Unsupervised Detection and Synthesis of Speech and Environmental Sounds Using Generative Networks

Shaun Charles Buckley

Submitted in accordance with the requirements for the degree of

PhD Computer Science

The University of Leeds

School of Computer Science

April 2025

Declaration

I confirm that the work submitted is my own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Shaun Charles Buckley, April 2025

Acknowledgements

I would first like to thank my supervisors Professor He Wang and Professor David Hogg for their invaluable guidance and support throughout my PhD study. Thank you for your belief in me and the mentorship you have provided.

I would also like to thank ZoneME Audio for their initial sponsorship of this project, which paved the way for this research to be possible.

Thank you to the team at the Environmental Investigation Agency, in particular Debbie Banks, for giving me the opportunity to apply my studies to a vital conservation project and work as part of a wonderful team.

Finally, I'd like to give a huge thanks to my family and friends, in particular my partner Lucy, who have provided endless support and encouragement throughout this process. There's no way I could have done this without you!

Abstract

The task of isolating specific sounds in a noisy environment is known as the “cocktail party problem”, where an individual needs to filter out all the irrelevant sounds and speakers to focus on a specific individual, for either localisation or understanding. Humans do this in a variety of settings outside of speech, such as focusing on individual instruments or vocals in a piece of music. This isn’t unique to just humans, as most animals capable of hearing need to focus on specific environmental noises for their own safety, whether that’s localizing where a branch just broke warning of an encroaching predator or tracking the sounds of potential prey. Filtering out irrelevant or unnecessary sounds to focus on a specific source is a natural function for hearing creatures, but a challenging mechanism to replicate in machines. Great advances have been made in the areas of speech denoising and music source separation, but the holy grail of universal sound separation that humans are capable of is still out of reach. Whilst there have been attempts to produce models that can separate several arbitrary sources, most make assumptions on either the number of sounds in a mixture and/or the number of classes in the dataset. Furthermore, these separation models are computationally expensive to train, often requiring multiple GPUs and large amounts of memory. To address these problems, first we combine the efficiency of convolutional networks with the global receptive field of axial transformers to produce a model for separating arbitrary sounds from entangled audio mixtures by capturing long-term dependencies within the data whilst reducing memory requirements. We then utilize the structured latent space of unsupervised VAEs to perform semi-supervised labelling of multi-class input such as audio mixtures by measuring the divergence between the latent distributions of a given sample and a set of given classes. Finally, we propose a generative audio framework using infinite generative adversarial networks for performing audio synthesis, class detection and audio classification.

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

1.1 Background and Context

Improving machine hearing to better approximate that of humans is an ongoing and important area of research, both for commercial and safety critical applications (Mirbeygi et al., 2022). The perception of sound is the experience of processing vibrations in the form of an acoustic wave, most commonly through air, by the brain. This oscillation in pressure propagating through a medium is what humans interpret as speech, music, environmental sounds and noise. Subtle variations in timbre are what allow us to pick out a specific individual amongst others, even if they are vocalising at the same frequency and volume (Li and Yang, 2021). Even amongst noisy environments, those with unimpeded hearing can often isolate specific sound sources for either focused perception or localisation. Approximating these natural functions was previously achieved by manually applying complex algorithms for filtering and processing audio data, though results and their applications outside of research were limited.

Deep learning has gained a huge amount of traction and interest in recent years with academics, industry and the general public due to a vast array of applications and potential for future research and growth. Research in computer vision specifically has seen many innovations that have pushed the field forward to create deeper, more powerful and accurate models on a variety of tasks such as image classification, segmentation and generation. This in part is owed to very large datasets of image/video data, much of which is strongly labelled. Deep learning research centred on audio is relatively sparse in comparison to computer vision, due to the lack of large datasets and the nature of the data itself. Where images are generally spatial 2D data of a fixed size and can be easily scaled up and down in size, single-channel audio is temporal 1D data with a much greater overall size, which relies on very specific and accurate frequencies, amplitudes and temporal structure to sound realistic and intelligible. This makes training audio deep learning models more complex and prone to noticeable distortion and artifacts. This is particularly prevalent with the audio source separation and raw end-to-end audio generation tasks, which this research is focused on.

1.2 Motivations

This research addresses the feasibility of universal audio source separation and generation, the current limitations of state-of-the-art methods and how these can be addressed. True universal audio source separation by definition would be the ability to separate any number of distinct sound sources within a single mixture from each other whilst retaining the individual quality of each source sample without the introduction of artifacts or distortion, whether they be partial samples of other audio samples or generated from the separation process itself. This is already an exceptionally difficult task due to the sheer variety of distinct sounds and the variation in the time-frequency characteristics between them (Tripathi and Mishra, 2021). To compound this, a universal audio source separator needs to be able to separate the distinct audio samples within a mixture without being given prior information of how many audio sources there are. The task of binary audio source separation is relatively simple in this respect, as it is essentially a denoising process where the task involves masking and removing any audio samples which don't correspond to the target class, leaving only the target remaining. This process can then be easily inverted to obtain the second distinct audio source, provided there is no other background noise. This can be especially powerful when the objective is to separate or denoise audio mixtures for a specific target class, such as speech. Training a binary audio denoising model with a strong target class dataset can lead to audio output which is greatly enhanced, such as completely isolating a speaker in a noisy background environment. A typical application of this is background denoising for isolating human speech, such as when speaking on the phone or recording a video of an individual in a noisy environment.

Audio source separation research is also popular with music centric datasets and tasks, for purposes such as isolating specific instruments within compositions recorded or saved as a single audio channel, making recovery of these instrument audio stems non-trivial. Example classes from the MUSDB18 (Stöter et al., 2018) dataset include drums, vocals, bass, accompaniment (e.g. guitar) and 'other'. Whilst the introduction of more complex and dynamic classes increases the complexity of the problem, the sound classes have strong long-term dependencies and are more predictable due to music's repetitive and structured nature.

Environmental sound separation, or more generally universal sound separation, is the task of being able to replicate this denoising or separation process on much more random and less salient sounds, such as those generated by objects (trains, glass-breaking), the environment (wind, running water) and living creatures (dog barking, crow cawing). Within each distinct audio class, variation in pitch, timbre, volume and duration can make learning generalised masks for each class extremely difficult in comparison to similar image tasks where a relatively consistent spatial mask shape that is simply translated, rotated and enlarged/shrunk can be learnt.

The factors that make audio source separation challenging also impact on the difficulty for raw audio synthesis. Where generated images are composed of homogeneous shapes where small variations in detail often do not impact the sentiment or interpretation of the output (e.g. a generated car being slightly smaller than normal or having irregular edges), minor changes to energy at different frequencies or when they occur within the sound can turn what would be intelligible speech into noise. Efficient and scalable universal audio synthesis would help to alleviate the current size limitations of environmental sound datasets, providing the resources for further improvement into machine hearing using methods such as deep learning.

In this research we will explore what has been achieved within the field so far, the techniques that were used and why. Specifically, there will be a focus on the application of computer vision techniques on audio data when converted into a similar 2D format to images such as mel-spectrograms and will explore the key similarities and differences between these two domains of data and how this impacts training and performance of deep learning models. This research will also investigate the specific limitations currently in the field of both image and audio separation and generation, what others have done to overcome these, and what else could still be done to push the field forward. Specifically, we propose a method to train generative models to automatically detect the number of audio classes within a dataset with nothing but a rough prediction whilst simultaneously improving their generation quality. We will highlight the challenges of training such a model, the potential applications of this technique to other tasks and the benefits it could provide for future work.

This study will help to highlight areas of deep learning for audio that are currently overlooked or sparsely explored which are necessary to contribute to the creation of truly generalised and useful deep learning audio models that require little to no prior knowledge of the data being fed into the application. The significance of this is to address the issue of huge amounts of data which are difficult to train with due to lack of annotation or labelling, as this data-preprocessing is expensive, time-consuming and prone to error both by human and machine agents. This would also enable models to respond to unseen classes of data more reliably without having to fully retrain with the added class, which can be problematic if the number of samples is significantly fewer than previously trained classes.

1.3 Contributions:

Exploring alternatives for tackling the universal sound separation problem is necessary if efficient and truly unsupervised and uninformed separation are to be achieved. With the application of the recently popularised transformer models in the image domain, we investigate how they can be applied to audio in the domain of source separation. Our proposed sound separation model combines the efficient local receptive field for extracting features convolutional models are famous for, with the global receptive

field that allow transformers to capture long-term dependencies within the data, which temporal data such as audio is especially sensitive to.

In our second piece of work, we test the feasibility of effectively modelling the latent space of audio datasets using variational autoencoders with unsupervised training for semi-supervised classification. The motivation for this comes from the benefits provided by providing conditioning in the form of automatic identification of classes within an audio mixture, to determine which sources must be separated and what to ignore as background noise. We test classification through the calculation of KL-divergence between the encoding of unknown audio samples and the distribution of a small batch of labelled samples per class in the latent space, taking the smallest divergence/s as predictions. We find given only a moderate number of labelled samples, we achieve relatively strong classification accuracy.

This led us to our third contribution, which attempts to leverage the mode-collapse problem that is common amongst generative adversarial networks. Raw audio synthesis is a challenging task, with other models instead performing style transfer to existing audio (Meier, 2017) or generating audio from lower-level representations such as mel-spectrograms (Kong et al., 2020; Kumar et al., 2019) rather than directly from a latent noise. One reason for this is mode collapse, which cause generative models to generate a similar subset of the data whilst ignoring all other data. In our work, we train infinite unsupervised generative adversarial networks (GANs) on different audio datasets, including speech and environmental sounds. Intuitively each GAN should eventually collapse to one or few individual classes within the dataset, which is then refined using an adversarial Chinese Restaurant Process (CRP) facilitated by an infinite Gaussian mixture model. We sought to leverage this by using our trained encoder from our VAE work, which computes likelihoods that samples belong to a given GAN, to determine whether the sample belongs to the cluster the audio class collapsed to. Our work finds that in contrast to images in this same task, mode collapse doesn’t necessarily follow the class modes, where each GAN produces a selection of all classes that cannot be refined further using the CRP process.

1.4 Thesis Structure Overview

In this work, the structure is as follows:

- **Chapter 2** provides an overview of the literature and related works across audio processing, separation and synthesis with respect to both earlier algorithmic methods and more recently deep learning. We cover state-of-the-art methods and the motivations for our work.
- **Chapter 3** presents our universal audio separation model utilizing mel-spectrogram inputs and a hybrid convolutional-transformer architecture, with axial attention for efficient processing. We evaluate the improvements an axial transformer can provide and the impact of model capacity on the sound separation task.
- **Chapter 4** explores our work using unsupervised VAEs to construct a structured latent space, for which we propose to perform semi-supervised audio classification through computation of class-distributions in the learned latent space for the labelling of audio separation datasets.
- **Chapter 5** introduces our work with unsupervised GANs, an IGMM and the CRP process for automatic classification and class-specific synthesis of audio datasets. We explore the key differences between images and audio in this task and how this effects the mode collapse and results of the model.
- **Chapter 6** gives a summary of the thesis, our findings and the limitations of our work. We also discuss potential future research directions based on our findings.

Chapter 2

2 Related Works

2.1 Time-domain vs. frequency-domain representations

In general, digitised audio is represented in the time-domain as a waveform. This format shows how an audio signal changes over time. The size of the waveform is determined by the sampling rate of the recorded sound and the duration of the recording itself, measured in samples. For example, a 5 second audio file recorded at a sampling rate of 16 kHz would consist of 80000 samples. Waveforms preserve as much information as possible from the original recording of the audio, containing the frequency, amplitude and phase information of the signal. Audio can also be represented in the frequency domain, which decomposes the waveform into its constituent frequencies and their respective amplitudes. This can be achieved using the short-time Fourier transform or STFT, which applies a fast Fourier transform or FFT to small windows of a given signal repeatedly. Whilst a FFT returns the amplitudes for the different frequencies within a signal, this is averaged over the entire duration of the segment of the signal it is being applied to. This is useful for stationary and unchanging signals such as those produced by machinery in diagnosing faults by measuring the frequencies of the vibrations (Goyal and Pabla, 2015), but for more interesting signals such as speech, music and environmental sounds, this doesn't provide a clear understanding of the overall signal. However, computing the FFT over small overlapping segments of time-domain data can decompose the signal into its constituent frequencies and their amplitudes whilst also describing how these change over time by arranging them in a 2D frequency-time representation. The window size is an important factor when computing the STFT of a signal, as a larger window size covering a greater segment of the signal will result in higher frequency resolution at a cost to the time resolution, and equally a smaller window size produces the inverse, lower frequency resolution and higher time resolution (Gharehpetian and Karami, 2023). The formula for computing the STFT is as follows:

$$S(f,\tau) = \sum_{t=0}^{N-1} x(t)\omega(t-\tau)e^{-j2\pi ft}$$

Where $x(t)$ is the audio time-domain signal, $\omega(t)$ is the window function centred on time τ, and N is the number of samples (Goyal and Pabla, 2015). This can be presented as a 2D spectrogram image of the data, which allows visual inspection of the intensity of different frequencies in each time window stacked from left to right where the left is the beginning of the signal, and the right is the end. An example of a waveform and the conversion of it to a spectrogram representation is shown below in Figure 2.1.

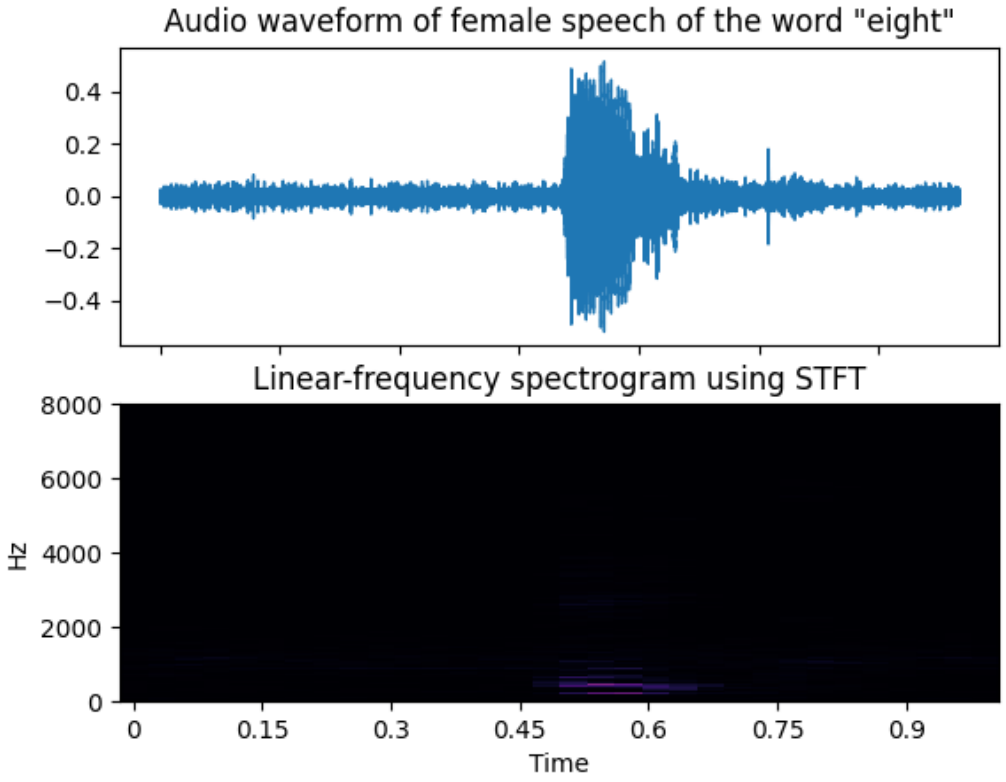

Figure 2.1: Audio waveform (top) of 1 second duration from adult female speaking the word "eight", linear frequency power spectrogram (bottom) of the same waveform decomposed into 2D spectrogram representation using STFT.

One might notice compared to the original waveform, there is not a great deal to see in the spectrogram, with a small amount of energy visible at the peak of the amplitude of the signal in the lowest frequency bands. This is because amplitude is linear and will only make the most intense frequencies in the signal visible. Substituting the linear amplitude scale for the logarithmic decibel scale gives us a much more comprehensive understanding of the decomposed signal, as shown in Figure 2.2. We can see most of the intensity in the spectrogram is located where the intensity of the waveform peaks, where lighter areas are louder and darker areas are quieter.

Humans interpret frequencies of a sound wave as pitch, but this is not perceived linearly. The greatest sensitivity we experience to frequency change is between 1 kHz and 2 kHz, where the change of just 0.2% in the pitch can be recognized as different (Oxenham, 2018). This sensitivity deteriorates dramatically as the frequency of a sound increases, where our ability to discriminate between higher pitches diminishes to the point of irrelevance the closer we get to our natural limit of 20 kHz. Therefore, when visualizing these signals in the frequency-domain representation, using a linear frequency scale is not reflective of human perception, where the most important intensity changes in a sound are squashed in a small frequency range and the remainder of the range contains relatively non-discriminatory information for understanding the signal. Similarly to how the decibel scale can be

applied to better visualize the intensity, a logarithmic frequency scale should be used to scale up the most important frequencies and scale down the least important higher frequencies. In 1937 the mel-scale was introduced, which was created by measuring at what intervals humans could reliably perceive changes in pitch to be equal distances from each other (Pedersen, 1965). The reference point was set at 1000 Hz which equals 1000 mels, where a doubling in the number of mels was equal to a doubling of the perceived pitch of the audio.

