers and future researchers.

7.2 Summary of novel findings

Throughout this research journey, we have delved deep into the capabilities and potential of fine-tuning SD for generating non-music audio spectrograms. This section aims to synthesise the key findings from each of the primary chapters of this research into a cohesive summary.

- **Model capabilities and limitations:** Our experiments revealed that SD can be successfully fine-tuned on consumer-hardware to generate non-music audio. While it excelled in certain areas, it faced challenges in others, emphasising the potential need for a more comprehensive training dataset. Simpler prompts yielded better results, suggesting a current limitation in the model's generalisability. However, we also discover our model's ability to generate spectrograms without explicit signifiers or keywords during training - which will be vital information for future research.

- **Training insights:** The relationship between loss curves and audio quality was nuanced. While loss curves indicated learning progression, they did not always correlate with audio quality. Training stability, longer training durations, and meticulous hyperparameter tuning emerged as crucial for optimal performance. The default values of the AdamW optimiser seemed to be well-suited for this fine-tuning task but future research could conduct a deeper dive into these parameters.

- **Performance benchmarks:** Although the model's performance did not surpass the current state-of-the-art, it showcased the potential for SD in audio generation. The model's performance was significantly influenced by the quality and completeness of its training dataset. There remains potential for performance improvement through a more exhaustive hyperparameter search.

- **SD ecosystem and versatility:** We also demonstrate the versatility and potential of the SD ecosystem. Inpainting and outpainting functionalities (among other SD functionality) offer users unprecedented control over audio generation. The model's ability to control the temporal dimension of audio opens avenues for further flexibility in applications as well as its application for text-to-speech (TTS) systems. Several functionalities within the SD ecosystem remain unexplored in this research, suggesting ample opportunities for future studies.

- **Consumer accessibility:** One of the primary objectives of this research was to assess the feasibility of SD as a TTA system, emphasising its potential for consumer accessibility. This research demonstrates that it is feasible to fine-tune SD on consumer-level hardware - opening the door for consumers to fine-tune their own SD models for TTA generation to suti their needs. The open-source nature and large community surrounding SD make it particularly suitable for democratising audio generation for consumers.

In conclusion, this research has provided a comprehensive exploration into SD capabilities, strengths, and limitations as a TTA system. The findings underscore the importance of dataset quality, meticulous parameter tuning, and the vast potential of the SD ecosystem. These insights not only lay a solid foundation for future research but also highlight the immense potential of SD in revolutionising audio generation for both researchers and consumers.

7.3 Future work

This project has demonstrated the potential and challenges of using Stable Diffusion (SD) for generating non-music audio spectrograms and represents the first step towards a wealth of further research in this space:

- **Performance enhancement:** A logical progression would be to bolster the model's performance. This can be achieved through a more exhaustive parameter search, such as a grid search, to delve deeper into the intricate relationships between various hyperparameters and their impact on training. Given the observed sensitivity of the model to training parameters, such a search could yield significant improvements.

- **Dataset expansion:** Our findings underscored the importance of dataset quality and completeness. Future endeavors could look into expanding the dataset by incorporating additional AudioSet classes, locating and reintegrating omitted portions of the AudioCaps dataset, and exploring additional sources such as the BBC Free Sound effects library [123].

- **Speech generation:** SD's ability to control the temporal dimension of audio, as demonstrated through inpainting and outpainting functionalities, suggests its potential applicability in text-to-speech systems. Exploring this avenue could lead to proving the feasibility of SD as a TTS system.

- **Human evaluation:** To provide a holistic assessment of the generated audio's quality, future research should consider human evaluations. Human evaluations within this space are often done with a panel of domain experts - including this element in further research would help improve the interpretability of the quality of our generated output, complementing quantitative metrics.

- **Exploring SD ecosystem:** The vast potential of the SD ecosystem remains largely untapped. Tools like ControlNet [88] offer a plethora of features, from improved consistency in inpainting to style-matching and image/text-guided generation. Harnessing these tools could usher in innovative paradigms of audio generation, providing users with additional control and versatility.

In summary, this research has offered a detailed examination of SD's potential and challenges in the realm of TTA generation. The insights gathered emphasise the significance of a robust dataset, careful parameter tuning, and the diverse capabilities within the SD ecosystem. These findings serve as a starting point, suggesting directions for further exploration and emphasising the potential of SD to make audio generation more accessible to a broader audience.

7.4 Ethics and usage concerns

7.4.1 Ethical concerns

There are two key ethical concerns surrounding generative models: misuse and copyright infringement. This specific research focuses on non-speech audio; this means the potential for misuse is much lower than what could arise from generation of speech audio. Generating bird song, or sounds of traffic could be seen as having lower potential harm than generating fake speech of real persons. However, there are instances where misuse could cause harm by generating audio with the purposes of misinformation - such as generating false sounds to portray a peaceful protest as violent. For example, image generation has previously been used by Amnesty International to create synthetic images of Colombia's 2021 protests - with the intention of further highlighting the political situation, but with the downside of potentially undermining their core message [124]. Similar misuse of audio generative models could also lead to confusion, misinformation and a potentially false portrayal of world events.

However, improving the accessibility of these systems can also aid creativity and productivity across a number of industries - especially creative industries. Creative work - such as film and video gaming - often relies on the artificial creation of sound effects through the physical manipulation of objects to recreate target sound. Text-to-audio (TTA) systems enable hobbyist creatives to have access to a multitude of different sounds, without needing to invest in recording equipment, or specialised knowledge; these systems may also free up resources from professional 'Foley' artists, allowing creative teams to redirect focus or resources towards other areas of interest.

If this research is extended towards text-to-speech (TTS) specifically, then this generative model may facilitate the creation of deepfakes by allowing the generation of fake speech of real persons. For example, celebrities are often targeted as subjects of abuse of misuse of generative audio systems - creating voice clips with their likeness speaking (unsavoury) things without the original person's consent [125]. Scammers have also been using TTS AI systems to generate realistic copies of loved ones speech - with the goal of extorting money from vulnerable people [126]. However, speech synthesis may also provide benefits - such as improving quality of life for persons with disabilities affecting their speech - by providing natural-sounding speech to use for communication.

Increasing accessibility to high-quality generation may increase ethical risks, and any increased benefits (including further democratisation of useful technology) need to be weighed against direct and indirect costs from misuse. A common complaint of AI systems is a lack of interpretability, especially for those unfamiliar with ML. Putting accessible systems into the hands of more people may directly increase hands-on experience across the population, and hopefully highlight the capabilities and limitations of this technology - leading to a greater ease of identification of misuse.

The second main concern is that of copyright infringement - especially regarding acquisition and use of training data. For our research, we use AudioSet [30] (and AudioCaps [29]), which are large-scale datasets based off 10 second intervals from tagged YouTube videos. These databases are provided under Creative Commons licenses, enabling access to users to share and adapt this data for any purposes, including commercially and academically. However, these datasets do not detail the ethical risks of the upstream data collection from YouTube itself. One might speculate that YouTube has clauses in their terms and conditions which grant Google the freedom to collect user-uploaded videos for dataset purposes; we also have to assume that this database did not include any obviously copyrighted material. However, with the rapid development of AI systems, it is entirely likely that video uploaders could not have foresaw their content being used to train models. Additionally, an answer in copyright law is yet to be fully determined - with confusion present in how potentially copyrighted training data, and generation built off this training data should be treated in our legal system [127, 128, 129].

This problem can also be seen within Stable Diffusion, which was trained on a subset of the LAION-5B dataset [85]. In the FAQs on the LAION website [130], they claim that their datasets are simply 'lists of URLs to the original image' without any strong indication of a consideration of the underlying copyrights of these images. This separation makes identification of copyright implications difficult, with one photographer suing LAION directly for his photos being included within their dataset [131]. Within the original paper for Latent Diffusion Models - there is no stipulation of any additional effort made to identify whether images in their training data is under copyright (which would be difficult to do for each of 2 billion images used).

Ultimately, AI research is built off the prior actions of prominent technology companies and research labs. AudioSet is one of the largest audio databases, and has been used for a multitude of downstream research involving model training and creation. This has the effect of any work built upon this downstream research, to also potentially carry the same ethical risks. SD is an open-source and accessible model but this increase in accessibility also means that any potential harm from copyright infringement is multiplied with its large user-base. However, without effective and clear regulation - along with transparency from the creations of datasets and foundational models (e.g. SD, ChatGPT) - we cannot aim to completely understand the scale of any copyright violations which may already be present throughout the field. With this research in particular, we can only draw attention to these potential problems and aim for transparency to minimise these risks.