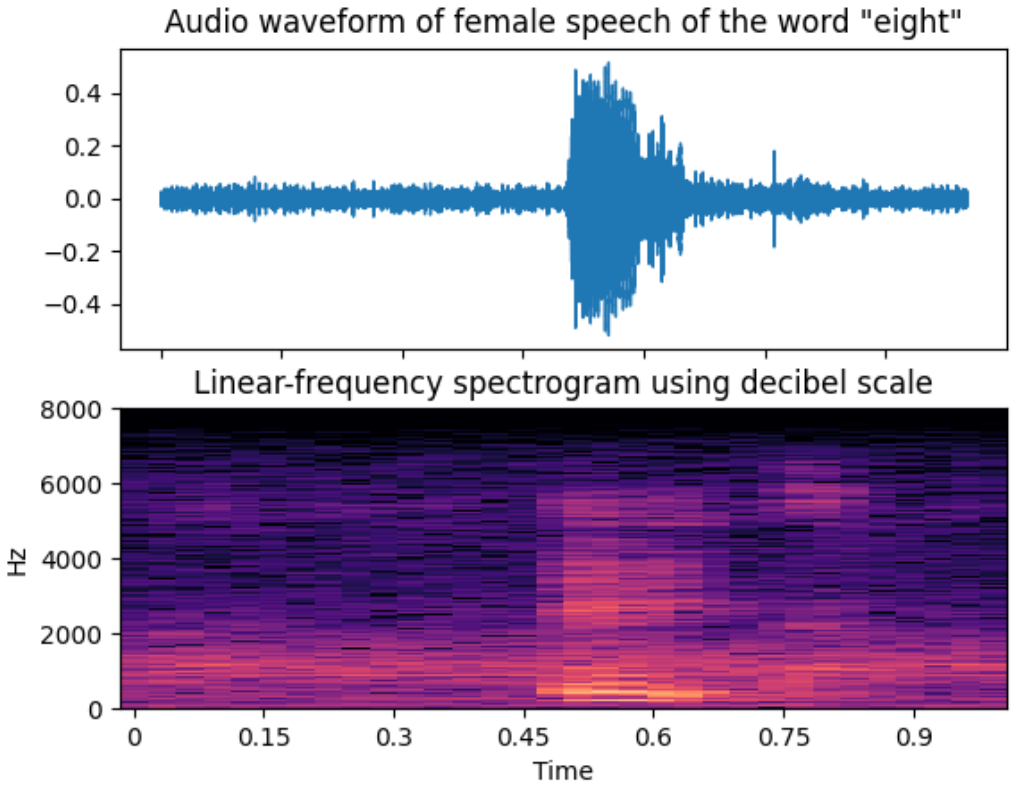

Figure 2.2: Audio waveform (top) of 1 second duration from adult female speaking the word "eight", linear frequency spectrogram (bottom) of the same waveform decomposed into 2D spectrogram representation using STFT.

The most perceivable range of changes in pitch are between 400-1000 Hz, where 400 Hz corresponds to approximately 500 mels and 1000 Hz corresponds to 1000 mels, giving an almost linear sensitivity in this range. This is because whether the difference in mels is between 1000-1100 or 3000-3100, the perceived difference between the two pitches is the same. Hence, the perceived difference between 400-1000 Hz is perceived to be a doubling in pitch of 500 mels. However, this sensitivity starts to decrease as the frequency increases, given that the difference between 1000 Hz and 2000 Hz is only perceived as a 50% increase in pitch at 1500 mels, which if linear would be 2000 mels. However, it is interpreted as the same pitch difference as 400-1000 Hz of around 500 mels, despite being almost double the frequency change. This gets even more dramatic past 4000 Hz, where the difference between 4 kHz and 10 kHz is only perceived as a shift of approximately 600-700 mels, indicating the

perceivable gap between the frequencies is the same as that between 400-1000 Hz, despite being 10 times the difference in frequency. Here, we can see the value in scaling these frequencies according to this scale, so the perceptive importance of each frequency range is highlighted appropriately. Mapping an audio signal's frequencies to the mel-scale after performing the STFT over the waveform allows us to compute the mel-spectrogram of the signal, shown in Figure 2.3.

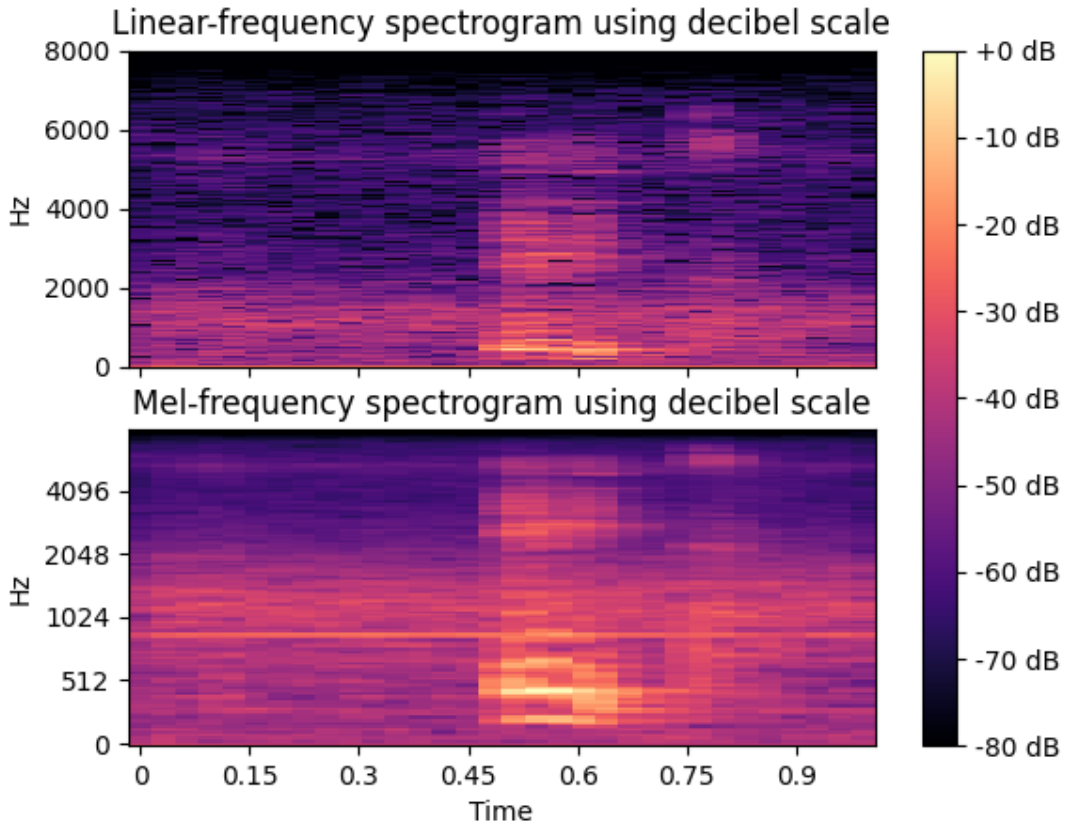

Figure 2.3: Linear frequency spectrogram (top) of 1 second duration from adult female speaking the word "eight", mel scale frequency spectrogram (bottom) of the same signal, amplitudes scaled to dB in both.

Whilst the two are similar, the frequency scaling in the mel-spectrogram expands the size of the lower frequency bands so the change in intensity of the sound energies at these more perceivably important frequencies is more visible. This feature extraction technique therefore is a method of preserving and increasing the weighting of the most perceptually important information contained in an audio signal. When it comes to digital audio processing, especially in deep learning, mel-spectrograms are often chosen over waveforms due to their lower dimensionality to simplify training, reduce computational complexity and guide the model to focus on the most significant patterns in the data. This also allows continuous time-domain audio signals to be processed in a similar way to images in computer vision, due to it being a 2D representation. A distinct difference however is an image is spatial, where x and y axes represent the spatial position in the image, whereas in a mel-spectrogram the x-axis represents time and the y axis the frequency, as it is still temporal data.

2.2 Deep Learning for Audio Processing

2.2.1 Machine Learning Fundamentals

Deep learning is a subfield within machine learning that applies the principles of the human brain to solving problems with artificial neural networks many layers deep (Noor and Ige, 2024), hence the name. The basic concept consists of putting data through an input layer, extracting features and learning to understand the patterns of the data with the intermediate layers, then finally producing a useful output in the final layer. This output depends on the task, of which there are many that have applied deep learning to solving them. For example, a deep classification network may take images as input, decompose the structure for feature extraction at different layers to understand them, then outputs a prediction of a class that they belong to. Initially, unless the model is pretrained or otherwise given prior information or tuned weights, these predictions will be guesses. To teach the neural network, it must have some metric of measuring its performance that it is trying to improve in order to get better at its task, instead of simply producing random outputs. To achieve this, neural networks use loss functions, which take the outputs of the model and compare them against the expected output, computing a gradient with respect to the model parameters, leading to improvement in performance on the sample being processed. The ultimate goal of training is for a model to continuously adjust its parameters for convergence to either increase or more often decrease the loss value. For example, in autoencoders the goal is encode the input and reconstruct it with as little distortion or differences as possible. This is known as reconstruction error, where the goal of the model is to reduce the measured difference between a reconstruction and the original input to 0.

Prior to the mass adoption of deep learning, machine learning relied on taking input and transforming it into a feature vector, which through analysis could be used to make inferences on the data for predictive modelling and other tasks (Noor and Ige, 2024). This requires feature engineering, where data domain experts would need to manually inspect datasets and iteratively perform feature transformation, extraction and scaling if their models were to succeed at their intended task, and can be as much trial and error as it is an exact science (Murel, 2024). Feature transformations can include grouping data by given categories or ranges of values known as ‘binning’, where continuous real values are grouped into discrete bands and weights are updated for each bin rather than every individual value. This preprocessing step helps to simplify the dataset and eliminate potential outliers or noise by amalgamating data with similar values and smoothing their means.

Feature extraction is useful when working with high-dimensional data, as this involves extracting the most statistically significant features whilst ignoring the rest. One common method of achieving this is principal component analysis (PCA), which transforms and reduces complex data to a lower dimensionality whilst preserving as much variability within the data as possible to preserve

statistically important information (Jolliffe and Cadima, 2016). Specifically, it reduces the data down to its principal components, where the first principal component contains the greatest amount of variability along its dimension, and the second principal component which contains the second most variability whilst being totally uncorrelated with the first principal component. This is to say, the second principal component must be orthogonal to the first with zero correlation between the two dimensions. This means that if PCA successfully extracts the most distinct and important features, when projected into a 2D visualization the similarity between two samples of data can be interpreted as the distance between them. More principal components can be extracted provided the rule of no correlation between components is preserved, which also means more information from the original data is preserved for learning from. However, this also makes the data harder to interpret, meaning selecting a balance of principal components whilst preserving a significant portion of total variance in the data after this reduction is important.

Feature scaling, otherwise known as normalization, is a technique for bounding data whose values may vary greatly in scale. This is because the statistical importance for some features in the data may lie within a small range of values, compared to others with a much larger range. This can lead models to learn biases towards features with greater variance in values and ignore the more important features with smaller variability, leading to less accuracy or predictive power (Amorim et al., 2023). Min-max normalisation is a common technique in data-preprocessing, as this bands all values within the range of 0-1, which can be useful for specific algorithms that rely on data existing within this range such as those that use binary cross-entropy loss to compute the probability distribution between the prediction and the target. This can be computed over an entire dataset with the following equation, where x is the original unscaled data:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

However, this doesn't address the problem of differing value ranges within the dataset and merely scales the problem to a much smaller range, whilst preserving the scale disparity and failing to equalize the mean and variance which can be unstable for machine learning pipelines (Amorim et al., 2023).

Another common technique is z-score normalization otherwise known as standardization. This involves scaling the data, so it has a mean of 0 and a standard deviation of 1. This is particularly useful in machine learning for making the model more robust against outliers (Kim et al., 2025), as the difference between the scale of typical and extreme values is massively reduced. To achieve this, the following equation can be used:

$$x' = \frac{x - \text{mean}(x)}{std(x)}$$

There is no rule that normalization in data needs to be applied, however it has been shown to improve the accuracy of predictions in machine learning algorithms, making it a common strategy in feature engineering.

A major advantage of deep learning over traditional machine learning is the ability for deep neural networks to automate many of these feature engineering techniques using hidden layers of artificial neurons. These successively extract features at different resolutions and representations in a hierarchical fashion (Noor and Ige, 2024), which in the example of images may be first extracting coarse features such as collections of pixels that form edges and corners in one layer, passing this to another layer which learns how these features combine to form more complicated features, which is again passed to later layers which build a more complex understanding of how these representations fit together that make the given input or class of data distinct from another. Feature engineering can also be manually applied to datasets as a preprocessing step for use in deep learning, which similar to traditional machine learning is likely to improve performance and accuracy. In our case, this includes feature extraction with conversion of waveforms to mel-spectrograms and normalization techniques.

2.2.2 Convolutional Neural Networks

Convolutional neural networks or CNNs are one of the most popular deep learning algorithms available to data scientists, due to their ability to extract important features from a range of data domains with no human supervision or intervention required. In the field of audio processing, CNNs have been used in a huge number of machine learning tasks such as classification (Hershey et al., 2017), speech enhancement (Drgas, 2023), audio source separation (Nath and Sarma, 2024) and data synthesis/generation (Božić and Horvat, 2024). The term convolutional neural network was coined as early as 1998 in the paper proposing LeNet, a network designed to extract features in images of hand written digits from the MNIST dataset for number identification (Lecun et al., 1998). Deep learning with convolutional networks became much more prominent in 2012 with introduction of AlexNet, when it achieved state-of-the-art (SOTA) performance on the ImageNet classification challenge, boasting a huge increase in accuracy over previous attempts with a 15.3% error rate, a full 10.9% less than the next best entry (Krizhevsky et al., 2017). In every subsequent year the ImageNet challenge was held, the winning model was always an implementation of the CNN algorithm, with ever increasing performance to the point the error rate reached 2.3%, surpassing the performance of humans who only achieved an error rate of 5% (Langlotz et al., 2019). The key feature of the CNN is the learnable kernel, low dimensional matrices that pass over the input to extract information.

Say we have a 2D image consisting of pixels in a value range of 0-255 and a 3x3 kernel of values: [0, -1, 0, -1, 5, -1, 0, -1, 0]. The original input of a 2D image is first padded by surrounding the data with values of 0, which is an optional step to prevent information loss from edge values in the input. Then the 3x3 kernel is passed over each original pixel value in the image, where the surrounding pixel

values are also considered. Each value the kernel is attending to is multiplied by the corresponding value defined in the kernel itself, the total of which is added together to give a new value for that pixel. For example, to calculate value of a feature map for an input pixel 54, which is where the kernel is currently centred on, the following operation is performed for centre and surrounding pixels [0, 38, 96, 0, 54, 103, 0, 72, 93]:

$$(0 \times 0) + (38 \times -1) + (96 \times 0) + (0 \times -1) + (54 \times 5) + (103 \times -1) + (0 \times 0) + (72 \times -1) + (93 \times 0) = 57.$$

The kernel convolves over every pixel in this way until a new feature map is produced, which in this case is the same dimensionality as the original input. The kernel values used in this case were hand-picked for the purpose of sharpening edges by increasing the weighting of the value the kernel is centred on and comparing this with adjacent pixels. This increases the contrast between higher and lower value pixels, resulting in edges with more depth and a sharper image (Ganesh, 2019). When using CNNs in a deep learning architecture, the values of these kernels are learned through backpropagation of gradients that adjust the parameters of the model and weights of each layer. Each channel in an input feature map will use its own kernel, which stacked together forms a filter, extracting features from each layer of an input.

Often, the size of the input and output in a CNN layer will be different, depending on hyperparameters such as stride. Stride is a term which defines the number of pixels to move the kernel across in each convolution as it slides across the data. To preserve the height and width of the input feature map the stride for both would be (1, 1), as each pixel would be computed and have a new value returned, whereas a stride of (2, 2) would result in a pixel being skipped after every convolution on both, resulting in lower dimensionality. This can be useful for eliminating pooling operations between convolutions and reducing computational cost whilst downsampling the input for successive layers performing feature extraction at different scales. The change in dimensionality for a given kernel size and stride is given by the following equation, where is W the input width size (substitute with H for height), K is the kernel size, P is the padding and S is the stride:

$$W' = \frac{W - K + 2P}{S} + 1$$

For an image of size 49x49 with a kernel size of 3x3 and stride (2, 2) and no padding, this gives an output size of $\frac{49-3+(2*0)}{2} + 1 = 24$, the new width of the feature map.

Convolutional layers are often stacked, extracting feature maps at different resolutions, but to prevent this from becoming a linear process, activation functions are sandwiched between them, common amongst many different deep learning algorithms. Activation functions mimic a process that happens in the human brain, where the combined electrical output from one neuron via the synapse known as

an ‘action potential’ is transmitted into another neuron, the strength of which determines whether the destination neuron is activated or inhibited if it is over a specific threshold (Platkiewicz and Brette, 2010). Some common activation functions in deep learning include Sigmoid, Hyperbolic Tangent (Tanh) and Rectified Linear Unit (ReLU), each with their own use cases, advantages and disadvantages.