7.4.2 Usage concerns

The LAION dataset, used to train SD, contains not-safe-for-work (NSFW) images. This means that this model is theoretically capable of generating unsuitable content for many audiences. Our fine-tuning does lead our model to generate spectrograms. However, in the event that NSFW content is detected, SD has an in-built filter preventing this content from being displayed - any inappropriate content which is accidentally generated should theoretically be prevented from causing any accidental harm.

AudioSet (and AudioCaps) consist of sound clips tagged from publicly YouTube clips - this means that any unsuitable content is either not present on the platform, or hidden behind an authentication barrier (preventing any NSFW sound from entering our dataset). This should mean that our model is incapable of producing any inappropriate audio.

7.4.3 Existing codebases used

In the interest of transparency, we also reiterate any existing codebases adapted for this research:

- **Diffusers' Training script**: We adapt Hugging Face's Diffusers' [24] 'train_text_to_image_lora.py' training script. We edit this script to generate validation and training loss per epoch, as well as adapt it to log our training, validation and inference outputs to Weights & Biases.
- **Riffusion Conversion Code**: We adapt Riffusion's code [23] used to convert audio to spectrograms and back. We adjust the hop length parameters to convert 10-second clips (as opposed to the 5-second clips Riffusion used) to 512x512 spectrograms.
- **AudioLDM Evaluation Code**: We directly use the evaluation code given by AudioLDM [31] for ease of comparison and usability.
- **WebUI Conversion Code**: We use a pre-existing script to convert our model checkpoints to safetensor files for use on the SD WebUI [132].

Bibliography

[1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, April 2022. arXiv:2112.10752 [cs]. pages 1, 8, 12, 13, 21

[2] Additive synthesis. url: https://support.apple.com/id-id/guide/logicpro-ipad/lpipd9c51c19/ipados [Accessed: 05/05/2023]. pages 1, 4

[3] How subtractive synthesizers work. url: https://support.apple.com/id-id/guide/logicpro/lgsife41a22f/mac [Accessed: 05/06/2023]. pages 1, 5

[4] Frequency modulation (FM) synthesis. url: https://support.apple.com/id-id/guide/logicpro/lgsife418213/mac [Accessed: 05/05/2023]. pages 1, 5

[5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio, September 2016. arXiv:1609.03499 [cs]. pages 1, 6

[6] Heiga Zen and Haşim Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4470–4474, April 2015. ISSN: 2379-190X. pages 1, 6

[7] Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, and Przemysław Szczepaniak. Fast, Compact, and High Quality LSTM-RNN Based Statistical Parametric Speech Synthesizers for Mobile Devices, June 2016. arXiv:1606.06061 [cs]. pages 1, 6

[8] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards End-to-End Speech Synthesis, April 2017. arXiv:1703.10135 [cs]. pages 1, 6

[9] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient Neural Audio Synthesis, June 2018. arXiv:1802.08435 [cs, eess]. pages 1, 6

[10] Chris Donahue, Julian McAuley, and M. Puckette. Synthesizing Audio with Generative Adversarial Networks. *ArXiv*, February 2018. pages 1, 7

[11] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial Audio Synthesis, February 2019. arXiv:1802.04208 [cs]. pages 1, 7

[12] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis, December 2019. arXiv:1910.06711 [cs, eess]. pages 1, 7

[13] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram, February 2020. arXiv:1910.11480 [cs, eess]. pages 1, 7

[14] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, Ming Liu, and Ming Zhou. Neural Speech Synthesis with Transformer Network, January 2019. arXiv:1809.08895 [cs]. pages 1, 7

[15] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. FastSpeech: Fast, Robust and Controllable Text to Speech. *ArXiv*, May 2019. pages 1, 7

[16] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech, August 2022. arXiv:2006.04558 [cs, eess]. pages 1, 7

[17] Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-TTS: A Denoising Diffusion Model for Text-to-Speech, April 2021. arXiv:2104.01409 [cs, eess]. pages 1, 14, 15

[18] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech, August 2021. arXiv:2105.06337 [cs, stat]. pages 1, 14, 15

[19] Nana Hou, Chenglin Xu, Van Tung Pham, Joey Tianyi Zhou, Eng Siong Chng, and Haizhou Li. Speaker and phoneme-aware speech bandwidth extension with residual dual-path network, 2020. pages 1, 14, 15

[20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis, March 2021. arXiv:2009.09761 [cs, eess, stat]. pages 1, 14, 15

[21] Max W. Y. Lam, Jun Wang, Dan Su, and Dong Yu. BDDM: Bilateral Denoising Diffusion Models for Fast and High-Quality Speech Synthesis, March 2022. arXiv:2203.13508 [cs, eess]. pages 1, 14, 15

[22] Dongchao Yang, Songxiang Liu, Jianwei Yu, Helin Wang, Chao Weng, and Yuexian Zou. NoreSpeech: Knowledge Distillation based Conditional Diffusion Model for Noise-robust Expressive TTS, November 2022. arXiv:2211.02448 [cs, eess]. pages 1, 15

[23] Seth* Forsgren and Hayk* Martiros. Riffusion - Stable diffusion for real-time music generation, 2022. pages 1, 15, 17, 22, 25, 64

[24] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models, 2022. pages 2, 14, 19, 22, 27, 32, 47, 64

[25] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. arXiv:2106.09685 [cs]. pages 2, 18, 27

[26] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation, March 2023. arXiv:2208.12242 [cs]. pages 2, 18, 27, 77

[27] AUTOMATIC1111. Stable Diffusion web UI, August 2023. pages 2

[28] Yan-Bo Lin, Yi-Lin Sung, Jie Lei, Mohit Bansal, and Gedas Bertasius. Vision Transformers are Parameter-Efficient Audio-Visual Learners, April 2023. arXiv:2212.07983 [cs, eess]. pages 2, 19

[29] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. 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)*, pages 119–132, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. pages 3, 22, 23, 63, 76

[30] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 776–780, 2017. pages 3, 22, 63

[31] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. AudioLDM: Text-to-Audio Generation with Latent Diffusion Models, February 2023. arXiv:2301.12503 [cs, eess] version: 2. pages 3, 14, 22, 23, 54, 55, 64

[32] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model, May 2023. arXiv:2304.13731 [cs, eess]. pages 3, 14, 22, 39, 54, 55

[33] Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D. Plumbley. AudioLDM 2: Learning Holistic Audio Generation with Self-supervised Pretraining, August 2023. arXiv:2308.05734 [cs, eess]. pages 3, 55

[34] Po-Yao Huang, Vasu Sharma, Hu Xu, Chaitanya Ryali, Haoqi Fan, Yanghao Li, Shang-Wen Li, Gargi Ghosh, Jitendra Malik, and Christoph Feichtenhofer. MAViL: Masked Audio-Video Learners, July 2023. arXiv:2212.08071 [cs, eess] version: 2. pages 3, 23

[35] Yuan Gong, Andrew Rouditchenko, Alexander H. Liu, David Harwath, Leonid Karlinsky, Hilde Kuehne, and James Glass. Contrastive Audio-Visual Masked Autoencoder, April 2023. arXiv:2210.07839 [cs, eess] version: 4. pages 3, 23

[36] Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, and Furu Wei. BEATs: Audio Pre-Training with Acoustic Tokenizers, December 2022. arXiv:2212.09058 [cs, eess] version: 1. pages 3, 23

[37] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Zero-shot Audio Source Separation through Query-based Learning from Weakly-labeled Data, February 2022. arXiv:2112.07891 [cs, eess] version: 4. pages 3, 23

[38] Youcef Tabet and Mohamed Boughazi. Speech synthesis techniques. A survey. In *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*, pages 67–70, May 2011. pages 5

[39] Jonathan Allen, Sharon Hunnicutt, Rolf Carlson, and Bjorn Granstrom. MITalk-79: The 1979 MIT text-to-speech system. *The Journal of the Acoustical Society of America*, 65(S1):S130, August 2005. pages 5

[40] D. Klatt. The klattalk text-to-speech conversion system. In *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 1589–1592, Paris, France, 1982. Institute of Electrical and Electronics Engineers. pages 5