Generally CNNs tend to use ReLU activations or variants of it (e.g. LeakyReLU) between the hidden layers in the network, but in recent years more modern activation functions have been introduced to address the limitations of ReLU such as Swish, which when used to replace ReLU layers in a CNN consistently outperformed it (Ramachandran et al., 2017). ReLU is more computationally simple, and for shallower networks Swish may not yield enough improvement to offset the extra complexity and training time and execution (Sharma, 2017), for this reason experimentation should be done with both to determine the optimal choice.

2.2.3 Transformers

Transformer networks are a form of sequence-to-sequence model that take an input sequence, feeds it through an encoder to produce a latent context vector to learn an understanding of the contextual information of the sequence, before decoding this latent representation into another sequence. This is commonly used in natural language processing (NLP) problems, with long sequences of characters, words and sentences. An excellent application of this is translation of text from one language to another, where the Seq2Seq model takes a sentence in one language, encodes it to a context vector, then decodes this vector into a different language whilst preserving the context of the original sequence. In the past, recurrent neural network (RNNs) models were used to tackle these tasks, but they suffer from issues such as the vanishing gradient problem (Sandu, 2024). Put simply, a vanishing gradient is when a model processes a relatively long sequence of data learning the relationship between them, but upon backpropagation through the layers of the network the gradient can decrease massively, preventing long-term dependencies from being learnt. Long short-term memory (LSTM) based models were created to address these problems (Pascanu et al., 2013), which introduced ‘forget’ gates that learnt when to ignore or use information from previous layers when updating the weights. This prevents the gradient from vanishing in earlier parts of the sequence, allowing long-term dependencies to be learnt. However, both RNNs and LSTMs must process data sequentially in the form of time-steps, which can lead to them being excessively slow over long sequences. In the ideal situation, entire sequences could be processed in parallel, capturing the relationships between all elements at the same time, both short and long-term, which is where transformers shine.

Transformer models are composed of blocks containing a multi-headed self-attention (MHSA) layer, a position-wise feed-forward network (FFN), normalization layers, residual connections and a positional embedding (Tay et al., 2023). Input is generally a sequence of tokens, which for NLP tasks

can be a sequence of individual words or phrases each represented by a vector, and 2D data (such as images) can be broken down into patches of data points that can be mapped to individual token vectors (Turner, 2024). However, the transformation of data into tokenized input doesn't preserve information about the order or position of the token within the original input data. This is important for understanding the relationship between different tokens, as the position of words in a sentence can fundamentally change the meaning of it (e.g. boy eats banana -> banana eats boy), patches of an image that should be combined to form a cohesive shape would leave badly finished jigsaw puzzle, and an audio waveform would be a jumbled mess of frequencies happening in a temporal order that would be totally unintelligible. This is why most transformers also add a positional encoding, which is added or concatenated to the tokenized input to preserve this information, using either a fixed vector of sinusoidal frequencies or a learnable embedding.

Self-attention is a mechanism by which the transformer determines the relevance between all tokens in a sequence, allowing it to capture both short and long-term dependencies within the data. This is accomplished using 3 vectors obtained through passing the input sequence through linear transformation layers: the query (Q), key (K) and value (V) vectors. The query represents what the transformer is looking for, the keys represent what is contained within the tokens, and the value represents the actual content of the tokens themselves. Transformers often use multi-head self-attention, which refers to multiple self-attentions happening in parallel, querying different features or relationships within the data with respect to each token. Scaled dot-product attention is performed using the Q, K, V vectors for each self-attention head, which can be mathematically represented in the following equation, where h represents a given attention head and α is a scaling factor:

$$A_h = Softmax(\alpha Q_h K_h^T) V_h$$

The results of this for each attention head are concatenated together before being passed into another linear layer to aggregate the weighted relationships and patterns within the sequence. After normalization, this serves as input to a position-wise feed-forward layer which are typically composed of two fully connected layers and a ReLU activation, which in contrast to the self-attention layer operates on each position in the sequence independently to learn to capture more features and patterns within the data.

Transformers have seen extensive use in language, image and audio based tasks, especially since the introduction of ViT (Dosovitskiy et al., 2021) which popularised the use of transformers on 2D data. However, transformers are prohibitively expensive due to the way attention must compute each token with respect to all other tokens, leading to a quadratic computation cost of $O(N^2)$, where N is the sequence length of the input. Therefore, with an increase in size of the sequence or dimensionality of the input data, there is a quadratic increase in memory and computational resources required to process it. This has led to many proposed variations of the transformer that reduce this complexity

whilst maintaining the benefits that the transformer provides, many of which use fixed attention patterns that provide a more sparse receptive field whilst maintaining a larger receptive field to capture long-term dependencies than those provided by convolutional networks. For higher dimensional data such as audio, these sparse transformers may not only be more suitable, but necessary for efficiency and feasibility of training on a single system or GPU.

2.2.4 Images vs Audio

Whilst audio can be converted to a time-frequency representation such as a spectrogram, there remain some critical differences between audio and images in the application of deep learning for solving similar tasks. For example, data augmentation can be an important step in the training pipeline of a deep learning model, as this can increase the size and diversity of the dataset resulting in a more reliable and versatile solution on evaluation. However, although the spectrograms are represented in two dimensions, they are not translationally or rotationally invariant like images. This is due to each axis representing very contrasting aspects of the audio, specifically frequency and time. If either axis were to be flipped, a common image augmentation technique, the resulting sound wouldn't reflect or bear any similarity to the original. Any rotation or translation of the audio in the 'image' would also completely change the perceived content and class of the input waveform, making it unrecognisable as the original sound. This means only techniques which preserve the delicate temporal nature of audio data and the important harmonic and long-term relationships within are appropriate for use, such as pitch-shifting and time-stretching using programming audio libraries such as Librosa.

Similarly, image-based architectures such as ViT and DCGAN, which have seen success in tasks such as classification and image generation, have seen adaptations to the audio domain. However, the size of each audio file in a dataset is typically much larger than in an equivalent image dataset. For example, in the ImageNet (Deng et al., 2009) dataset, each image tends to be cropped or pre-processed for use at around 224x224 pixels, resulting in a sample of size 50,176. In contrast, AudioSet (Gemmeke et al., 2017), which is often used in training audio classification models, consists of 10 second clips which can have waveforms extracted, typically with a sampling rate of at least 16kHz. This means each second of audio contains 16000 samples, or 160000 in total, over 3 times larger than our comparative image, which is also quite large in comparison to other image datasets such as MNIST (Li Deng, 2012), being only 28x28 pixels in size. This means in attention-based architectures like ViT, which scale computational cost quadratically, even small increases in data size can quickly make typical training unfeasible without huge computational overheads. Therefore, reducing the size of audio data prior to training through encoders or using strided convolutions may be necessary for using similar image-based approaches.

Another key difference between images and audio is the importance of long-term dependencies within audio data, as often distinct entities within images are locally constrained or connected, whereas in

audio data, local receptive fields could struggle to capture important relationships both between frequencies and temporally. This motivates the use of larger or global receptive fields such as those provided by transformers, though this necessitates dealing with the computational cost that accompanies it, or using more efficient transformer architectures than the typical ViT model structure. These differences highlight that although image-based techniques in deep learning can be extremely effective when dealing with the audio domain, considerations and adjustments must be made to avoid problems and poor performance caused by their inherent differences.

2.3 Audio Source Separation

Audio source separation is at its core a segmentation problem, where a mixture of overlapping sound sources is disentangled to reconstruct their original independent waveforms. In the image domain the equivalent would be separating out individual objects/entities from the background or each-other, such as separating a dog from the field behind it or the pigeon next to it. In this example, the dog is analogous to the target sound source, the background is noise, and the pigeon is another distinct intelligible audio source. In the context of humans, this is commonly known as the “cocktail party problem” (Cherry, 1953), which is the process by which an individual at a busy cocktail party full of other people speaking can identify a specific individual’s voice that they are looking for or trying to listen to. The process for which we can naturally do this so well is not clear (Ephrat et al., 2018) but researchers have endeavoured to develop methods of replicating this with machines on digital audio signals. Prior to the use of deep learning to solve this problem, several unsupervised manual methods were used to separate audio, such as nonnegative matrix factorisation (NMF) (Pimpale et al., 2016).

Our work doesn’t involve this method so we won’t cover it in depth, but we will address the basic concept and its drawbacks. NMF in the context of audio source separation is the process of decomposing a given matrix of an audio signal containing a mixture of two different sound sources (or one sound source and background noise) in the form of a spectrogram into two non-negative matrices, which each represent the original audio source stems (Matevosyan, 2023). When the product of these matrices is taken, the original mixture should be obtained. To reverse this process, these individual nonnegative matrices must be approximated. To achieve this, Lee and Seung proposed multiplicative update rules that can iteratively solve for the original matrices by randomly initializing them and minimizing a reconstruction loss by changing the values in the matrices until convergence (Lee and Seung, 2001). This can lead to accurate decompositions of mixture spectrograms and reveals hidden patterns in the data. However, the computational complexity of NMF increases drastically with larger datasets, and is heavily dependent on the initialisation of the component matrices (Otten, 2023). More modern approaches favour the utilization of deep learning networks, often aiming to learn estimates for separation masks, which when multiplied over the original input mixture leave only the target sound energy, removing or ‘masking’ all other noise or sources.

2.3.1 Speech Separation

Speech Separation is the challenging task of disentangling different speakers within a target mixture; this is because they occupy similar frequency bands and amplitude ranges. This is especially difficult for speakers with similar characteristics such as gender, which can provide distinctive variations in timbre or recognizable features (Agrawal et al., 2023). Generally, speech separation is trained in a supervised manner, taking input mixtures and performing separation for a pre-defined number of speakers. A long-lasting problem in the field was the permutation problem, where a separator network needed to assign a speaker label to each separated source for comparison against the ground truth source stems. For example, when a mixture in the form of a single channel is split into two channels, each containing the sound energy corresponding to one of the separated sources, each of these channels must be compared to the original sound sources, which each also occupy a single channel. However, say for example the first channel in the ground truth represents a speaker 'Sam', and the second represents 'Chris', how does the model decide which of the separated channels corresponds to who? It could be that the separated sources are of good quality, yet if compared against the wrong original source, would yield poor results, meaning a specific permutation of speaker assignment must be chosen. This could work for separation on a dataset of two speakers (or n speakers for a mixture containing n sources), where each speaker is assigned a single expected channel, but this fails to scale for multi-speaker datasets and requires prior knowledge.

There have been attempts to solve this problem, including deep clustering of speaker spectrogram embeddings (Hershey et al., 2015), which outputs an embedding vector and performs unsupervised clustering to group features of individual speakers together, rather than assigning channels to specific sources directly. However, the separation of these features is performed in the embedding space and therefore doesn't result in an end-to-end mapping between the target and predicted source, meaning an additional network is needed to unfold these features, reducing quality. The deep attractor network (DANet) was introduced to solve this problem, where instead of applying deep clustering after inference, 'attractor points' are learned that represent the target sources, and the embeddings of the separated sources are softly assigned to these attractors using similarity estimates, which are used to calculate masks for end-to-end separation (Chen et al., 2017). However, although the number of attractor points are not explicitly pre-defined, DANet was only tested on mixtures containing a pre-defined number of sources, meaning unless it was developed further or proven to work on a dynamic number of sources as it does with deep clustering, this is a limitation for generalisation. Permutation invariant training (PIT) is another solution that takes a different and much simpler approach to solving the problem. The separator network outputs a prediction for S sources relating to the S speakers in the mixture, then PIT calculates the score, using a metric such as MSE, across each of the channels with the ground-truth sources over $S!$ possible assignments between targets and predictions. With these scores, PIT selects the permutation with the lowest score, as this most likely to correspond to all

predictions being assigned to their correct target (Yu et al., 2017a). This theoretically allows PIT to adapt to an unfixed number of sources, however the computational complexity dramatically increases with increasing numbers of speakers as $n!$ assignments must be calculated, making it unsuitable for mixtures with many different sources.

Popular models in speech separation include Conv-TasNet, due to their performance in separating speech on 2 and 3 speaker datasets operating only in the time-domain, rather than the often-used time-frequency domain. This allows end-to-end separation whilst preserving information such as phase that is lost when performing feature extraction or conversion to a time-frequency representation (Luo and Mesgarani, 2019). This works by first encoding the input waveform using a simple convolutional layer and feeding this to a separator module. The separator consists of blocks of convolutional layers with different levels of dilation to increase the receptive field after each successive operation, before feeding into the next block. The learned features from these layers are then summed using skip connections to produce masks, which are applied to the encoded mixture to produce separated source predictions. These are then decoded with another simple convolutional layer to reconstruct the original target waveforms.

More recently, transformers have been applied to the speech separation problem, as showcased by SepFormer (Subakan et al., 2021a). This model, like Conv Tas-Net, applies an auto-encoder style architecture to the problem, where first the input is encoded to a latent representation before being used as input into a separation module for the learning and estimation of masks. These masks are applied to the encoded input using element-wise multiplication before being decoded to the predicted output waveforms. The key difference between the two is the separation/masking module, where Conv Tas-Net uses a primarily convolutional approach, and SepFormer uses attention modules. Specifically, the masking network uses two different blocks, an ‘IntraTransformer’ (IntraT) and an ‘InterTransformer’ (InterT) block. This works by splitting the input in the time domain into overlapping chunks to reduce the input into smaller snippets and adding a sinusoidal positional encoding, where IntraT learns dependencies within the data independent of other chunks. The data is then permuted in the last two dimensions to allow InterT to learn long-term dependencies between the chunks of audio data. The architecture for both transformers follows the same design of normalization and multi-headed attention, but for capturing a different receptive field. This purely transformer based approach to learning the separation masks achieved SOTA results over previous approaches which mostly consisted of purely convolutional approaches, including Conv Tas-Net (Subakan et al., 2021a), showcasing the power transformers can have in the audio domain.

In recent years, the SOTA has been pushed even further, using architectures and training setups with a significant focus on transformers, both in the 1D time and 2D time-frequency tasks. Based on the standard benchmark of scale-invariant signal-to-noise-ratio improvement (SI-SNRi) (Roux et al.,

2018) on the WSJ0-2Mix dataset (Hershey et al., 2015), often evaluated as the baseline dataset across speech separation models, the top performers are currently the MossFormer2 (S. Zhao et al., 2024), TF-Locoformer (Saijo et al., 2024) and SepReformer (Shin et al., 2025) models.

The MossFormer2 model sought to achieve SOTA without the use of recurrent neural network (RNN) blocks, instead using feed-forward sequential memory network (FSMN) blocks, which capture fine scale recurrent patterns in the data without the need for RNNs, enabling parallel processing of entire sequences whilst increasing separation performance. Their recurrent module first reduces dimensionality of the time-domain sequence through a bottleneck layer consisting of a 1x1 convolution, PReLU activation and LayerNorm whilst retaining important features. This is both used as input to, and combined with the output of, the gated-convolutional unit (GCU) layer, finally passing through an inversion of the initial bottleneck layer (minus the PReLU activation). Their GCU is inspired by the GLU (Dauphin et al., 2017) but benefitting from reduced dimensionality through convolutional units rather than linear, enabled by their bottleneck layer.

TF-LocoFormer takes a different approach, operating over a time-frequency representation obtained through applying an STFT, then processing each dimension separately with a dual-path framework. This isn't novel in and of itself; specifically, they sought to achieve SOTA results with this approach whilst eliminating the need for RNNs which the previous best dual-path time-frequency speech separation model TF-GridNet uses. This maintains the benefits of capturing long-term dependencies using transformers' global attention mechanism without weakening the ability to learn local finer features, or sacrificing parallelisation through RNNs and thus reducing training time. This is achieved by applying FFNs with Swish gated linear units (SwiGLU) before and after self-attention, using 1D convolutional/deconvolutional layers in place of linear layers. They also apply a novel normalisation technique they call RMSGroupNorm, a variation of root mean square normalisation (RMSNorm) which they claim can improve multi-speaker separation through the grouping of the vectors into equal sized groups based on the number of speakers and normalising each separately.

SepReformer takes inspiration from SepFormer through the application of separate transformer blocks for global and local attention for capturing coarse and fine features. However, there are a significant number of differences, leading to SepReformer achieving a large increase in performance. Firstly, the model follows a U-Net style multi-scale encoder-decoder architecture using skip connections, though this is asymmetric. They also perform speaker separation at the bottleneck between the encoder and decoder, where each speaker stream is separately reconstructed, whilst incorporating temporal features at each resolution from the encoder. Also, their global and local attention modules don't follow a dual-path system like SepFormer's Intra and Inter transformer blocks and don't require chunking, facilitating the efficient processing of long sequences. To address the problem of clustering speech features into other speaker channels, the decoder also contains a cross-speaker transformer, directly

following the output of the global and local transformer modules. This applies attention across the speaker dimension whilst processing the temporal dimension separately, disentangling components from different speakers per frame to recover lost information through the split into speaker channels.