[41] Martin Russ. *Sound Synthesis and Sampling*. Taylor & Francis, 2004. pages 5

[42] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech Synthesis Based on Hidden Markov Models. *Proceedings of the IEEE*, 101(5):1234–1252, May 2013. Conference Name: Proceedings of the IEEE. pages 5

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. pages 6

[44] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, December 2015. arXiv:1511.08458 [cs]. pages 6

[45] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks, December 2014. arXiv:1409.3215 [cs]. pages 6

[46] D. Griffin and Jae Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, April 1984. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing. pages 6, 18

[47] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions, February 2018. arXiv:1712.05884 [cs]. pages 6

[48] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, June 2014. arXiv:1406.2661 [cs, stat]. pages 7

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs]. pages 7, 8

[50] Naihan Li, Yanqing Liu, Yu Wu, Shujie Liu, Sheng Zhao, and Ming Liu. RobuTrans: A Robust Transformer-Based Text-to-Speech Model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8228–8235, April 2020. ISSN: 2374-3468, 2159-5399 Issue: 05 Journal Abbreviation: AAAI. pages 7

[51] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015. arXiv:1503.03585 [cond-mat, q-bio, stat]. pages 7, 9

[52] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020. arXiv:2006.11239 [cs, stat]. pages 7, 8, 9, 54

[53] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in neural information processing systems*, volume 31. Curran Associates, Inc., 2018. pages 8

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs]. pages 8

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs]. pages 8, 21

[56] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks, June 2017. arXiv:1605.07146 [cs]. pages 8

[57] Yuxin Wu and Kaiming He. Group Normalization, June 2018. arXiv:1803.08494 [cs]. pages 8

[58] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. pages 9

[59] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, January 2018. arXiv:1706.08500 [cs, stat]. pages 9, 54

[60] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders, June 2016. arXiv:1606.05328 [cs]. pages 9, 54

[61] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers, April 2019. arXiv:1904.10509 [cs, stat]. pages 9, 54

[62] Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive Quantile Networks for Generative Modeling, June 2018. pages 9, 54

[63] Yilun Du and Igor Mordatch. Implicit Generation and Generalization in Energy-Based Models, June 2020. arXiv:1903.08689 [cs, stat]. pages 9, 54

[64] Yang Song and Stefano Ermon. Improved Techniques for Training Score-Based Generative Models, October 2020. arXiv:2006.09011 [cs, stat]. pages 9, 54

[65] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution, October 2020. arXiv:1907.05600 [cs, stat]. pages 9, 54

[66] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks, February 2018. arXiv:1802.05957 [cs, stat]. pages 9, 54

[67] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling, July 2021. arXiv:2003.06060 [cs, stat]. pages 9, 54

[68] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data, October 2020. arXiv:2006.06676 [cs, stat]. pages 9

[69] Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models, February 2021. arXiv:2102.09672 [cs, stat]. pages 10, 11

[70] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis, June 2021. arXiv:2105.05233 [cs, stat]. pages 10, 11, 12, 54

[71] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis, February 2019. arXiv:1809.11096 [cs, stat]. pages 10, 11

[72] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021. arXiv:2011.13456 [cs, stat]. pages 10

[73] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks, March 2019. arXiv:1812.04948 [cs, stat]. pages 10

[74] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN, March 2020. arXiv:1912.04958 [cs, eess, stat]. pages 10, 54

[75] Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. LOGAN: Latent Optimisation for Generative Adversarial Networks, July 2020. arXiv:1912.00953 [cs, stat]. pages 11

[76] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance, July 2022. arXiv:2207.12598 [cs]. pages 11

[77] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, March 2022. arXiv:2112.10741 [cs]. pages 11

[78] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, February 2021. arXiv:2103.00020 [cs]. pages 11, 21, 53

[79] Federico A. Galatolo, Mario G. C. A. Cimino, and Gigliola Vaglini. Generating images from caption and vice versa via CLIP-Guided Generative Latent Space Search. In *Proceedings of the International Conference on Image Processing and Vision Engineering*, pages 166–174, 2021. arXiv:2102.01645 [cs]. pages 11

[80] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery, March 2021. arXiv:2103.17249 [cs]. pages 11

[81] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. StyleGAN-NADA: CLIP-Guided Domain Adaptation of Image Generators, December 2021. arXiv:2108.00946 [cs]. pages 11

[82] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. arXiv:1312.6114 [cs, stat]. pages 12, 21

[83] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation, February 2021. arXiv:2102.12092 [cs]. pages 13

[84] Midjourney. Available at https://www.midjourney.com [Accessed: 06/06/2023]. pages 13

[85] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models, October 2022. arXiv:2210.08402 [cs]. pages 13, 21, 63

[86] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021. pages 14

[87] AUTOMATIC1111. Stable diffusion web UI, August 2022. pages 14, 56

[88] Lvmin Zhang and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models, February 2023. arXiv:2302.05543 [cs]. pages 14, 58, 61

[89] Chenshuang Zhang, Chaoning Zhang, Sheng Zheng, Mengchun Zhang, Maryam Qamar, Sung-Ho Bae, and In So Kweon. A Survey on Audio Diffusion Models: Text To Speech Synthesis and Enhancement in Generative AI, April 2023. arXiv:2303.13336 [cs, eess]. pages 14, 15

[90] Jiawei Huang, Yi Ren, Rongjie Huang, Dongchao Yang, Zhenhui Ye, Chen Zhang, Jinglin Liu, Xiang Yin, Zejun Ma, and Zhou Zhao. Make-An-Audio 2: Temporal-Enhanced Text-to-Audio Generation, May 2023. arXiv:2305.18474 [cs, eess]. pages 14, 55

[91] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fr\'echet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms, January 2019. arXiv:1812.08466 [cs, eess]. pages 15, 54

[92] Leland Roberts. Understanding the Mel Spectrogram, August 2022. url: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 [Accessed: 22/05/2023]. pages 16

[93] S. S. Stevens, J. Volkmann, and E. B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, June 2005. pages 16, 17

[94] Shinnosuke Takamichi, Yuki Saito, Norihiro Takamune, Daichi Kitamura, and Hiroshi Saruwatari. Phase reconstruction from amplitude spectrograms based on von-Mises-distribution deep neural network, July 2018. arXiv:1807.03474 [cs, eess]. pages 18

[95] Nathanaël Perraudin, Peter Balazs, and Peter L. Søndergaard. A fast griffin-lim algorithm. In *2013 IEEE workshop on applications of signal processing to audio and acoustics*, pages 1–4, 2013. pages 18

[96] Paul Magron, Roland Badeau, and Bertrand David. Phase reconstruction of spectrograms with linear unwrapping: application to audio signal restoration, May 2016. arXiv:1605.07467 [cs]. pages 18

[97] Raja Bedoui, Zied Mnasri, and Benzarti Faouzi. Phase Retrieval: Application to Audio Signal Reconstruction, June 2022. pages 18

[98] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion, August 2022. arXiv:2208.01618 [cs]. pages 18, 77

[99] Simo Ryu. Low-rank Adaptation for Fast Text-to-Image Diffusion Fine-tuning, June 2023. original-date: 2022-12-08T00:09:05Z. pages 18, 27

[100] Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. Colossal-AI: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*, 2021. pages 19

[101] Valerii Likhosherstov, Anurag Arnab, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, and Mostafa Dehghani. PolyViT: Co-training Vision Transformers on Images, Videos and Audio, November 2021. arXiv:2111.12993 [cs]. pages 19

[102] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained Transformers as Universal Computation Engines, June 2021. arXiv:2103.05247 [cs]. pages 19

[103] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning, October 2022. arXiv:2206.13559 [cs]. pages 19

[104] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. VL-Adapter: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks, March 2022. arXiv:2112.06825 [cs]. pages 19

[105] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A Single Model for Many Visual Modalities, March 2022. arXiv:2201.08377 [cs]. pages 19

[106] Rohit Girdhar, Alaaeldin El-Nouby, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. OmniMAE: Single Model Masked Pretraining on Images and Videos, May 2023. arXiv:2206.08356 [cs, stat]. pages 19

[107] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. Stable Diffusion v1.5 Model Card, 2022. Details obtained from https://huggingface.co/runwayml/stable-diffusion-v1-5. pages 21

[108] Remita Amine. youtube-dl, 2021. GitHub repository. Obtained at:https://github.com/ytdl-org/youtube-dl. pages 23

[109] FFmpeg development team. FFmpeg, 2023. Available at: https://www.ffmpeg.org/. pages 23

[110] C.E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, January 1949. Publisher: Institute of Electrical and Electronics Engineers (IEEE). pages 24

[111] John Watkinson. *The art of digital audio*. Butterworth-Heinemann, USA, 2 edition, 1993. pages 24

[112] Suraj Patil, Pedro Cuenca, and Valentine Kozin. Training Stable Diffusion with Dreambooth using Diffusers, July 2022. Available at: https://huggingface.co/blog/dreambooth. pages 27

[113] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient Diffusion Training via Min-SNR Weighting Strategy, March 2023. arXiv:2303.09556 [cs]. pages 27

[114] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs]. pages 28, 29, 47

[115] Justin N. M. Pinkney. Pokemon BLIP captions, 2022. pages 32, 80

[116] Justin Pinkney. How to fine tune stable diffusion: how we made the text-to-pokemon model at Lambda, September 2022. pages 32

[117] Steven Bird and Edward Loper. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics. pages 39

[118] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A Reference-free Evaluation Metric for Image Captioning, March 2022. arXiv:2104.08718 [cs]. pages 53

[119] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs, June 2016. arXiv:1606.03498 [cs]. pages 54

[120] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Boston, MA, USA, June 2015. IEEE. pages 54

[121] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition, August 2020. arXiv:1912.10211 [cs, eess]. pages 54

[122] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-An-Audio: Text-To-Audio Generation with Prompt-Enhanced Diffusion Models, January 2023. arXiv:2301.12661 [cs, eess]. pages 55

[123] BBC Sound Effects. pages 61

[124] Luke Taylor. Amnesty International criticised for using AI-generated images. *The Guardian*, May 2023. pages 62

[125] Joseph Cox. AI-Generated Voice Firm Clamps Down After 4chan Makes Celebrity Voices for Abuse, January 2023. pages 62

[126] Pranshu Verma. They thought loved ones were calling for help. It was an AI scam. *Washington Post*, March 2023. pages 62

[127] Bird & Bird. AI training and copyright – how does the new Finnish Copyright Act deal with using copyright-protected works to train AI?, May 2023. pages 63

[128] Clifford Chance. AI-Generated Music and Copyright, April 2023. pages 63

[129] Farrer & Co. AI-generated music: timeless legacy or copyright breach?, August 2023. pages 63

[130] LAION. FAQ | LAION. Available at: https://laion.ai/faq. pages 63

[131] Andres Guadamuz. Photographer sues LAION for copyright infringement, May 2023. pages 63

[132] Harry Wang. Convert to Safetensors script, February 2023. Accessible at: https://github.com/harrywang/finetune-sd/blob/main/convert-to-safetensors.py. pages 64

[133] Jarosław Kochanowicz, Maciej Domagała, Dawid Stachowiak, and Krzysztof Dziedzic. Diffusion models in practice. Part 1: A primers, March 2023. Section: Generative AI. pages 78

Appendix A

AudioCaps ontology

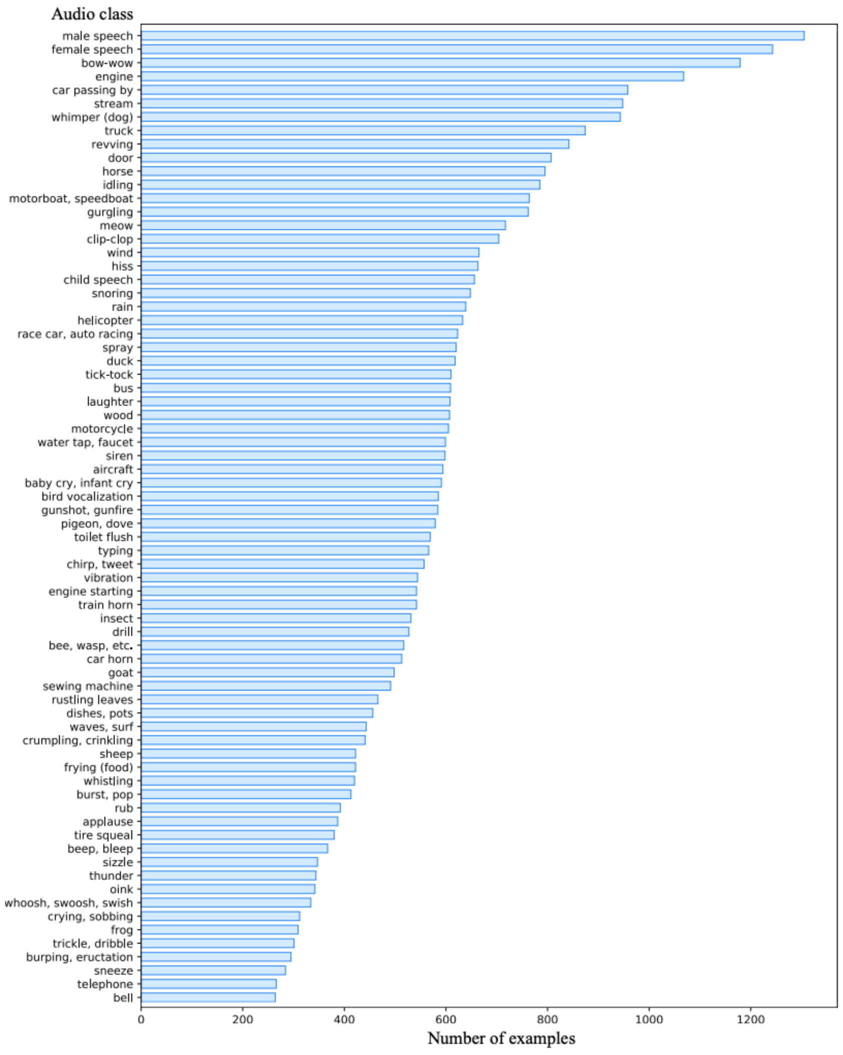


Figure A.1: Frequencies of annotated instances per category for AudioCaps. Figure reproduced from [29].

Appendix B

Fine-tuning illustrations

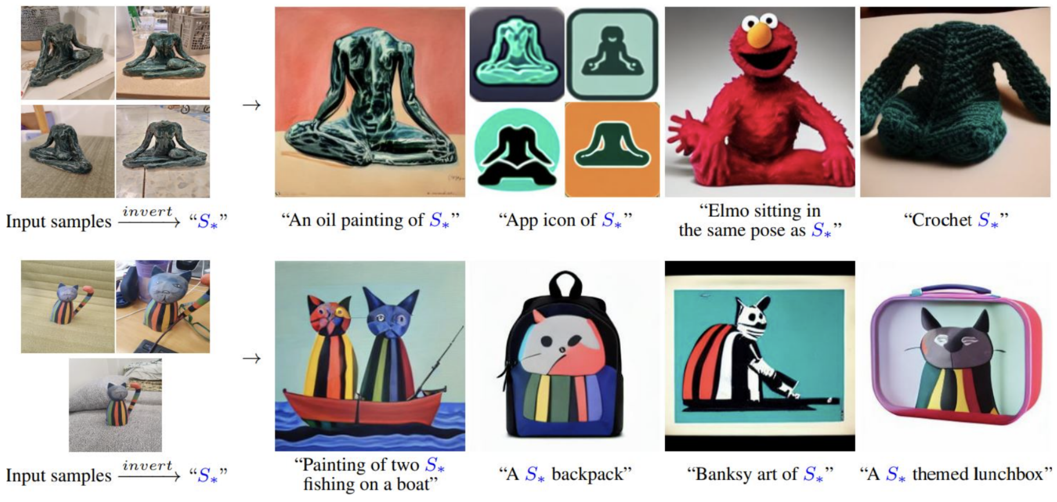

Input samples $\xrightarrow{invert}$ "S_*"
"An oil painting of S_*"
"App icon of S_*"
"Elmo sitting in the same pose as S_*"
"Crochet S_*"
Input samples $\xrightarrow{invert}$ "S_*"
"Painting of two S_* fishing on a boat"
"A S_* backpack"
"Banksy art of S_*"
"A S_* themed lunchbox"

Figure B.1: Textual inversion examples. Reproduced from [98].