These methods were all tested on the WSJ0-2Mix dataset, their best results of which are displayed in Table 2.1. Some of these implementations use dynamic mixing, a data augmentation technique which mixes new speaker mixtures each epoch, which can increase performance. SepReformer (L) and TF-LocoFormer (L) achieve the greatest scores, though the difference between all models listed is fairly small, especially when this is not the only measure of performance and can ignore other factors that could affect performance of these models, such as real-world reverberations and noise, which WSJ0-2Mix doesn't contain due to being a 'clean' audio dataset. The WHAM! (Wichern et al., 2019) dataset addresses this lack of realism in the speech separation problem by introducing real-world noise to 2-speaker mixtures, which WHAMR! (Maciejewski et al., 2020) extends even further by adding simulated reverberation. Libri2Mix (Cosentino et al., 2020) is a large speaker dataset used as a baseline in this task which can have WHAM! noise incorporated, though is often used in its clean format, as is the case with these models. Even taking this into account, results on the noisier datasets indicate that whilst performance is significantly lower, models which achieve SOTA results on WSJ0-2Mix also achieve the highest scores on more realistic data.

Table 2.1: SI-SNRi scores measured in dB of SOTA models in speech separation. Scores achieved using dynamic mixing marked with ''.*

Model	SI-SNRi (dB)			
	WSJ0-2Mix	WHAM!	WHAMR!	Libri2Mix
Sepformer	20.5*	16.4*	14.0*	19.2
MossFormer2	24.1*	18.1*	17.0*	21.7
TF-LocoFormer (M)	24.6*	-	**18.5**	**22.1**
TF-LocoFormer (L)	**25.1***	-	-	-
SepReformer (M)	24.2	17.8	-	22.0
SepReformer (L)	**25.1***	**18.4***	17.2*	-

2.3.2 Music Separation

Music separation is a similar task, with the added challenge of mixtures generally containing more than 2 distinct sources (unless performing speech denoising), meaning this would no longer be viable as a binary separation task as is common in speech models. One of the most common music separation datasets 'MUSDB18' contains music tracks which consist of overlapping sound stems that compose each song belonging to one of 5 defined classes: drums, bass, accompaniment, vocals and other. This means each input mixture will need to be disentangled into 5 predicted sound stems - a

challenging task. Although these instruments may be more distinct from each other than individual speakers (as speech samples may be considered to come from a single class), they naturally have a large proportion of overlapping harmonics and exist in similar frequency ranges (Woodruff et al., 2008), with potentially huge variance in amplitude/volume between instruments. Their strict structure regarding rhythm and tempo and complex composition of chords and other specific harmonic structures also makes long-term dependencies even more important to learn for effective separation.

A variety of models have tackled the task of music separation in recent years, including the highly efficient Spleeter model (Hennequin et al., 2019). They trained a fully convolutional U-Net architecture with a depth of 12 layers, 6 for the encoder and 6 for the decoder, with skip connections between layers to aggregate information across different feature map resolutions. The U-Net takes spectrograms as input and produces separated spectrogram predictions by applying soft-masking, which is a masking approach that allows overlapping sound energy between separated audio stems. This can introduce artifacts from other sources or noise but also helps ensure all the target sound energy is retained. Though they release only pre-trained models on a private dataset, they show fast and effective music source separation is possible.

The Demucs model was shown to produce strong results only operating in the time-domain, using a more complex implementation composed of an encoder-decoder architecture similar to a U-Net but using faster strided convolutions rather than explicit downsampling and inserting a bi-directional LSTM in the bottleneck, taking the lowest dimensional feature map as input from the output of the encoder, then passing its result to the decoder as input (Défossez et al., 2019). An improved version named Hybrid Demucs, due to its implementation training on both waveforms and spectrograms in parallel, achieved even greater performance and was considered a SOTA solution to the problem on the MUSDB dataset. However, their improvements come at the cost of an even greater increase in model complexity, needing careful alignment of the temporal and frequency branches of the U-Net to be stable (Défossez, 2022). This was taken even further with Hybrid Transformer Demucs (HT Demucs) (Rouard et al., 2022), through the introduction of transformers in the innermost layers of the U-Net architecture applying self-attention and cross-attention between the frequency and temporal domains. Their changes increased performance by a substantial margin of 0.45dB on their curated music separation dataset, showing the potential for improvement by incorporating transformers into a U-Net based separator.

A more recent SOTA approach to music source separation was introduced with the BS-RoFormer (Lu et al., 2023), a model which incorporates rotary position embeddings proposed in the original RoFormer (Su et al., 2023) architecture with a frequency band-split approach. A hierarchical stack of transformers processes inner and inter band sequences to learn band-wise features, without the drawbacks of RNNs which were originally used in this approach in Band-Split RNN (Luo and Yu,

2022). This results in an extremely powerful solution and impressive signal-to-distortion ratio (SDR) scores surpassing HT Demucs, even using a smaller implementation without the use of additional training data to complement the original MUSDB18HQ dataset. Even so, their solution is computationally expensive to train, requiring many GPUs and weeks of training time. A more efficient solution which also achieves SOTA results was SCNet (Tong et al., 2024), another multi-band solution which compresses audio spectrograms into 3 bands of similar information density through the pruning of empty or uninformative frequency bands. This addresses one of the key-issues of audio processing by reducing data size dramatically whilst preserving salient audio features needed for effective separation. Whilst their results may not surpass those of the largest BS-RoFormer architecture, they also require far less data and computational resources to achieve their excellent SDR scores.

2.3.3 Environmental/Universal Source Separation

Universal source separation is the ultimate challenge in audio source separation, as this seeks to disentangle any number of sources, from any number of classes, from a mixture. This is an extremely challenging task, meaning compromises are currently necessary to move the field closer towards this goal. One compromise is fixing the number of sources to separate, another is having a fully labelled dataset for a known set of classes. As with music source separation, mixtures may often contain more than 2 sources, eliminating denoising or binary masking as an easy option. Also, similar to speech separation, mixtures may contain multiple instances of the same class, leading to significant frequency overlap or difficulty differentiating distinct sources from each other. There are plenty of applications for universal source separation outside of speech denoising, which normally seeks to remove environmental sounds rather than extract them.

This includes:

- Threat detection and isolation by targeting classes such as gunshots, glass breaking, screaming etc.
- Research into wildlife such as the monitoring of animal populations, whether this be separating specific creatures from each other in a noisy wild environment (e.g. bird calls from lion growls) or separating animal sounds from the environment itself (e.g. whale song from ocean noise).
- Audio remixing through extraction of distinct sound stems from each source in a given piece of media, such as a movie, for post processing. This would be particularly useful for old media recorded in single channel, for remastering in multi-channel or surround sound, complimenting the wave of visual remasters on classic TV and movies.

At the time of our research, the SuDoRM-RF model (Efthymios Tzinis et al., 2020a) achieved close to or improved upon results from SOTA separation models such as Conv Tas-Net and Demucs on this task, whilst being much more efficient. The model takes waveforms as input which are encoded to a latent representation in a similar fashion to Conv Tas-Net. Following this, their architecture consists of U-Net style blocks, that successively downsample input whilst extracting information at each resolution, before upsampling again whilst aggregating information between each encoder-decoder layer with skip connections. The output of this block maintains the original temporal resolution of the original encoded representation of the waveform, which is then passed into the next block. The separator network is essentially a stack of convolutional U-Nets which repetitively extract features to produce masks for each source, which are element-wise multiplied with the original encoded input, before being decoded back into separated waveforms.

SuDoRM-RF was trained on the ESC50 environmental classification dataset, by taking 4 second crops of sounds from 2 different classes, downsampling to 8kHz (from 44.1kHz), performing augmentation and normalization to synthesise a training set of 20000 mixtures per epoch, with 3000 mixtures for validation. The ESC50 dataset is small and extremely varied with 50 different classes and only 40 distinct samples each, making this a challenging task, but the compromise of training for binary separation of the two sources makes this easier. Their model doesn't apply PIT to combat the permutation problem, or use attention to capture long-term dependencies, but their performance on USS is impressive considering how efficient the model is and served as our primary competition.

Recently, a new approach to universal source separation was introduced (Kong et al., 2023) that allowed their model to leverage a huge amount of weakly labelled environmental/arbitrary audio data contained in the AudioSet dataset (Gemmeke et al., 2017). This process involved several steps: sampling, anchor segment mining, embedding extraction, mixing of anchor segments, and finally query based separation using a residual U-Net (ResUNet) model. The first step involved sampling a balanced set of samples from AudioSet, which itself contains 527 classes and 5800 hours of weakly labelled audio data, orders of magnitude greater than ESC50 or even the free universal sound separation (FUSS) dataset (Wisdom et al., 2020), which are both small but cleanly labelled. However, within this dataset there is a huge imbalance of samples, where the speech and music classes dominate with over a million audio clips, and others such as environmental classes only have 10s of clips (Gemmeke et al., 2017; Kong et al., 2023). Therefore, balanced sampling is applied, ensuring each class is sampled from evenly across the dataset to reduce the dominance of speech and music over the training. They then use anchor segment mining with pre-trained sound event detection (SED) models to identify where the audio class occurs in the relatively long audio clip and extract the relevant segment, which they define as having a duration of 2 seconds. These segments have conditional embeddings extracted which are passed to the separation network along with the combination of the

segments in the form of an audio mixture, which is converted to a magnitude spectrogram before passing through their ResUNet.

Their residual U-Net architecture contains 6 encoder blocks, 4 bottleneck blocks, and 6 decoder blocks, containing residual convolutional blocks and skip connections between each encoder-decoder layer of the same depth. Their approach has helped to solve one of the most pressing problems in universal sound separation, which is the lack of strongly labelled data for training, especially in comparison to similar image or video-based tasks. This SOTA approach was further improved by applying a pre-trained self-supervised audio masked autoencoder (A-MAE) (J. Zhao et al., 2024) to extract rich features from audio input, which are concatenated with the original spectrogram input to the ResUNet model. This provides the separation network with a much stronger understanding of the acoustic patterns within the data prior to training the network to produce source masks without requiring strongly labelled data. These breakthroughs were made after pivoting to a new direction in our research investigating audio synthesis for class detection and dataset synthesis, but incorporating these techniques into our own separator model could yield additional performance due to the vast increase in training data.

2.4 Generative Audio Models

Audio synthesis has received much attention in recent years due to the large number of practical applications it can provide. Generative speech models can provide text-to-speech functionality to help the vision impaired and AI driven speech synthesis can be leveraged by smart assistants for more realistic sounding voices (Khanam et al., 2022). Generative music models can synthesise individual instruments for use in composition with specific timbre and effects (Lavault, 2023), or entire music tracks of a given genre or style (Dhariwal et al., 2020). Generative audio models can even be trained to synthesize environmental sounds for use as sound effects, whether curated specifically for foley in TV and movies (Ghose and Prevost, 2021) or creating soundscapes on the fly for use in media such as video games (Marrinan et al., 2024). Generative models can also be used in research for several purposes, one notable purpose being synthetic dataset generation (Lee et al., 2023), useful for tasks where original in-the-wild and strongly labelled data is scarce, such as that for universal sound separation. Some generative networks are also purposely designed to model a structured latent space, where similar encoded samples (generally from the same class) are clustered close together whilst being far from dissimilar samples, variational autoencoders being a prime example (Kingma and Welling, 2022). Generative adversarial networks (GANs) are another popular approach, where two networks are trained in tandem in an adversarial manner, where one attempts to generate realistic synthesised data, and the other attempts to identify whether a sample is fake (synthesised) or real. This training scheme encourages both networks to improve, as when one gets better at generation or discrimination, the other must improve to counter this. This has led to excellent results in both the

image and audio domain, though each generative approach has its strengths and weaknesses, such as variety or quality of generated samples, and stability of training due to problems such as mode collapse. With universal sound separation lacking large datasets and requiring much in the way of prior knowledge, we sought to explore how generative networks could be used to solve these problems.

2.4.1 VAEs

Autoencoders are networks that are designed to learn to encode data into a lower dimensional latent representation whilst preserving important information, then to reconstruct the higher dimensional original or target data using this representation whilst maintaining as much accuracy as possible (Li et al., 2023). Variational autoencoders are very similar, but instead of a deterministic mapping, the audio is encoded to obtain the mean μ and standard deviation σ for the sampling of a continuous probabilistic distribution such as a Gaussian, using a parametric inference model $q_{\emptyset}(z|x)$ such that the variational parameters $\emptyset$ in this encoder are optimized to approximate the posterior distribution $p_{\theta}(z|x)$ (Kingma and Welling, 2019). After sampling the distribution to obtain a latent vector z, this is passed into the decoder where it attempts to reconstruct the original input, minimizing the reconstruction loss (often MSE or BCE). To ensure the latent space is well structured and smooth, a KL divergence loss is also minimized to approximate the prior distribution $p(z)$ with the learned latent distribution as closely as possible. This formulation encourages the model to learn a low-dimensional representation of the training data and to group similar samples without supervision or conditioning, as well as enabling variation in generated output through random sampling of the latent distribution after training. This means VAEs can learn to reconstruct various classes of data without prior information whilst clustering these classes together, giving them the potential to learn the number and distribution of classes in a dataset.

RAVE is a popular audio VAE for its ability to generate high quality music output whilst remaining exceptionally efficient enabling real-time synthesis (Caillon and Esling, 2021), and serves as the foundation for much subsequent research in this area. There, they reduce input waveforms from a continuous signal across the entire spectrum of frequencies, to a multiband representation where each band contains a given range of frequencies within the signal. This passes as input into a convolutional encoder-decoder architecture which aims to reduce the spectral distance between the original and synthesised multiband audio representation, whilst ensuring a well-structured and compact latent space is produced. They achieve this compact representation both via downsampling input to a lower-dimensional vector and using a fidelity parameter that dictates the weighting of the reconstruction quality. A higher fidelity value results in a larger latent space to encode more information to improve quality, but small reduction in this weighting can reduce the dimensionality required in the latent space dramatically whilst preserving reasonable generation quality. To further improve reconstruction

quality, they utilize a second stage of fine-tuning through an adversarial process similar to that found in generative adversarial networks (GANs). The benefits provided by this adversarial training are likely one of the reasons GANs are more popular for the audio synthesis task than a purely vanilla VAE training approach.

2.4.2 GANs

Fundamentally, the goal of a GAN is to reduce the loss of both the generator and discriminator networks in parallel, with the joint goal of increasing the quality and realism of generated data. Generally, the generator network samples a latent vector from some distribution such as a normal gaussian and attempts to progressively reconstruct a batch of original samples with as much accuracy as possible. The discriminator is trained with ground truth samples, and separately with synthetic samples produced by the generator network, where it processes the data to predict whether a sample is real, often trying to minimize the average BCE loss. There are many formulations for training GANs, including non-saturating GAN loss, least-squares (L2) GAN loss (Mao et al., 2017a) and Wasserstein GAN (WGAN) loss (Arjovsky et al., 2017a), which each sought to help tackle common problems with training GANs such as saturation, vanishing gradients and mode collapse. Briefly, generator saturation occurs when a discriminator improves too quickly for the generator to be able to keep up, leading to a stall in training and being totally incapable of fooling the discriminator. Vanishing gradients are the phenomenon where generated samples are past the decision boundary of the discriminator to classify them as real, yet they are still very far from the original input leading to low quality reconstructions and samples (Mao et al., 2017b), as a result of using sigmoid cross entropy loss. Mode collapse occurs when a generator learns to consistently generate a subset of the data that fails to cover all modes, resulting in a lack of diversity in samples. This could be due to a number of reasons, such as a lack of capacity in the generator to learn to accurately capture the complexity of a broad range of data leading it to rely on specific modes to fool the discriminator, or the discriminator lacking capacity to identify the lack of diversity in the generated samples leading to no penalisation (Kossale et al., 2022a).

Although raw end-to-end generation using a randomly sampled latent vector from a distribution is the most common form of training GANs, within the audio domain there are many examples where mel-spectrograms are instead used as input to the generator network, such as HiFi GAN (Kong et al., 2020) and MelGAN (Kumar et al., 2019). The generator processes the mel-spectrogram, transforming and upsampling the input to output a waveform. Essentially, rather than generating a new sound, the model becomes a vocoder, reconstructing the original waveform using a low-dimensional perceptually motivated representation, without the benefit of the original phase information. These models are useful as a downstream application for improving the quality of generated mel-spectrograms when

converting back into the time-domain for use as real sounds but remain unsuitable for raw end-to-end generation.

WaveGAN (Donahue et al., 2019a) is one of the most well-known models for audio generation, a variant of the deep convolutional generative adversarial network (DCGAN) (Radford et al., 2016), a fully convolutional network proposed for use in the unsupervised image generation task. They propose two models, WaveGAN for end-to-end waveform generation, and SpecGAN which aims to synthesize mel-spectrograms from mel-spectrogram input. For evaluation of the audio quality of mel-spectrogram outputs, they are first inverted to the linear STFT, after which the Griffin-Lim algorithm (Griffin and Lim, 1984) is applied to estimate the missing phase information and recover the waveform. Their results show that their SpecGAN converges faster and produces samples of greater distinguishability evidenced by human labellers having more accuracy on these as opposed to samples produced by WaveGAN, capturing greater variance between the data. However, when evaluating the quality of the samples, WaveGAN won out, though this is likely at least partially a result of the lossy nature of the Griffin-Lim algorithm introducing distortion. This problem can be mitigated to an extent with neural vocoders such as HiFi GAN, making mel-spectrograms viable as an option for low-dimensional input for faster training.