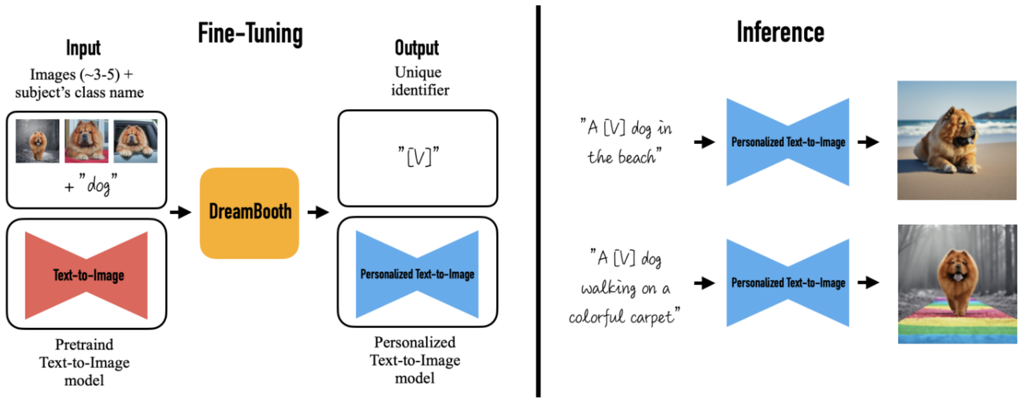

Input
Images (~3-5) + subject's class name
+ "dog"
Text-to-Image
Pretraind Text-to-Image model
Fine-Tuning
DreamBooth
Output
Unique identifier
"[V]"
Personalized Text-to-Image
Personalized Text-to-Image model
Inference
"A [V] dog in the beach"
Personalized Text-to-Image
"A [V] dog walking on a colorful carpet"
Personalized Text-to-Image

Figure B.2: DreamBooth approach overview. Reproduced from [26].

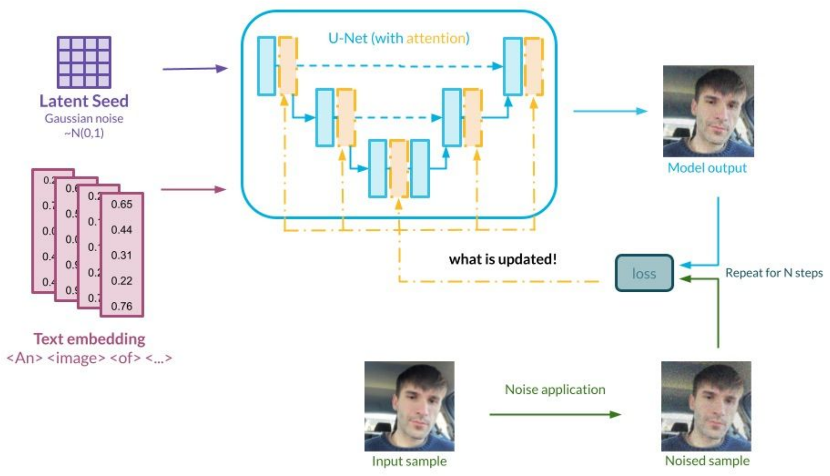


Figure B.3: Diagram illustrating LoRA weight injection, reproduced from [133].

Appendix C

Overview of experiments

Index	Learning Rate	warm-up Steps
1	$3e^{-4}$	16000
2	$3e^{-3}$	16000
3	$1e^{-4}$	16000
4	$1e^{-3}$	16000
5	$1e^{-3}$	30000
6	$3e^{-3}$	60000
7	$3e^{-3}$	32000
8	$1e^{-4}$	32000
9	$7e^{-4}$	28000
10	$5e^{-4}$	28000
11	$7e^{-4}$	32000
12	$5e^{-4}$	32000

Table C.1: Summary of 50 epoch experiments.

Index	Learning Rate	Warm-up Steps
1	$7e^{-4}$	28000
2	$5e^{-4}$	28000
3	$7e^{-4}$	32000
4	$5e^{-4}$	32000

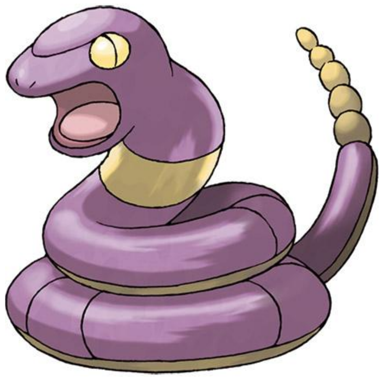
Table C.2: Summary of 75 epoch experiments.

Appendix D

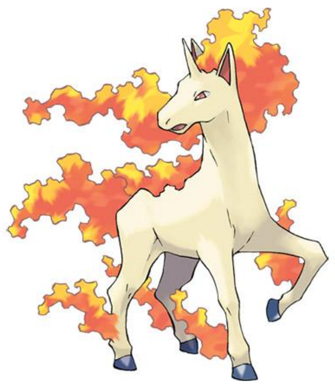
Pokémon BLIPS data samples

(a) A drawing of a green Pokémon with red eyes

(b) A bird with a long beak flying through the air



(c) A purple snake sitting on top of a white background



(d) A picture of a white horse with orange and yellow flames

Figure D.1: Samples from Pokémon BLIP caption database [115].

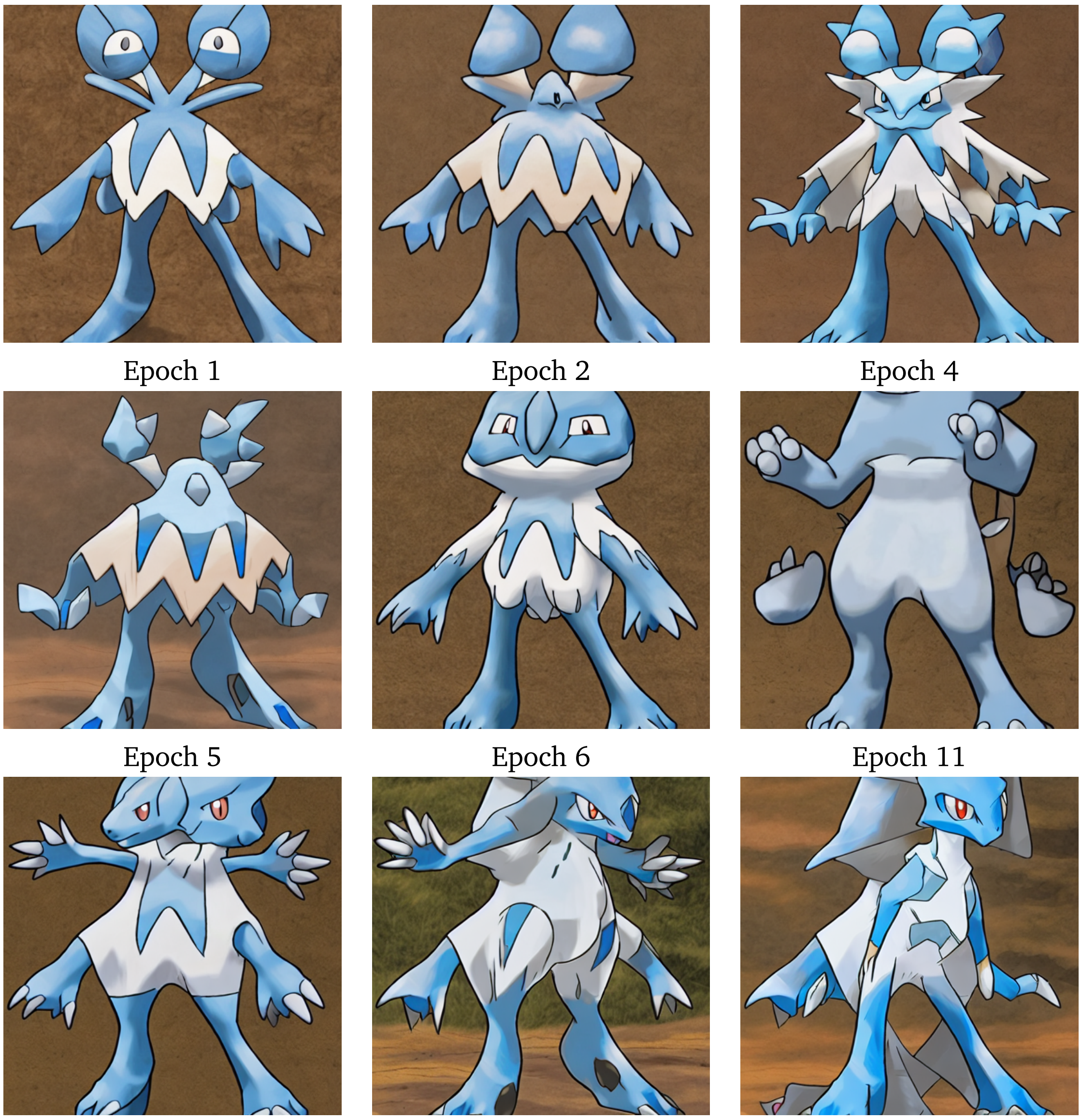
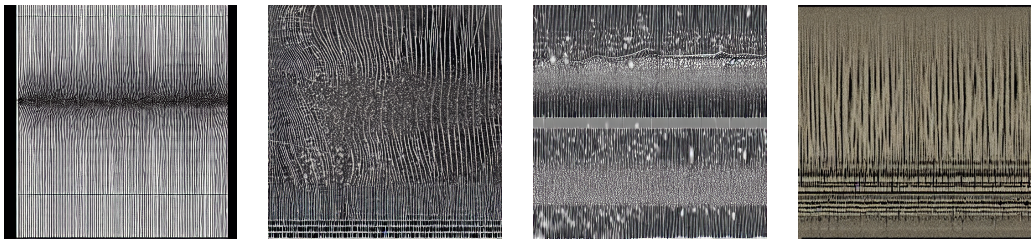
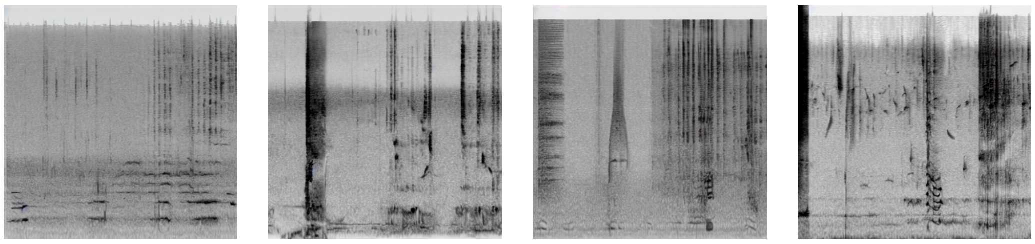


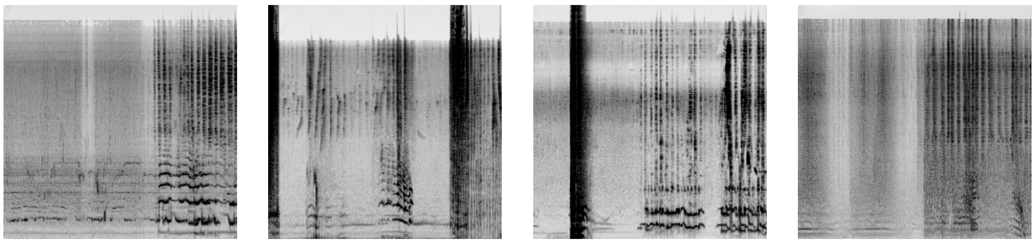
Figure D.2: Result of image generation using the prompt “a pokemon with blue eyes” of fine-tuning SD1.5 on Pokémon BLIP captions after different epochs during fine-tuning.

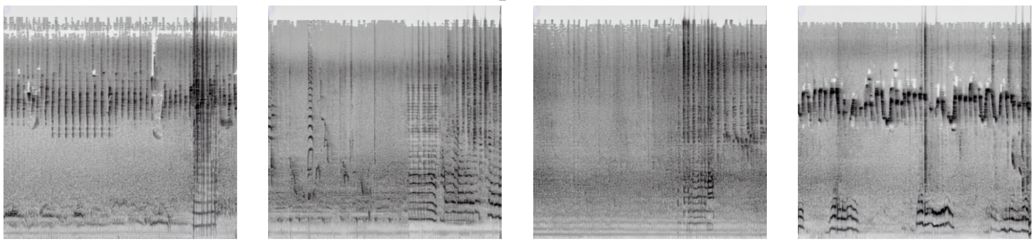
Appendix E

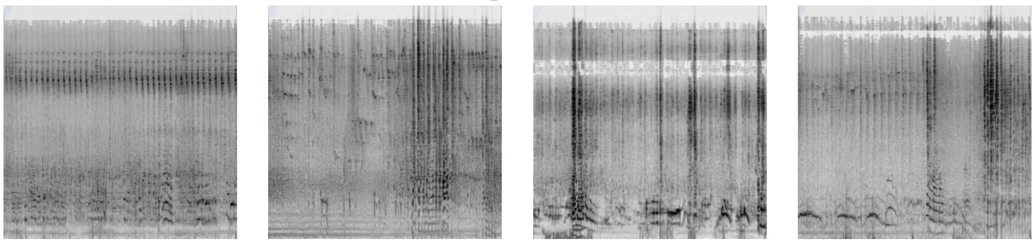
Experiment spectrogram examples


(a) Epoch 1


(b) Epoch 10


(e) Epoch 40


(c) Epoch 100


(d) Epoch 125

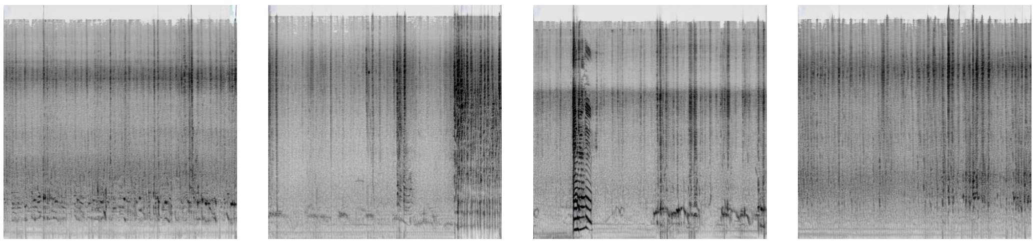
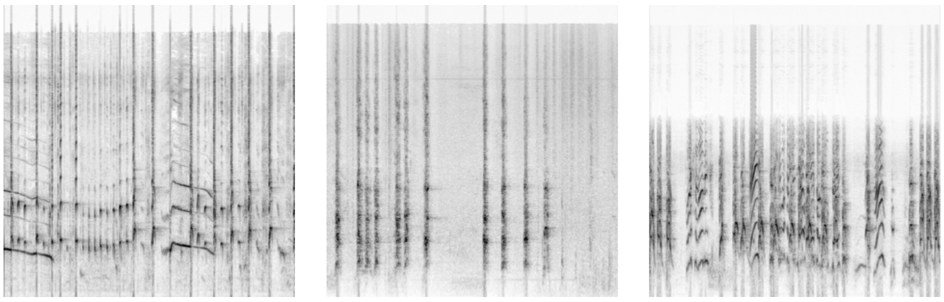
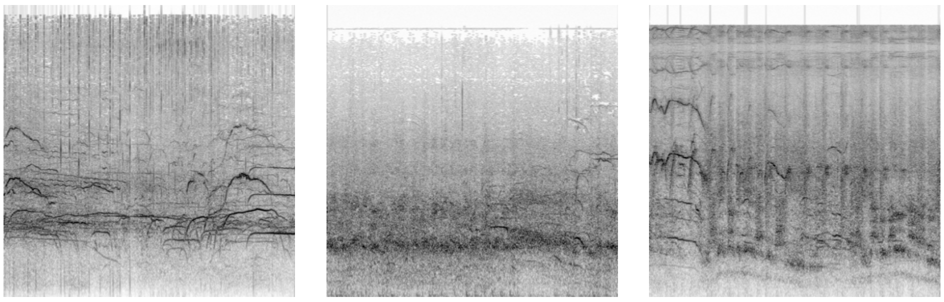
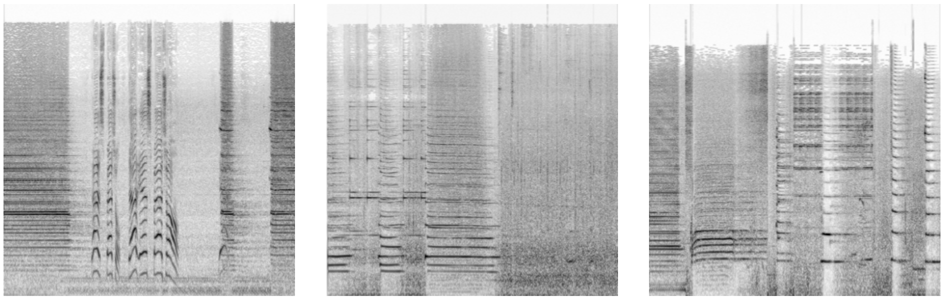
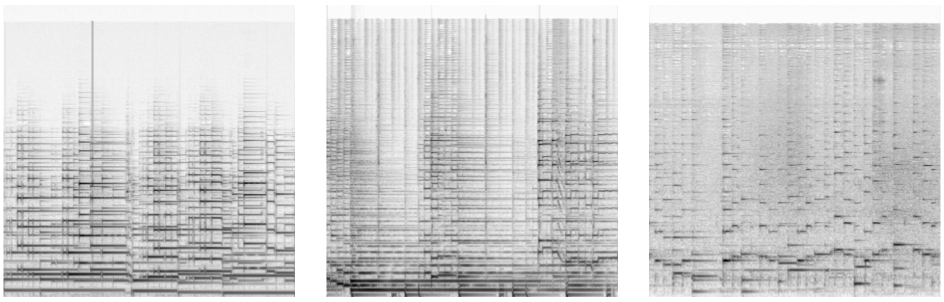