Since the introduction of the DCGAN class of model, vast improvements were proposed by the newer StyleGAN (Karras et al., 2019) models, which provided not only improved image quality, but greater variation and fine-tuned control over the generated output. This is through the use of an intermediate mapping layer, that takes a random noise z and through an 8-layer multi-layer perceptron (MLP) maps it to a new latent vector w, that is more disentangled than the original latent space. Rather than feeding this directly into an input layer, a learned constant is processed by ‘style blocks’ consisting of convolutional and adaptive instance normalization (AdaIN) layers. The new latent vector w is passed through an affine transform for conversion to ‘styles’ which AdaIN layers take as input for control over the generated samples. They also introduce uncorrelated Gaussian noise injection for stochastic variation of fine details in the output images before each AdaIN layer. In the context of image generation of human faces, this may include variation in small details such as freckles and hair.

Although diffusion models have become popular for generation tasks in recent years including for audio (Božić and Horvat, 2024), the StyleGAN architecture has been shown to challenge and even surpass the results of these networks for speech synthesis, as shown by ASGAN (Baas and Kamper, 2022), an unsupervised StyleGAN model that takes audio pre-processed into either mel-spectrograms or HuBERT (Hsu et al., 2021) audio features, which are learned discrete units of audio produced by a pre-trained HuBERT model that are disentangled from noise and common factors between speech samples, leading to an increase in generated speech quality. Their architecture is similar to the original StyleGAN, with the addition of a Fourier feature layer for encoding the input to the generator to

produce cosine features, and low pass filter layers to prevent high-frequency artifacts introduced through non-linearity from allowing the generator to fool the discriminator, leading to a deterioration in synthetic audio quality. They achieved SOTA results on the SC09 dataset for spoken digit generation (Donahue et al., 2019a) over WaveGAN and diffusion-based model DiffWave (Kong et al., 2021). However, their model is far more complex than WaveGAN, leading to generation speed being almost 3 times slower. Regardless of individual performance, both WaveGAN/SpecGAN and ASGAN can generate high quality audio samples without the need for supervised training in the form of conditioning using ground truth labels or guiding with known audio features, making both suitable for our purposes of automatic class detection and synthesis.

Chapter 3

3 U-AxialNet: Axial Transformer Augmented U-Net for Computationally Efficient Universal Sound Separation

3.1 Introduction

Audio separation is fundamentally a segmentation problem, a task found in other domains such as images and video. Generally, convolutional networks achieved SOTA results in separating different classes within an image (e.g. a dog and a car) by extracting features to apply a mask to the original image returning only the target. Naturally most image-based targets are homogenous clusters of pixels meaning unless there is obfuscation or occlusion of the target this task can be relatively simple. These models are often trained with smaller resolution images as these relationships and recognizability of the classes is maintained at smaller data sizes, which leads to faster and more efficient training of the models. Whilst the data is different in many ways compared to images, audio separation is essentially the same problem, separating groups of data points from others to disentangle one entity (in this case, a sound source) from others. Universal audio separation is the task of being able to separate a sound source belonging to any arbitrary class of sound from any other, in a mixture of unknown (and theoretically infinite) numbers of distinct sounds. The practical use of this as a tool cannot be understated. One use for a model capable of this could be its integration into hearing devices for masking background noise and focusing on specific targets that a user selects such as speech and alarms but could be dynamically activated and deactivated at will without the need to retrain the model. Another would enable the remastering of old single channel audio, through the high-quality extraction of each audio source for restaging in a stereo or surround sound format as multi-channel audio. This could be used to compliment the remastering of visuals from old media such as movies for modern technology, giving an even greater sense of immersion from both the visual and auditory perspective.

The premise is simple: separate one distinct class object/entity from all others and any ambient noise or background information, which in the context of speech could be isolating a single speaker in a noisy environment or from other distinct sound sources. This can be exceptionally useful if the target and the remainder after separation is already known and well understood and is the only point of

interest in the given data, leading to highly accurate and clean separation due to the simplicity of the training set and the need to only tune the model to mask the mixture for a single class. This gets more complicated when the target and remainder in the mixture are similar (e.g. multiple speakers), yet strong solutions (Chang et al., 2019; Isik et al., 2016; Yu et al., 2017b) have been created that achieve high fidelity separation. However, universal audio separation remains the 'holy grail' of audio separation, due to the boundless general utility it could provide, yet is limited by its complexity and scarcity of training data. Furthermore, audio separation models are also computationally expensive, especially when operating over high-dimensional raw waveforms. We propose a model to leverage the efficiency and performance of the convolutional U-Net architecture in combination with a memory efficient implementation of the recently popularised self-attention mechanism in transformers for capturing long-term dependencies to perform arbitrary audio source separation.

3.2 Related Work

3.2.1 Universal Sound Separation

At this point in our research, there had been a small number of attempts to create universal audio separation models using deep learning, such as the SuDoRM-RF (E. Tzinis et al., 2020), which trained both speech and non-speech versions of their model as a direct comparison between standard binary speech separation, and binary universal separation. This is somewhat contradictory, as a truly universal audio separator would be capable of more than binary separation in the supervised manner SuDoRM-RF employs, but with the available tools this a reasonable compromise for the ability to separate a number of different highly complex classes of data from each other, especially when the availability of data is so low. This is yet another reason as to why a truly universal audio separator would have great practical significance, as it would provide not only excellent utility in several applications but also help to alleviate the data scarcity faced in this and other tasks that would use any number of different audio classes, such as audio classification or super-resolution.

SuDoRM-RF achieves excellent results in comparison to previous separators, through the application of a U-Net like architecture of several downsampling and upsampling blocks, first introduced in U-net: Convolutional networks for biomedical image segmentation (Ronneberger et al., 2015). After passing the waveforms through an encoder to yield a latent representation of the mixture, these blocks are used to successively extract features at different resolutions through downsampling and upsampling the input, aggregating the information across representations with the help of skip-connections, as used in Conv-TasNet (Luo and Mesgarani, 2019). Their solution is effective, however their convolutional architecture isn't designed to specifically capture long-term dependencies, whilst still requiring a huge amount of memory for training due to the number of layers needed for multiple U-Net style blocks.

3.2.2 Transformers in Segmentation

Around this time, computer vision saw a leap forward in the form of the Vision Transformer, commonly referred to as ViT (Dosovitskiy et al., 2021), which took the transformer model often used in natural language processing (NLP) tasks, and applied it to computer vision, with unprecedented results for image classification (Dosovitskiy et al., 2021). What followed was a mass-adoption of the concept, with others producing their own versions of the vision-transformers for other tasks such as segmentation (Cheng et al., 2021; Liu et al., 2021a). For speech separation, the SepFormer (Subakan et al., 2021b) was proposed, consisting of an encoder, masking network and decoder architecture. Unlike previous models, their approach is entirely transformer oriented, the backbone of which consists of multiple SepFormer blocks each containing an 'IntraTransformer' for capturing short-term dependencies within the chunked audio data independent of other chunks, and an 'InterTransformer' that captures long-term dependencies between the chunks. Their approach achieved SOTA results in speech separation, surpassing convolutional models such as SuDoRM-RF and RNN based models like DualPathRNN which employ a similar chunking process for learning dependencies. This highlighted the potential for totally transformer-based separation architectures to achieve high-quality audio separation without drastic changes to the approach. However, this performance increase is computationally expensive and time-consuming to train, requiring either high capacity or multiple GPUs for training due to self-attention on the high dimensional input across multiple attention heads and transformer blocks.

In the similar task of medical image segmentation, another approach was taken that combined the computational efficiency and segmentation power of the convolutional U-Net with the long-term dependency modelling ability of transformers, resulting in a hybrid CNN-Transformer U-Net architecture named U-Transformer (Petit et al., 2021). Their model takes a standard CNN U-Net backbone and injects a multi-head self-attention transformer (MHSA) at the lowest level of downsampling to capture long-term dependencies between the highest-level feature maps of the data. The skip-connections are also augmented with multi-head cross attention (MHCA) layers, which take the lower-level feature maps in the encoder stage and use cross-attention to evaluate the relevance of high-resolution information, adjusting the weighting to only propagate relevant information for producing the masks. This combination of a MHSA bottleneck and MHCA skip connections are shown to increase performance over the purely convolutional approach, whilst leveraging the power of the multi-resolution feature extraction of the U-Net without an entirely transformer based more computationally expensive approach.

3.2.3 Efficient Transformers

As transformers have become more popular, efficiency was a large concern due to the sheer number of parameters needed to model the data with self-attention and demand for computing resources such

as memory. This spurred the creation of efficient transformers, designed to maintain the global receptive field that the transformer provides as one of its main advantages over CNNs, whilst reducing computational demand significantly. Transformer models using the traditional attention mechanism have complexity of the form $\mathcal{O}(N^2)$ where N is the sequence length of input, which in the case of 2D input is equal to $N = hw$ where h is the input height and w the width. In the case of uncompressed audio waveforms, for a 5 second clip sampled at 16KHz, the sequence length would be 80000, meaning the computational complexity in these transformer models would equal 6.4 billion. This leads to huge memory requirements and models consisting of 100s of millions of parameters, making training and inference on single-GPU or low-capacity hardware completely unfeasible. Feature extraction techniques for lower dimensional representations such as encoders or transforms, such as conversion to mel-spectrograms, can reduce the sequence length significantly, however this doesn't address the core issue. This is because even after reduction of the audio to its principal frequency components, the quadratic complexity factor makes scaling the solution to longer audio clips difficult without significant compression or loss of information.

Approaches to reduce the complexity of the attention mechanism itself include incorporating neural memory (Lee et al., 2019), kernels (Choromanski et al., 2022), downsampling (Jaegle et al., 2021; Liu et al., 2021b) and low-rank methods (Wang et al., 2020), each with their upsides and drawbacks but in general decreasing computational complexity by a significant factor. Another common complexity reduction technique is using fixed-patterns or combination of fixed-patterns of attention, such as the Sparse Transformer (Child et al., 2019) and Axial Transformer (Ho et al., 2019). Rather than attending to all data samples in the input with respect to all others, fixed patterns reduce the coverage of attention over the input whilst maintaining a global receptive field to capture long-term dependencies. Using a combination of these patterns helps reduce the information loss incurred through using a sparser attention matrix by combining learned information across different sets of samples. The Sparse Transformer combines patterns of strided and local attention, but their implementation requires custom GPU kernels which restricts the ease of its implementation. The Axial Transformer has no such restrictions and is straightforward to implement, whilst granting a significant complexity reduction to $\mathcal{O}(N\sqrt{N})$. This is implemented with a combination of two fixed patterns of attention along each axis of a two-dimensional input. This allows the transformer to learn correlations between data on a specific axis independent of the other axes. In the context of an audio mel-spectrogram 2D input, long-term correlations between different frequency bands in specific time windows can be learnt (x axis attention) whilst simultaneously learning the correlations between energy at given frequency bands over the duration of the audio clip (y axis attention).

3.2.4 Hybrid-Transformers

As research into source separation matured, there was exploration of combining transformer layers with existing U-Net approaches, such as Hybrid Transformer Demucs, which outperformed the original Hybrid Demucs model by 0.45 dB (Rouard et al., 2022). They maintain their 4 outermost convolutional layers in the encoder and decoder of their U-Net, replacing the innermost 2 encoder and decoder layers with a cross-domain transformer, for processing both waveforms and spectrograms in parallel for improved separation. Each transformer layer consists of a simple self or cross attention and feed-forward block with normalisation typical of the literature, combined with LayerScale (Touvron et al., 2021) for stabilisation. They use depths of either 5 or 7 layers of alternating self-attention and cross-attention layers to build their transformer bottleneck. This allows the model to capture long-term dependencies within and between each domain without needing to perform attention on data of huge dimensionality, mitigating some of the computational strain added through the introduction of transformers, similar to our approach. However, their method uses a traditional transformer, which still incurs a large computational cost due to attention being applied across each entire data sample through a global receptive field. As this closely aligns with our own approach of a U-Net transformer hybrid architecture, we compare our results with this in addition to other baselines in speech and non-speech models.

3.3 U-AxialNet

At the time of this research, most audio source separation networks relied on purely convolutional or purely transformer-based architectures, where the former could struggle to capture long-term dependencies, and the latter was computationally expensive. Inspired by both the global attention mechanism for capturing long-term dependencies in transformers and the performance provided by U-Net architectures in the context audio source separation, this work investigates the combination of both which we call U-AxialNet, illustrated in Figure 3.1. We combine transformer blocks with a convolutional U-Net encoder-decoder architecture for the task of audio source separation, learned through the minimisation of negative scale-invariant signal-to-distortion ratio (SI-SDR) loss (Roux et al., 2018). Unlike SepFormer whose transformer architecture follows a similar design to the original (Vaswani et al., 2023), we opt for a more efficient fixed pattern transformer which requires fewer computational resources due to the attention mechanism not computing every data point for each attention.

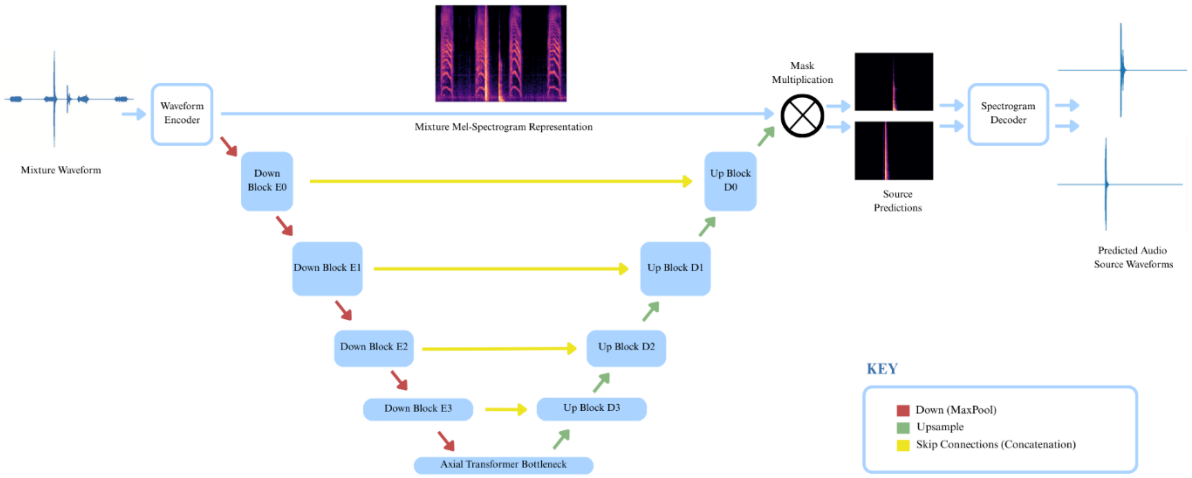

Figure 3.1: U-AxialNet architecture.

3.3.1 Waveform Encoder

Input to the model consists of raw audio waveforms of mixtures sampled from the dataset of the form $X \in \mathbb{R}^{B,1,T}$ where B is the batch size and T is the number of samples in the time domain. For memory requirement reduction in the model, we test training with both direct transformation to a mel-spectrogram representation $X' = MelScale(STFT(X))$ and a learned 2D representation using an encoder consisting of a single convolutional layer and ReLU activation $X' = ReLU(Conv1D(X))$, where $X' \in \mathbb{R}^{B,1,F,L}$ after adding an extra channel dimension, F is the number of frequency bands and L is the number of frames in the time domain. For audio sampled at 16kHz, the convolutional layer matches the mel-spectrogram transform with a kernel size of 1024 and stride of 512. For ease of computation during downsampling and upsampling, the time dimension of the 2D audio representation is padded to a target length with trailing zeroes that can be repeatedly halved to the required depth without rounding required, following the algorithm:

$$while\ targetlength\ \%\ 2^N\ !=\ 0\text{:}\ targetlength\text{+=}\ 1$$

where N is the depth of the U-Net and $targetlength$ is initialised to the initial size for input's time dimension. For example, a 2D audio representation with shape [128, 77] with initial $targetlength = 77$ would be padded to [128, 80].

3.3.2 U-Net Encoder

Our U-Net encoder consists of a stack of convolutional blocks for downsampling the input and extracting features at different resolutions. Each convolutional block consists of n stacks of Conv2D layers, with a batch norm applied to the output and finally ReLU activation to ensure non-negativity.

The initial input $e_0 = X'$ is the output from the initial encoder layer or mel-spectrogram transformation. The output from each stack in each downsampling block therefore is $b_n = ReLU(BatchNorm2D(Conv2D(b_{n-1})$, where n is the total depth. The output of each block can be represented as $e_i = Down_i(\mathrm{E}_i(e_{i-1}))$ for $i = 1, \dots, N$ for N downsampling blocks, where $\mathrm{E}_i(e) = (\prod_{j=1}^{n} b_{i,j})(e)$ and $Down$ is achieved with a max pooling operation of kernel size 2. In each $Conv2D$ layer we used a kernel size of 3, stride 1, padding 1 and the number of output channels from a block is double the input channels. Therefore, after the initial block the dimensionality of the output feature map follows $e_i \in \mathbb{R}^{B,2\times ch_{i-1},\frac{F_{i-1}}{2},\frac{L_{i-1}}{2}}$.