(f) Epoch 155

Figure E.1: Inference outputs for “a spectrogram of bird song” at different epochs


(a) Dog bark


(b) Cheering


(e) Train horn


(c) Acoustic guitar

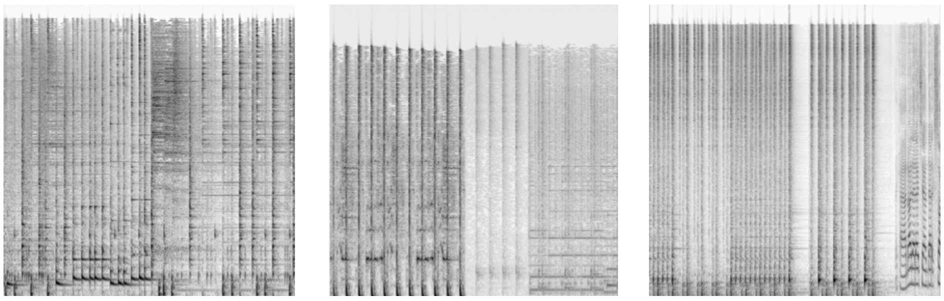
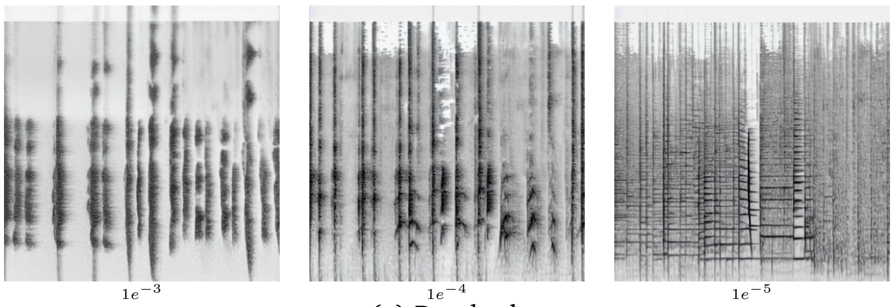
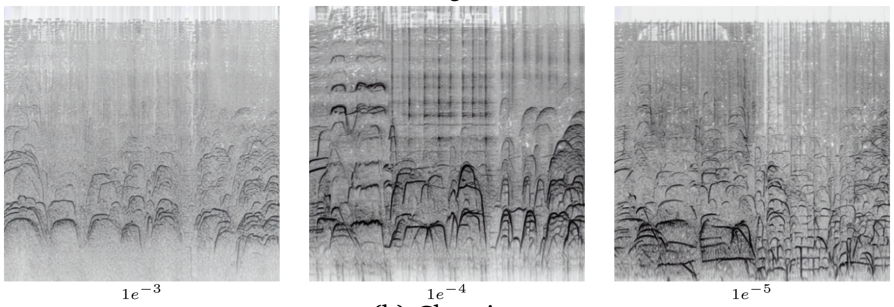

(d) Snare drum

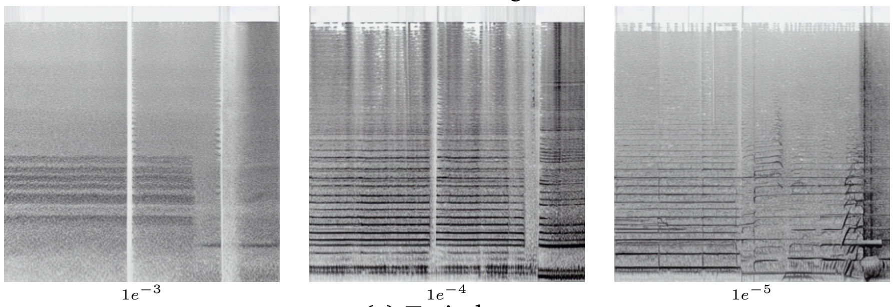
Figure E.2: Selected samples from 5-class AS training dataset.



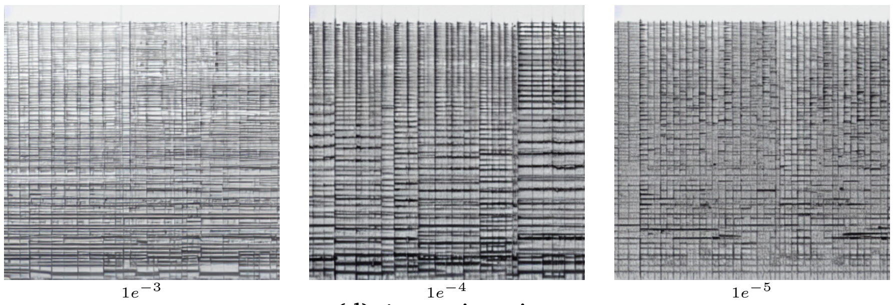
(a) Dog bark



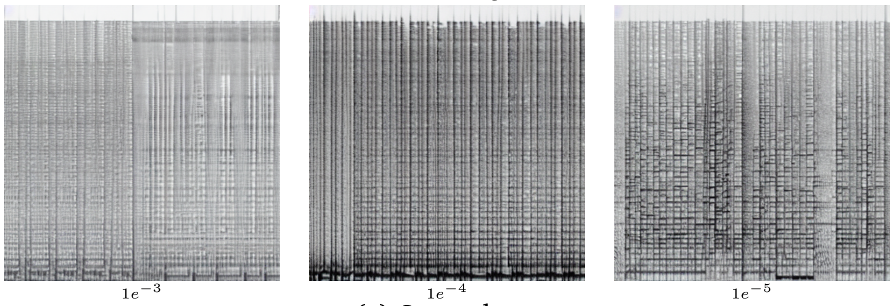
(b) Cheering



(c) Train horn



(d) Acoustic guitar



(e) Snare drum

Figure E.3: Selection of inference outputs from prompt “a spectrogram of [class]” at different learning rates after training for 61 epochs incl. 1,000 warm-up steps.