3.3.3 Axial Transformer Bottleneck

Embedded between the encoder and decoder of the U-Net is an axial transformer bottleneck, which aggregates information from axes independent attentions to learn long-term dependencies across the entire input sequence. First, a positional embedding is added to the input $h_0 = e_N + Pos(e_N)$ which are a sum of $C \times H \times 1$ and $C \times 1 \times W$ embeddings for the rows and columns respectively. The self-attention mechanism follows the standard practice definition of $Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, where Q, K, V are the query, key and value matrices separately linearly projected from the input and $d_k = \frac{dim_k}{heads}$, which is the dimensionality of each key or query per attention head. Normally a transformer requires the input to be a 1D sequence of tokens, which requires 2D input such as images or mel-spectrograms to be linearised in raster order. However, axial attention computes attention along a single axis at a time, meaning the input $h \in \mathbb{R}^{B,C,F,L}$ is reshaped for attention on permutations of $h \to h_f \in \mathbb{R}^{B\times L,C,F}$ where each row is a token and $h \to h_l \in \mathbb{R}^{B\times F,C,L}$ where each column is a token, treating the 2D input as two separate 1D token sequences. First, layer normalisation is applied and attention is computed along one axis, then a residual connection combines this with the original input. This is used as input for the other axis where this is repeated. Layer normalisation is applied a final time before being fed into a multi-layer perceptron (MLP) and a residual connection combines the final learned attention matrix with the MLP output. This facilitates learning dependencies between data along each axis and between the axes themselves, and the non-linear transformations through the MLP help learn more abstract features. Each step in the axial attention block can be represented mathematically by 3 separate equations, the initial attention $a_1 = h_i + Axial_f(LayerNorm(h_i))$ feeds into the second attention $a_2 = a_1 + Axial_l(LayerNorm(a_1))$, which is then processed by the MLP for the final output $h_{i+1} = a_2 + MLP(LayerNorm(a_2))$. The MLP is a simple combination of Conv2D layers sandwiching a ReLU activation layer acting as a feed forward block. These axial attention blocks can be stacked J times whose output is $h_{i+1} = \mathrm{A}_i(h_i)$ given $i = 0, \dots, J-1$, yielding h_J as input for the U-Net decoder. This process for an individual block of axial attention with input h_i is illustrated in Figure 3.2.

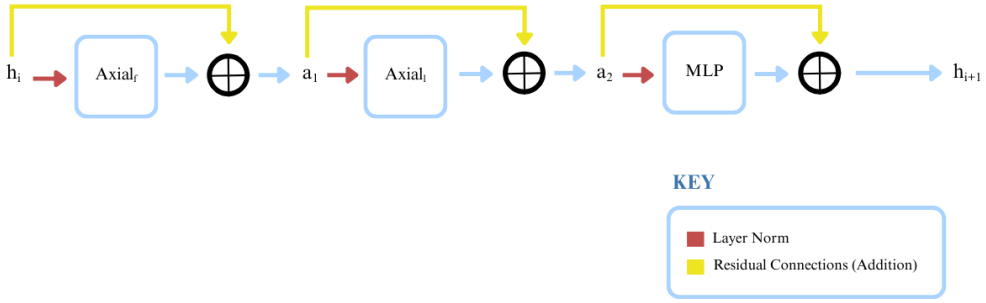

Figure 3.2: Axial attention bottleneck block.

3.3.4 U-Net Decoder

Initial input for the U-Net decoder is the direct output from the axial transformer bottleneck, where $d_N = h_J$. The structure of the decoder follows the same but inverted architecture of the U-Net encoder, with N total convolutional blocks. Before passing into block $\mathcal{D}$, the input is first upsampled to double the input resolution using an upsampling layer via linear interpolation, then mirroring the encoder is passed through the same number of stacks of convolutional blocks, where the channel dimension is halved. A skip-connection s is then used to fuse low-level features in the encoder at the same resolution depth using concatenation. We also propose replacing these skip connections with an axial cross attention layer which is explained in depth in Section 3.3.6, though this would require more GPU memory capacity than we have available to test due to the huge number of points to attend to, even with efficient axial attention. The output of each block is therefore $d_i = \mathcal{D}_i(Up(d_{i+1})) + s_i$ for $i = N-1, \dots, 0$ where $d_i \in \mathbb{R}^{B,\frac{ch_{i+1}}{2},2F_{i+1},2L_{i+1}}$.

3.3.5 Mask Separation and Spectrogram Decoder

Output of the U-Net d_0 passes through a final Conv2D layer to generate the time-frequency masks, where the number of out channels is equal to the number of sources we wish to separate. So, for model input of single-channel audio in a time-frequency representation $X' \in \mathbb{R}^{B,1,F,L}$ our masks are of the form $\widehat{M} \in \mathbb{R}^{B,S,F,L}$, where S is the number of sources to separate. These masks pass through a softmax operation, then they are applied to a copy of the original input mixture using element-wise multiplication to perform soft-mask separation $\widehat{Y'} = X' \odot \widehat{M}$, yielding $\widehat{Y'} \in \mathbb{R}^{B,S,F,L}$ containing the estimated sound sources in the time-frequency domain. Any padding with trailing zeroes added after the initial encoder step are now removed, returning the predicted sources to the length of the original mixture. These estimated sources are then converted back to the time-domain using a decoder layer $\hat{Y} = ConvTranspose1D(\hat{Y}')$ which inverts the original encoder layer with equal kernel size, stride and padding. To invert predictions from input obtained directly via the mel-spectrogram transform $\hat{Y} = iSTFT(InverseMelScale(\hat{Y}'), Phase_X)$, with the option of using the original mixture phase to

avoid lossy inversion using the Griffin-Lim algorithm. These predictions are finally evaluated against the ground-truth waveforms using permutation invariant SI-SDR loss.

3.3.6 Axial Skip Connections

We also propose an advanced version of our model through the introduction of axial cross-attention skip connections. Between each resolution depth of the U-Net, skip connections typically fuse the output of the encoder block with the output of the decoder block via concatenation. However, this higher capacity model uses axial cross attention to leverage information from higher resolution features in the encoder to guide the decoder in constructing a mask on the most relevant information. Therefore, the query matrix Q is linearly projected from positionally encoded decoder output $d'_i = Pos(Up(d_{i+1}))$ and the key and value matrices K, V are projected from the positionally encoded encoder output $e'_i = Pos(e_i)$. With these matrices, axial attentions on the frequency and time axes are computed as before but separately rather than one feeding into the other, and the output of each is additively combined with the decoder output to fuse lower-level features from the encoder with the decoder at each level such that $s_i = AxialCrossAttn_f(d'_i, e'_i) + AxialCrossAttn_l(d'_i, e'_i)$ for depth i, relating back to our original decoder output $d_i = \mathcal{D}_i(Up(d_{i+1})) + s_i$. This is much more computationally expensive than our current formulation and impossible for us to train with our current equipment but could represent a huge increase in separation performance through the combination of high-resolution features and sparse attention to capture long-term dependencies at every feature map resolution.

3.4 Experiments

3.4.1 Datasets

We train our model on separate experiments with speech and non-speech for comparison on our model's performance between the two tasks.

3.4.1.1 Speech Separation

For the speech separation task we use Libri2Mix (Cosentino et al., 2020), which is derived from noise-free audio sources time-stretched using the Librosa library between factors of 0.8 to 1.2 to avoid repetitions, useful for training models on binary speech separation tasks. We choose to train with the train-360-clean variant consisting of 364 hours of speech split between 927 speakers and perform validation and testing with the development and test sets consisting of 5.4 hours of speech from 40 speakers each. Training with this relatively large dataset of samples with high amounts of shared frequency information and semantic content between speakers represents a theoretically simpler task than the more challenging problem of environmental or universal sound source separation with datasets of huge sample diversity and relatively small size.

3.4.1.2 Environmental Separation

For the environmental separation task, we train with mixtures generated from the recently released Free Universal Sound Separation (FUSS) dataset (Wisdom et al., 2020), composed of 357 sound classes with a total duration of 23 hours of audio clips natively sampled at 16KHz. This is relatively large for an environmental audio dataset, yet 10 times smaller than LibriMix. FUSS consists of a wide array of classes, from dogs to speech, glass-breaking to car-engines. This means the classes are extremely distinct from one another, where some exist almost entirely in lower frequency bands as a mostly repetitive signal, and others are highly dynamic, with large shifts in pitch and volume at different moments in the sample and complex long-term frequency relationships. Class labels are not available for these mixtures, but our proposed model doesn't incorporate conditional information for generating masks, eliminating this lack of labelling as a limiting factor.

3.4.2 Training Procedure

3.4.2.1 Hyperparameters

In all our experiments, we train with a learning rate of 0.001 using the Adam (Kingma and Ba, 2017) optimiser. The U-Net encoder-decoder blocks each consist of double stacks of convolutional layers with batch normalisation and ReLU activations at a depth of 4 blocks for downsampling and upsampling, and the axial transformer bottleneck has a depth of 2 attention blocks, each with 8 attention heads. We train using an Nvidia GeForce RTX 3080 8GB GPU, representing a computational capacity for training much smaller than that used to train other models such as SepFormer (Subakan et al., 2021c) using full self-attention and SuDoRM-RF (E. Tzinis et al., 2020) whose inference is efficient but training requires a huge amount of memory. With these parameters, we train using a batch size of 4.

3.4.2.2 Loss Functions

When training our model, negative SI-SDR (Roux et al., 2018) is used as our loss function and for quantitative evaluation of the model's performance. Traditional SDR was the standard metric in the past, but variations in permutation and scale of the separated sound source compared to the ground truth led to erroneous or misrepresentative scores (Roux et al., 2018). SI-SDR remedies this issue through normalization of the predicted source projected on to the target source for the least overall error between the two, thereby eliminating the effect of scaling as the two are now scale-aligned. This ensures strong SI-SDR scores cannot be achieved without the estimated signal matching the target closely in terms of both magnitude, shape and phase. This is computed on waveforms, meaning predicted separated sources in the frequency-time domain must be converted back to the 1D time-domain, hence the inversion of these sources in our model using either a decoder or inverse mel-spectrogram transform.

We also train our model separately using MSE loss, which is computed by taking the mean squared error between the target and separated audio sources. This means in theory it can be used directly on the predicted audio mel-spectrogram sources and the ground truth targets, allowing us to train our audio separator more like an image-segmentation model, avoiding erroneous results due to lossy inversion recovering phase with the Griffin-Lim algorithm or degraded quality through the final decoder layer back to a waveform.

3.4.2.3 Experiments

We train our hybrid model consisting of a vanilla convolutional U-Net architecture with the incorporation of our axial transformer bottleneck. To evaluate our model, we compare our results with a vanilla U-Net architecture and SuDoRM-RF using 2 U-Net style blocks for similar model capacity requirements on speech and environmental source separation for mixtures containing 2 sources and/or background noise. All experiments use the same hyperparameters and model depth where applicable for fair comparison. Unfortunately, we can't perform experiments using our proposed cross-attention skip-connections due to hardware restrictions.

3.5 Results and Discussion

In our initial experiments, we trained both the vanilla U-Net and our U-AxialNet on the FUSS dataset for 20 epochs, where loss consistently plateaus. Despite repeated attempts to improve performance through hyperparameter changes and convolutional block depth, our model fundamentally couldn't learn to generate masks for separation, resulting in the extremely low SI-SDR scores in Table 3.1. We can see this is in stark contrast to scores produced by models such as SepFormer in the speech task and Hybrid Transformer Demucs in the non-speech task, both of which improve upon previous entirely convolutional approaches through the introduction of transformers. The lower depth SuDoRM-RF performed better than both our convolutional log mel-spectrogram based U-Net and U-AxialNet, though their scores are still relatively poor in comparison to the expected performance for strong separator networks, including their full depth implementation which is much more computationally demanding. This indicates that both our model and the shallower implementation of SuDoRM-RF with 2 U-Net blocks are likely suffering from capacity issues. However, although their model is unable to effectively separate sources given their score, there also appears to be a fundamental flaw in the training scheme using the non-linear and heavily dimensionally reduced mel-spectrograms, which we investigated.

Table 3.1: SI-SDR scores (lower is better) for models trained on Libri2Mix and FUSS

Model	SI-SDR	
	Speech	Non-Speech
SuDoRM-RF (small)	2.25	0.188
U-Net	-35.5	-48.3
U-AxialNet	-37.7	-48.9
SuDoRM-RF (full)	17.02	8.35
Sepformer	**20.5**	-
Hybrid Demucs	-	8.34
Hybrid Transformer Demucs	-	**9.00**

Visualising an example of the source predictions for a given mixture from Libri2Mix in Figure 3.3, we can see the model has identified one source contains more sound energy in the lower frequency bands, whereas the other contains high amplitudes in a much wider frequency range. However, the model is unable to capture any of the fine features of the timbre or shape of the audio, resulting in a flat mask to separate low frequency energy distributed sharply along the axes, likely due to the axial attention bottleneck, and another to capture a more dynamic range, though this is heavily smoothed. Subjective evaluation leads to heavily distorted and perceptibly unintelligible audio. This further indicates a problem in both generalisation capacity of the model and compression of the audio features through the mel-spectrogram transform and downsampling process leading to overly smoothed and low-resolution masking. In contrast, SuDoRM-RF's masking process is far less aggressive, leading to source predictions which are simply slight variations of the original mixture. Subjective evaluation shows a very small amount of disentanglement, though this is mainly a small change in perceived volume of each source rather than true separation, leading to their low SI-SDR scores.

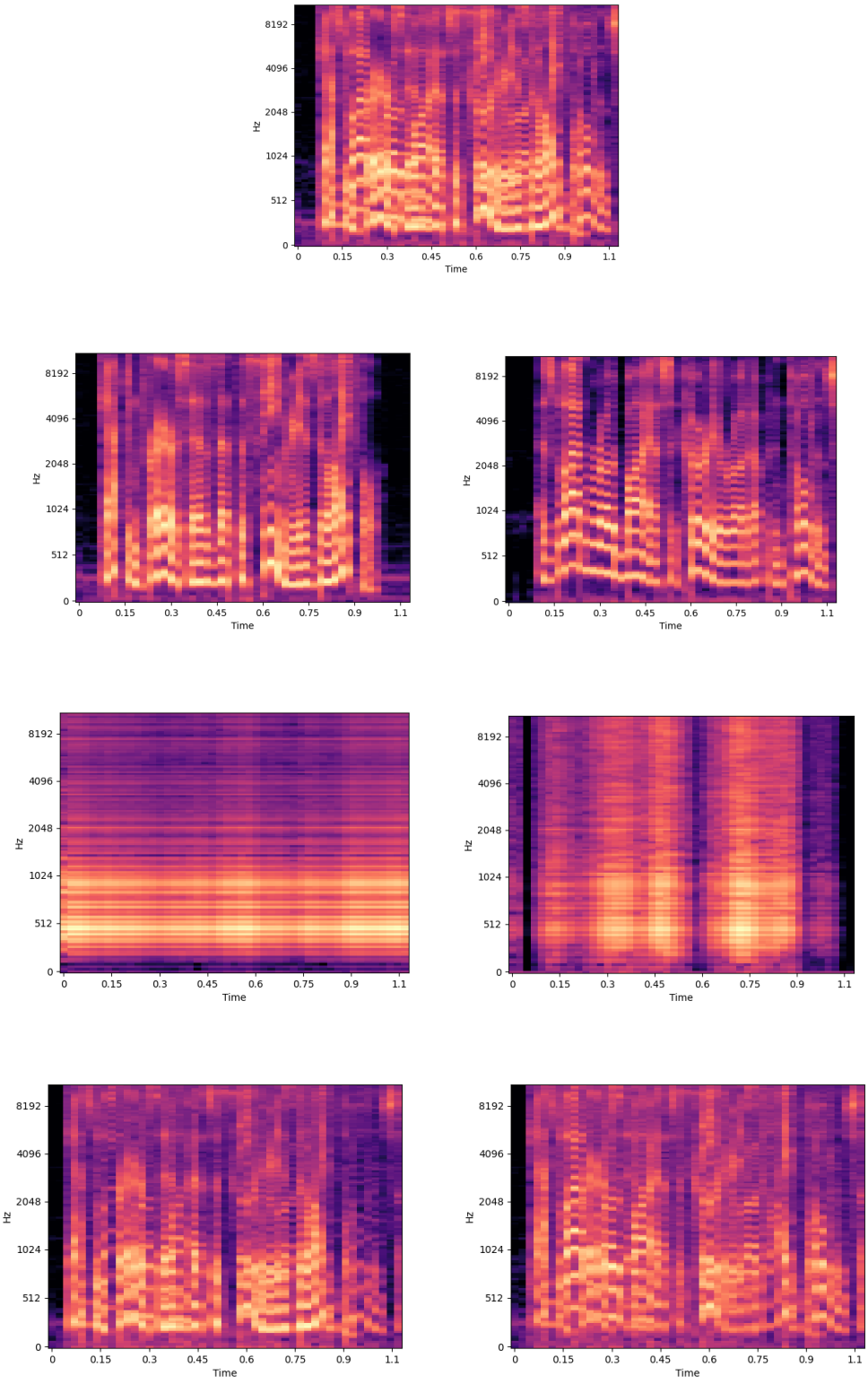

Figure 3.3: Audio source log mel-spectrogram predictions using U-AxialNet (third row) and SuDoRM-RF (bottom) of ground truth sources (second row) contained in mixture sample (top) from Libri2Mix using.