Appendix F

Inference prompts used during training

Our set of validation prompts (each prefixed with "a spectrogram of") for 50 epoch experiments are:

- "An animal hissing followed by a man mumbling then a pig oinking while birds chirp in the background"
- "Music plays and someone speaks before gunfire and an explosion occurs"
- "Someone is typing on a computer keyboard"
- "A horn honking followed by a man laughing then talking and plastic clanking on a hard surface"
- "A dog barking"
- "A train horn"
- "A hammer hitting a hard surface"
- "A man laughing"

Final set of validation prompts:

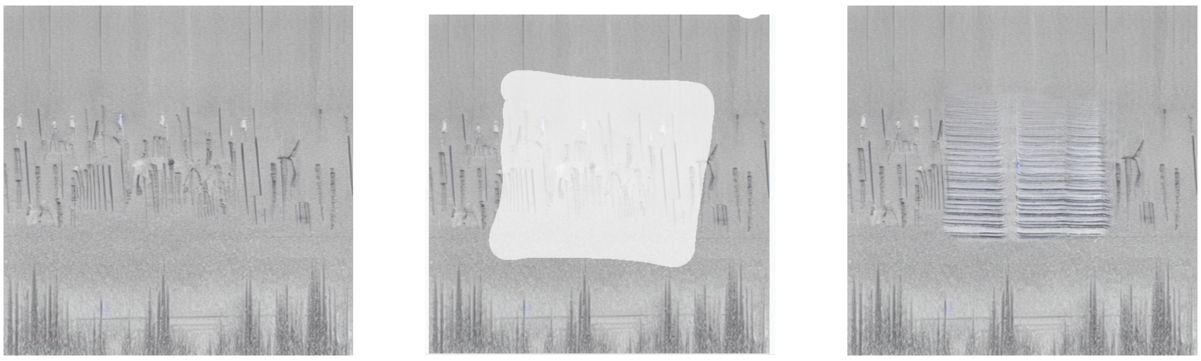
- "An animal hissing followed by a man mumbling then a pig oinking while birds chirp in the background"
- "Music plays and someone speaks before gunfire and an explosion occurs"
- "Someone is typing on a computer keyboard"
- "A horn honking followed by a man laughing then talking and plastic clanking

on a hard surface”

- “A dog barking”
- “A train horn”
- “Bird song”
- “Birds chirping”
- “A snare drum”
- “Horse hooves clip-clop and voices in the background”
- “Sheep bleats”

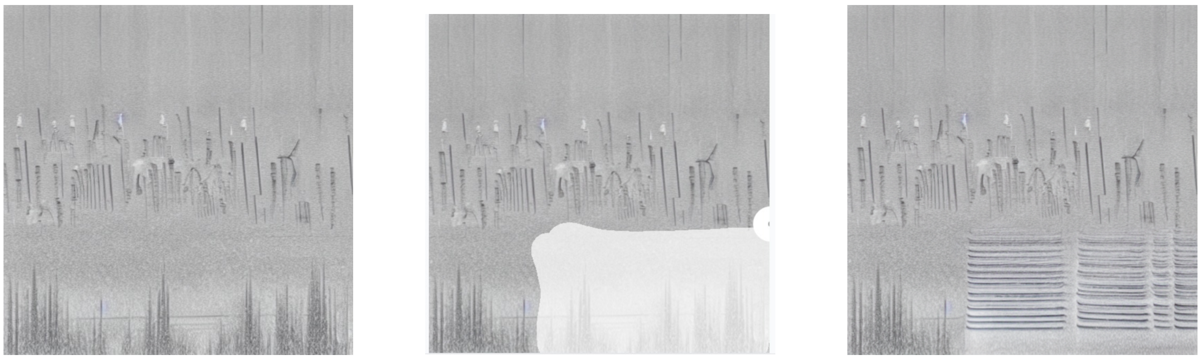
Appendix G

Inpainting and outpainting examples



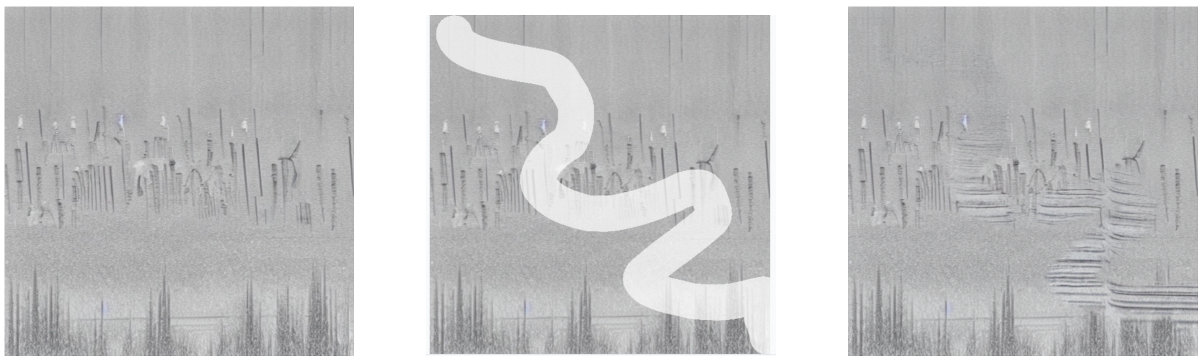
(a) Original image (b) Drawn mask (in light grey) (c) Inpainting result

Figure G.1: Demonstration of inpainting 'train horn' over the middle of a spectrogram generated with prompt 'a calm forest ambiance with birds chirping and distant rustling leaves'.



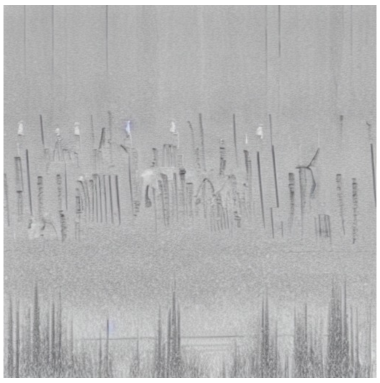
(a) Original image (b) Drawn mask (in light grey) (c) Inpainting result

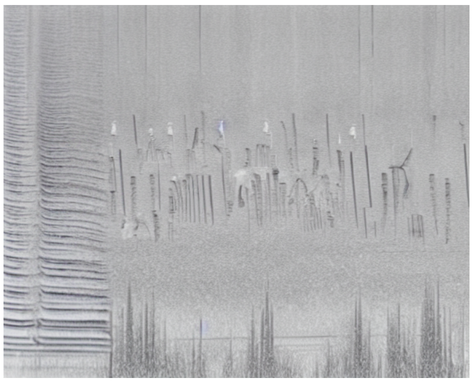
Figure G.2: Demonstration of inpainting 'train horn' over the bottom-right of a spectrogram generated with prompt 'a calm forest ambiance with birds chirping and distant rustling leaves'.

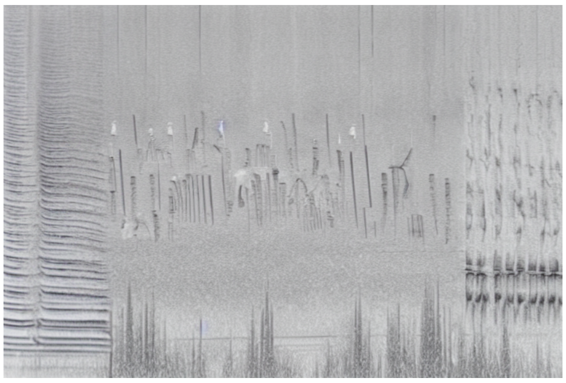


(a) Original image (b) Drawn mask (in light grey) (c) Inpainting result

Figure G.3: Demonstration of inpainting ‘train horn’ using irregular masking over a spectrogram generated with prompt ‘a calm forest ambiance with birds chirping and distant rustling leaves’.


(a) Original image: ‘a calm forest ambiance with birds chirping and distant rustling leaves’.


(b) Outpainting ‘train horn’ to the left (adding 2.5 seconds to duration).


(c) Outpainting ‘train horn’ to the left and ‘dog bark’ to the right (adding 5 seconds to duration).

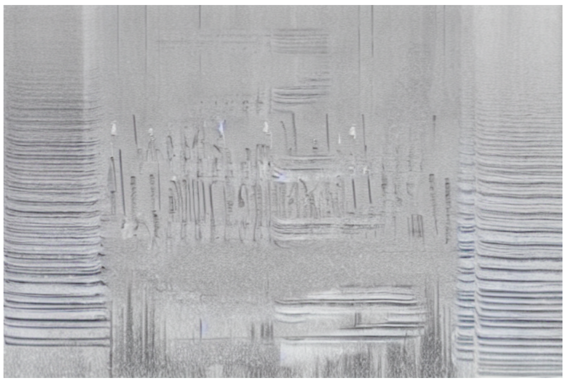

(d) Outpainting to the left and right, and inpainting ‘train horn’ over the original image.

Figure G.4: Demonstration of outpainting from SD WebUI.