Therefore, to investigate the degree to which mel-spectrograms impact training in comparison to low model-capacity, we retrain our model using linear-scale spectrograms. Although this increases the

dimensionality of the input significantly, this provides the benefit of learning from simpler linear input, retention of more frequency and phase information for waveform reconstruction and remaining 2D for valid use of our axial-attention bottleneck. First, we verify the model can easily learn to separate a single mixture using this scheme through direct source prediction shown in Figure 3.4, which captures sharp high-frequency detail and cleanly disentangles sources with little frequency smoothing. There remains a low level of background noise, but subjective evaluation shows this is perceptually minor, leading to strong separation. Repeated experiments on different samples indicate SI-SDR scores between predicted and ground truth sources of >25dB, which is an extremely strong score.

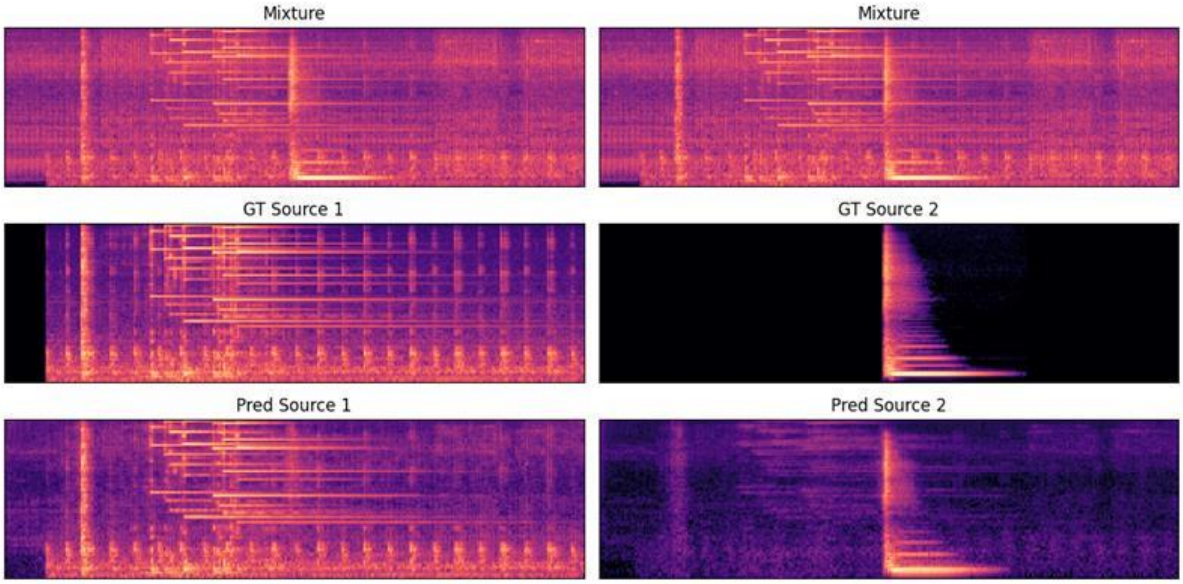

Figure 3.4: Single-mixture separation test on FUSS dataset using linear scale spectrograms with U-AxialNet.

Knowing the model can learn to capture these fine details using linear scale spectrograms, we retrain our model both with and without the axial-attention bottleneck to evaluate limitations of model capacity on the FUSS dataset. To eliminate erroneous results and scores through phase alignment during waveform reconstruction, we train directly using MSE loss between real and predicted spectrograms, as the core task of the model is to predict the magnitude spectrogram, with waveform inversion being a post-processing step that can be improved with methods such as vocoders or learning phase reconstruction separately, which we don't address in this work. The results in Table 3.2 indicate similar performance in the task, with our axial attention bottleneck slightly outperforming our purely convolutional U-Net model.

Table 3.2: MSE scores training vanilla U-Net and U-AxialNet on FUSS dataset.

Model	MSE Score
U-Net	0.020
U-AxialNet	**0.017**

For subjective evaluation of performance, we visually compare the predicted sources to the ground truth of the same mixture we tested earlier in the single-sample experiment between the two models. In Figure 3.5, we can immediately see a sharp decrease in prediction quality for both models. Although this is to be expected when comparing to the predictions for an overfitted single-sample experiment, the disentanglement is still weaker than expected for 100 epochs of training. Between the two models, we see our axial bottleneck introduces less smoothing and captures finer detail, whereas the convolutional network overly smooths the predictions. Overall, the predictions clearly suffer from being unable to ignore the background noise of the original mixture, which itself could be considered another distinct source, unable to cleanly denoise and simultaneously separate the sounds.

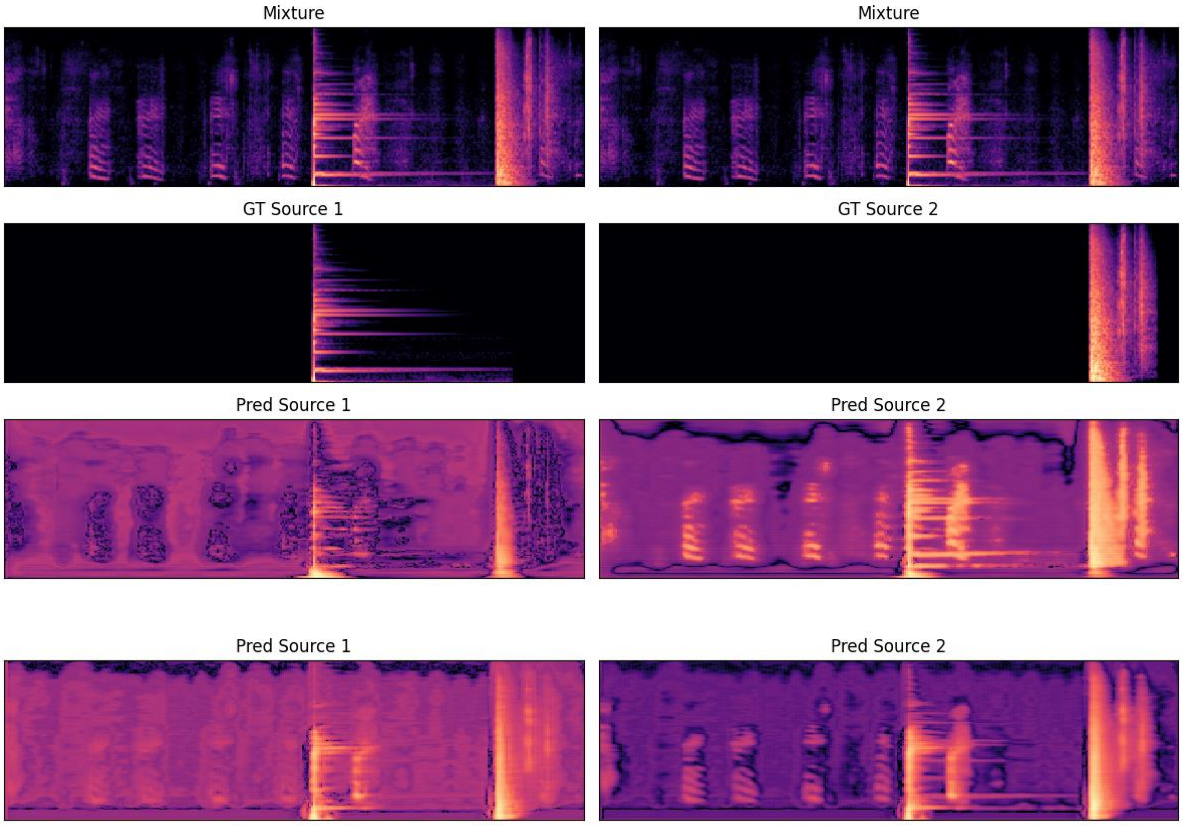

Figure 3.5: Source predictions on mixture (top) from FUSS dataset after 100 epochs using U-AxialNet (third row) and convolutional U-Net (bottom) for ground sources (second row).

Based on these experiments with our models and initial results with the low-capacity implementation of SuDoRM-RF, we conclude the key issue with our model in its current state is a lack of learning capacity for generalisation.

3.6 Conclusion

We proposed a model to address the high-computational demands of training sound separation models by compressing the data input size using the log mel-spectrogram transform and incorporating axial attention to capture long-term dependencies. Through our experimentation, this architecture was unable to perform effective sound source disentanglement, producing results far worse than we would expect on baseline datasets such as LibriMix. Due to resource constraints our model is shallow and designed for efficiency rather than performance, meaning several improvements could be made to partially address this issue in future work. This could include multiple U-Net blocks similar to the full SuDoRM-RF model whilst incorporating axial attention bottlenecks into each or between blocks, at the cost of much greater computational capacity cost. We could also extend our model to operate in the time-domain for direct end-to-end disentanglement of audio features which seems to provide stronger performance (Luo and Mesgarani, 2019; Subakan et al., 2021c), though again with increased memory requirements due to data size. Either way, the axial attention approach itself may be translating poorly to the time-frequency domain, exacerbated by the loss of frequency and phase information in the log-mel spectrogram representation we trained on, leading us to believe deeper networks with full transformers trained on either linear scale spectrograms or hybrid time and time-frequency networks to be more suitable for tackling universal sound separation.

Due to hardware limitations and changes in circumstances regarding the overall project, we pivot our work towards researching methods that could indirectly improve source separation through the automatic discovery and labelling of classes within audio datasets, which could then be used as conditioning information. Several studies have found incorporating class-conditional embeddings into music separation (Slizovskaia et al., 2019) and universal sound separation (Efthymios Tzinis et al., 2020b) models achieve improved performance over those without conditioning. However, this either requires training on a strongly labelled audio separation dataset, or the use of a pre-trained classifier network on a large and labelled audio dataset. Relying on this kind of prior information and supervised approach reduces the capacity for generalisation in universal sound separation, leading us to explore methods for automatic classification of audio classes using unsupervised generative methods in Chapter 4.

Chapter 4

4 Unsupervised VAEs for Semi-Supervised Multi-Label Prediction using KL-Divergence in the Latent Space

4.1 Introduction

Universal sound separation is a challenging task, a challenge that is compounded by data scarcity, hardware requirements and pattern complexity. Larger datasets such as the free universal sound separation commonly referred to as FUSS (Razzaq, 2020) contain no labelling for use as a conditional embedding to improve separation performance. Furthermore, most sound separation systems require prior knowledge of the number of sources in the original mixture to separate. Traditional classifiers can work very well at predicting a given sample belongs to a specific class within a dataset, however they often require strongly labelled datasets to accomplish this and don't generalise well to unseen examples, assuming all classes appear in the training set, known as the 'closed world' assumption (Shu et al., 2018). In the context of audio source separation, multiple audio classes are entangled and if we wanted to know which classes these were and how many, the classifier would need to predict an unknown number of multiple labels on input it is unlikely to have seen in its training set, given the set of possible mixtures is practically infinite. To mitigate these limitations, we explore the potential to leverage the power of variational autoencoders (VAEs) to model the latent space of a dataset in a structured and well-defined way, to automatically detect the likelihood a sample audio mixture belongs to each of the classes in the latent space and how many classes are likely within the mixture by computing their KL-divergence scores between each class probability distribution. This labelling could then theoretically be used to guide the separator in generating more accurate masks for disentanglement.

The VAE is a generative model, whose goal is to take input, encode it into a distinct latent representation drawn from a distribution (e.g. gaussian), which is then reconstructed into the target. A well-defined latent space of several different classes should intuitively maximize the distance between samples from different classes and minimize the distance between those of the same class. This can be visualized using techniques such as principal component analysis (PCA) or the t-distributed stochastic neighbour embedding (t-SNE) algorithm. Standard autoencoders are similar in that they reduce the

dimensionality of the input to a latent representation, but this is mapped to a fixed point in the latent space. This means its task is simply to take input, perform dimensionality reduction to capture the most important features of the data, then decode this into a reconstruction that matches the original as closely as possible. However, this makes interpolation in the latent space impossible, as it is designed to reconstruct the original data and not new unseen instances. For our purposes, this isn't suitable, as we need a continuous latent space from which we can derive probabilities that a sample is drawn from a given space in the distribution. Not only would this be helpful in identifying target samples in a mixture through low KL-divergence values but could also give a good indication of whether the mixture is composed of unseen data in the training set, with every class having similarly high KL-divergence scores.

4.2 Related Work

4.2.1 Unsupervised Audio VAEs

There have been several attempts at applying the VAE algorithm to the task of audio synthesis, seeing excellent performance in unsupervised speech reconstruction (Lu et al., 2022; Oord et al., 2018) with the early implementation SpeechVAE (Hsu et al., 2017) trained on mel-spectrograms of real-speech from male and female speakers to learn their latent representation, enabling speech reconstruction and voice conversion between different speakers whilst maintaining the original phonetic content. However, their approach is specifically speech based and the approach is rather basic, surpassed by more modern approaches for better quality reconstruction. Music reconstruction (Engel et al., 2017) and timbre transfer (Bitton et al., 2018) have seen success with VAEs, demonstrated by great performance in MusicVAE (Roberts et al., 2019) which addresses the complexity of the latent space in music by applying the VAE process but with a unique approach to segment audio data in the decoding stage using what they call a 'conductor'. Their architecture is comprised of bi-directional LSTM layers to encode the audio capturing long-term dependencies and a hierarchical two-part RNN based decoder for reconstruction on embeddings of separate measures or bars of music, which are sequentially decoded and combined to form the output audio reconstruction. However, this approach is only possible using MIDI audio files, which unlike standard waveforms contain a vast amount of specific information on notes played for each instrument, making this unsuitable for general use. Jukebox (Dhariwal et al., 2020) is an extremely powerful VQ-VAE, that discretizes the latent encoding and is capable of producing much longer music audio sequences than previous approaches. However, the model is computationally expensive, and the latent representation isn't continuous and drawn from a learned distribution, but rather a sequence of vector-quantised tokens, making it unsuitable for our purposes. The RAVE (Caillon and Esling, 2021) model was shown to be capable of both high-quality general audio reconstruction from a compact latent space on speech and music data, whilst being fast and efficient. They take waveforms as input on which they perform multiband

decomposition using PQMF (Nguyen, 1994) to reduce temporal dimensionality and encode the audio to sample a 128-dimensional latent vector. This is fed into a decoder similar to the generator network in MelGAN (Bitton et al., 2018), which is modified to generate 3 separate vectors, the multiband waveform, an amplitude vector and noise. They also train the VAE decoder in conjunction with a discriminator for adversarial training to improve reconstruction quality. This makes for a far more lightweight solution than Jukebox whilst encoding continuous latent representations for which the KL-divergence can be easily computed, but still takes a great deal of time for training and has an unnecessarily complicated decoder scheme for our purposes. The spec-VAE used in LyricJam is part of a two-stage training network for generating lyrics conditioned on input spectrograms. The model first trains the spec-VAE for high-quality reconstructions and a well regularized latent space, as the vectors of means and standard deviations calculated from the encoded input are used as conditioning information in training their text-CVAE. This requires stage 1 to produce meaningful latent encodings to be useful as class information, meaning disentanglement is necessary. As our goal is to produce a disentangled and well-structured latent space in the audio domain, we adopt a similar approach, using log-mel spectrograms as input and a similar architecture to their spec-VAE.

4.2.2 Multi-Class Labelling with VAEs

Conditional VAEs (Sohn et al., 2015) incorporate supervised embeddings projected from prior information such as class labels to guide generation and inform the structuring of the latent space. However, a study (Wang et al., n.d.) showed that even when incorporating class-conditional information, the combined latent space centred at a mean of 0 can ignore this when forming, leading to an entangled latent space with little class-specific clustering. They attempted to mitigate this issue using a combination of k-means clustering to eliminate the need for conditional labels, instead training the model to capture the categories of data automatically, whilst enforcing class specific clustering by computing the kl-divergence between a sample and its nearest cluster and assigning this as its label, refining the cluster means with each training step. Although this produced a more disentangled latent space with isolated clusters, this requires the clusters to automatically capture the correct semantic content requiring well defined diverse classes and knowing the number of classes in advance of training, both adding constraints and prior information dependency to training. C-GMVAE (Bai et al., 2022) showed that by mapping conditional class label embeddings to a mixture of individual latent Gaussian distributions and using contrastive learning, labels that commonly occur together (i.e. many samples may be labelled both “sea” and “beach”) have their means centred closer together, whereas those that appear less often are pushed apart. Using a fixed class embedding during testing, they use their model for predicting probabilities of a class label for the given input. Whilst this technique is powerful for processing, reconstructing and structuring a latent space for multi-class input, it heavily depends on conditional class information to achieve this. Furthermore, it doesn’t utilize the latent space directly and must fully reconstruct the input data before passing this in

combination with a fixed class embedding through a sigmoid function for this prediction. In our work, we will explore the potential for inferring the likelihood audio samples belong to a given class in a semi-supervised fashion, requiring only a very small amount of labelled data for comparison.

4.3 VAE Model Training

4.3.1 Model

To assess class capture in the latent space, we train a VAE consisting of a simple encoder-decoder architecture on log mel-spectrogram data. For efficiency with training and computational requirements, this is built using convolutional blocks. Our implementation is similar to the spec-VAE in LyricJam (Vechtomova et al., 2021) and (Vechtomova et al., 2020), with some additional improvements including skip-connections, similar to our U-AxialNet architecture minus the axial-attention and double convolutional blocks resulting in the MelSpec U-Net VAE. We also test without skip-connections to investigate the degree to which the model can encode and structure the space without additional information from high-resolution representations earlier in the network, as a pure MelSpec VAE.

4.3.1.1 Encoder

Our MelSpec VAE encoder is composed an initial encoding convolutional layer, then 4 double convolutional downsampling blocks to extract features at different resolutions, comprised of 2 stacks of Conv2D layers, followed by batch normalisation and ReLU activations. A 2D mel-spectrogram x is passed into the downsampling blocks of the encoder with model weights ϕ, whose output is flattened and passed into separate Linear layers with out channels set to the dimension of the latent space, to compute the mean and log variance parameters:

$$[\mu(x), \log \sigma^2(x)] = Encoder_\phi(x)$$

to approximate the posterior distribution:

$$q_\phi(z|x) = \mathcal{N}\left(z; \mu(x), diag\left(\sigma^2(x)\right)\right)$$

4.3.1.2 Latent Code Sampling

Sampling the continuous latent variable $z \sim q_\phi(z|x)$ isn't done directly, as we need the process to be differentiable for backpropagation, thus the reparameterization trick (Kingma and Welling, 2022) is applied such that $z = \mu(x) + \sigma(x)\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$ is auxiliary noise. This is passed through another Linear layer to project the latent encoding to the initial flattened feature map produced by the final layers in the encoder for upsampling by the decoder.

4.3.1.3 Decoder

Our VAE decoder mirrors the encoder architecture, with 4 double upsampling blocks of ConvTranspose2D, batch normalisation and ReLU activation layers, ending in a final convolutional decoding layer. The latent representation z is passed through the decoder and combined with embeddings at each layer in the encoder from skip connections s (if enabled) to obtain the reconstruction $\hat{x} = Decoder_{\phi}(z, s)$ or $\hat{x} = Decoder_{\phi}(z)$.

4.3.1.4 Datasets

Three different datasets were used in separate experiments to model the latent space of speech and non-speech data, for potential further use in both audio separation tasks. To train for reconstructing speech data, we use the LibriSpeech dataset consisting of 360 hours of clean speech data from male and female speakers at a split of roughly 50:50. This is originally stored as waveforms, recorded at a sampling rate of 16 kHz and a duration of 6 seconds per audio file. The audio is single-channel by default, meaning each waveform consists of around 96000 samples; for use in this experiment, this size needs to be reduced significantly. This is one of the largest labelled audio datasets available, meaning it is suitable for training a baseline VAE to determine how effectively the latent space is structured for different genders in unsupervised training.

We also train on the SC09 (Donahue et al., 2019b) spoken digits dataset for the speech task, consisting of different speakers of different genders' utterances of the digits 0 through 9. This is semantically similar to the MNIST (Lecun et al., 1998), but much greater in size, consisting of one-second single-channel waveforms sampled at 16kHz totalling 16000 datapoints, vs small monochrome images of height and width 28, totally $1 \times 28 \times 28 = 784$, making this task much harder. Experiments with SC09 are to determine if the VAE can disentangle and cluster semantic content close together rather than focusing on the timbre relating to speaker identity as with LibriSpeech, where we consider each of the digits 0-9 separate classes.

ESC50 on the other hand is composed entirely of distinct and varied classes of isolated audio samples. This means that in theory, if the VAE can be trained to understand the data, the latent space should be structured effectively with each class being closely clustered together with little overlap. However, ESC50 consists of only 40 samples per class, with a much larger 50 classes in total. Each of these samples are 5 seconds in length, giving a total of only 2000 recordings coming to a total duration of 10000 seconds or less than 3 hours. Several particularly distinct classes were hand-picked for a better chance of stable training and useful results, which further reduces the amount of training data the model can learn from, meaning data augmentation was used to help alleviate the problem. This includes pitch shifting between up or down 2 semi-tones, or time stretching between 0.8x to 1.2x the duration of the original audio using the Librosa library. These ranges are chosen to preserve the patterns in the data and semantically important information whilst introducing realistic variance.

Unlike in spatial problems like image-based tasks, removing random sections of the data can lead to severe degradation in the quality of the signal and cause it to lose its meaning or recognizability, as audio is continuous and sequential.

4.3.1.5 Data Pre-Processing

For our MelSpec VAE, rather than modelling the latent space directly from waveforms, the training data is reduced to a much smaller size using the short-time Fourier transform (STFT) and applying the mel-scale to convert it into a logarithmic scale in-line with human hearing. This 2D representation provides faster and more efficient training on the same data, whilst allowing the model to extract key features during the encoding process to structure the latent space for effective reconstruction. This also provides access to a visual sanity check on the quality of the data that can be easily done with images to determine what the model may or may not be struggling to learn or reconstruct, which isn't obvious by observing a raw waveform. As high-quality reconstructions are not the priority of training our model as we seek to utilize the latent space itself, artifacts and distortions through inverting the mel-spectrogram are not a concern here. For improved training using MSE-loss, we normalise our mel-spectrograms using z-score (Anggoro and Supriyanti, 2019) normalisation to a mean of 0 and standard deviation of 1.

4.3.1.6 Loss Function

The overall VAE loss function is composed of two separate terms, KL (Kullback and Leibler, 1951) divergence which is a regularization loss, and reconstruction loss. For a well-structured latent space, there needs to be continuity and completeness, meaning that sampling from points close together in the latent space should yield similar results, and that these results make sense given the training objective. For example, when training with multiple audio classes such as bird song and train sounds, sampling from the latent space around one cluster should consistently return similar sounds, either the sound of birds or the sound of trains. If sampling similar points in the latent space and either the class is seemingly random or the quality is so degraded it's unrecognisable from the original dataset, then the model has failed to structure the latent space distinctly.

Minimizing the KL-divergence ensures that the distributions are close to a normal distribution, with a mean close to 0 and the covariance matrices close to the identity such that $\mathcal{N}(0, I)$. This keeps the encoded representations of the data close together to ensure meaningful content is sampled from around the latent space through the overlapping of distributions, where mixes of different classes can be sampled through interpolation. The KL-divergence between the output distribution $q(z|x)$ and the prior distribution $p(z)$ can be computed in the form $D_{KL}(q(z|x) \parallel p(z)) = \frac{1}{2}\sum_{j=1}^{L}[{\sigma_j}^2 + {\mu_j}^2 - 1 - \log {\sigma_j}^2]$ across L latent dimensions.

The second loss term is concerned with the quality of the decoded (or reconstructed) latent representation, which is the actual synthesis result of the generative network $\hat{x}$. This captures the overall similarity between the training data and the generated data by calculating the distance between them. This can be defined as $\mathbb{E}_{q_\phi(z|x)}[logp_\phi(x|z)]$ for model weights ϕ yielding the log-likelihood a sample x given latent code z.

This combination of regularization and reconstruction loss in the context of VAEs is also known as the negative evidence lower bound or ELBO loss, defined as:

$$\mathcal{L}_{ELBO} = -\mathbb{E}_{q(z|x)}[logp(x|z)] + D_{KL}(q(z|x) \parallel p(z))$$

We calculate the negative log-likelihood reconstruction term using MSE loss, such that:

$$-\mathbb{E}_{q(z|x)}[logp(x|z)] \propto MSE(x, \hat{x})$$

Therefore, our final training objective becomes minimizing the sum of the mean-squared error and KL-divergence, for the ELBO formulation:

$$\mathcal{L}_{ELBO} = MSE(x, \hat{x}) + D_{KL}(q(z|x) \parallel p(z))$$

The goal of the VAE is to reconstruct high quality data that resembles the original training set, whilst properly structuring the latent space. However, minimizing one loss often increases the other (Asperti and Trentin, 2020a), as ignoring the regularization for better reconstruction quality can allow the model to overfit the data with a more specific and fragmented latent space. Equally, focusing more on structuring the latent space can lead to poor reconstructions through factors such as too much overlapping (Asperti and Trentin, 2020b), bearing little to no resemblance of the original data. Therefore, the main challenge during training is balancing these two loss terms for stable training, ensuring that on average across the epochs both are decreasing steadily. β-VAE (Higgins et al., 2017) introduces a weighting factor β to the KL-divergence term to address this issue and found increasing the factor improved latent space disentanglement and model performance. However, some have found these improvements difficult to reproduce in standard training (Fil et al., 2021), instead finding improved performance through decreasing this factor. Therefore, we train our model with a KL weighting factor β with values less than or equal to 1, relying on hyperparameter tuning for stable and balanced training for structuring more than overly biasing the KL term. During our experiments we found we ran into the KL-vanishing problem (Fu et al., 2019), where the decoder learns to ignore the latent codes when reconstructing the samples, due to poor initial latent encoding early in training and pressure from the KL divergence term. To mitigate this, we implement a KL-annealing scheme for gradually ramping up the β term to a maximum value, as this has been shown to help in preventing the KL-vanishing problem (Bowman et al., 2016; Fu et al., 2019). This makes our final unsupervised loss term:

$$\mathcal{L}_{ELBO} = MSE(x, \hat{x}) + \beta \times D_{KL}(q(z|x) \parallel p(z))$$

4.3.1.7 Hyperparameters

Our model is trained for up to 20 epochs on LibriSpeech and up to 200 epochs on smaller datasets SC09 and ESC50, with early stopping enabled if validation loss does not improve. A learning rate of 0.0001 using the Adam optimiser with a batch size of 16 provides the most stable training whilst still achieving fast convergence. The split across all datasets is 80/20 for training and validation respectively. The data is pre-processed requiring no transformations to the 2D mel-spectrogram format during training. We repeated the experiment with different latent dimensions to explore the effect this has on how well structured the latent space is across different datasets; these include dimensions of 4, 8, 16, 32, 64, 128 and 256. We found little difference in clustering between the latent dimensions and far worse performance at lower latent dimensionality due to extreme compression of rich frequency features and temporal information, meaning most of our final experiments maintain a latent dimension size of 128 or 256. This allows more features to be learned and mapped from the input to the embedding, leading to greater reconstruction quality. However, the goal of the model is to discover the key features relevant to the data, prioritising those that most differentiate the data from each other. Therefore, the smallest latent dimension that preserves training quality and a continuous and smooth latent space is desirable, preventing us from increasing this further. Although we experimented with projecting the initial input channels to a greater number of base channels, we found an initial encoding to a base of 32 channels to be stable whilst providing enough capacity for learning and is therefore our setting for the experiments in this chapter.

4.4 Semi-Supervised Multi-Class Labelling

We assume a well-structured latent space is formed on a sufficiently pre-trained unconditional and unsupervised VAE model through the minimization of the KL-divergence for regularisation and reconstruction loss for intelligible and distinct audio sampled from the latent space, enabling training on larger unlabelled datasets. Post-training, we can now take a small sample of labelled audio data $\{(x_i, y_i)\}$ with $y_i \in \{0,1,\dots,U-1\}$ for U known classes that each belong to at least one class u in the training set. First, we take this small set of audio samples and encode each class of samples $\{ x_i^{(u)}\}$ into the latent space

$$q_\phi\left(z\middle|x_i^{(u)}\right) = \mathcal{N}(z; \mu(x_i^{(u)}), diag(\sigma^2(x_i^{(u)})))$$

separately to estimate a latent class conditional distribution $q^{(u)}(z) \approx N(\mu^{(u)}, diag({\sigma^{(u)}}^2)$ where $\mu^{(u)}$ is the average of the means across each sample in the class subset

$\mu^{(u)} = \frac{1}{N_u}\sum_{i=1}^{N_u}\mu_i^{(u)}$. We then take our new test samples and encode them independently such that $q_\phi(z|x_{test}) = \mathcal{N}\left(\mu(x_{test}), diag\left(\sigma^2(x_{test})\right)\right)$. Now given our independent class-specific distributions, we can compute the KL-divergence between them and each test sample's latent distribution to calculate the distance between them. Using the standard equation for KL-divergence between two multivariate gaussian distributions with diagonal co-variance, for our test sample and class specific distributions we get:

$$D_{KL}(q_\phi(z|x_{test})\|q^{(u)}(z)) = \frac{1}{2}\sum_{j=1}^{L}\left[\frac{{\sigma_{test,j}}^2}{{\sigma_j^{(u)}}^2} + \frac{(\mu_{test,j} - \mu_j^{(u)})^2}{{\sigma_j^{(u)}}^2} - 1 - \log\left(\frac{{\sigma_j^{(u)}}^2}{\sigma_{test,j}^2}\right)\right]$$

Taking the negative KL-divergence between an encoded sample and all prior latent class distributions, we can compute the most likely class through an argmax operation, where the least negative score is the most likely class. We can easily extend this to multi-label estimation by choosing the top-k scores. To prevent erroneous class estimation, we can use a threshold below some value to filter out unlikely classes, which could also facilitate detection of out-of-distribution samples or classes.

4.5 Results and Discussion

4.5.1 MelSpec U-Net VAE

Our experiments on the LibriSpeech360 dataset are with skip connections enabled. Our VAE quickly converges within 5 epochs of training, on both a dataset split of 80/20 and 50/50 for training and validation sets as shown in Table 4.1.

Table 4.1: Validation loss for U-Net based MelSpec VAE on LibriSpeech dataset with different train/validation splits.

Epoch	Validation Loss	
	80/20 Split	50/50 Split
1	41.67	67.91
2	4.91	13.34
3	0.21	1.68
4	0.11	0.23
5	0.10	0.29

This leads to near perfect reconstructions of the initial log-mel spectrograms and minimisation of the KL-divergence between the encoded distributions and a standard gaussian distribution $\mathcal{N}(0, I)$. We visualise the latent space of the encoded speech, using the gender ID to assign class labels 0 (blue) and 1 (orange) for male and female speakers. As shown in Figure 4.1, the latent space is more structured by the final epoch, but speakers of different genders haven't been disentangled clearly, with a huge amount of overlap and spread.

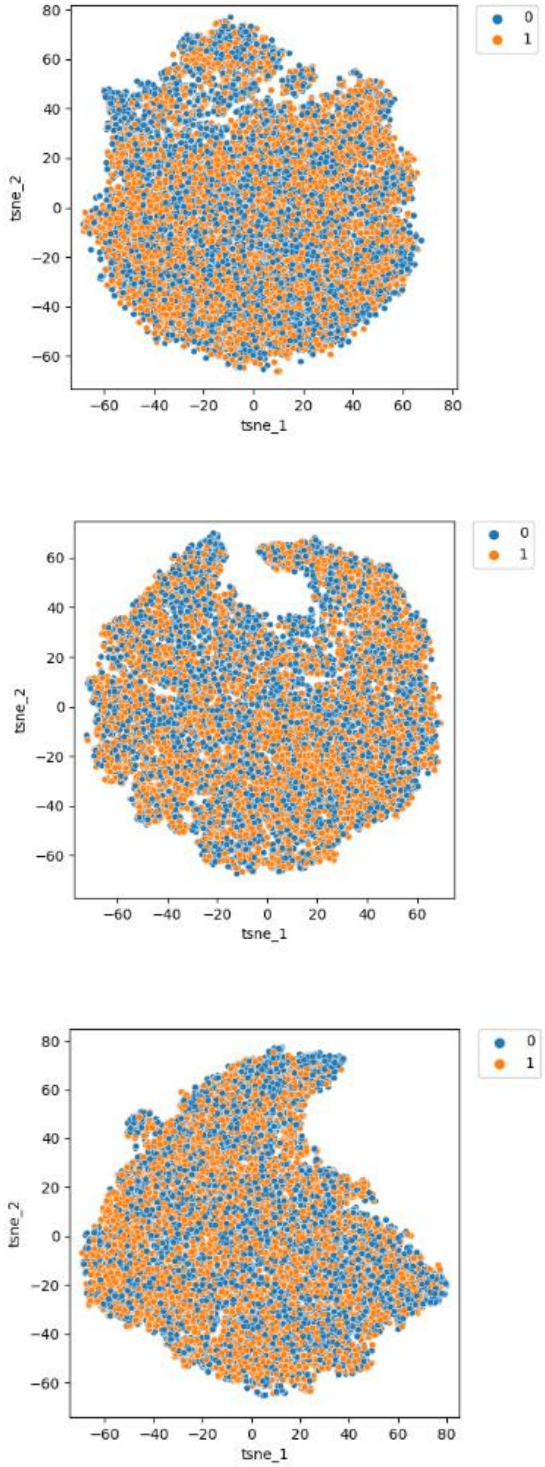

Figure 4.1: Latent space of encoded LibriSpeech plotted via t-SNE at the beginning of training (top), during (middle) and after (bottom).

Based on these results, we make several modifications to our experiments. First, we disable skip-connections to eliminate the possibility of the VAE decoder to rely on high-resolution features preserved before the latent encoding is obtained by the bottleneck, as reconstructions should depend mainly on the low-dimensional representation. In turn, this encourages the model to effectively

encode the most distinguishing and perceptually important features required to reconstruct initial input. We also train on smaller datasets with more distinct classes. For speech, we train to disentangle semantic content within the overall class of speech in the form of spoken digits. For environmental sounds, we train to disentangle many overall classes with high intra-class variance between samples.

4.5.2 Pure Mel-Spectrogram VAE

After disabling the skip-connections, first we verify the VAE can meaningfully encode and decode mel-spectrogram input as a regular autoencoder. After training to minimise reconstruction loss to plateau with a β value of 0.0 to ignore the KL-divergence loss term, we obtain reconstructions of reasonable quality as shown in Figure 4.2, with some smoothing and loss of sharpness in the higher frequency bands but overall retention of the temporal structure.

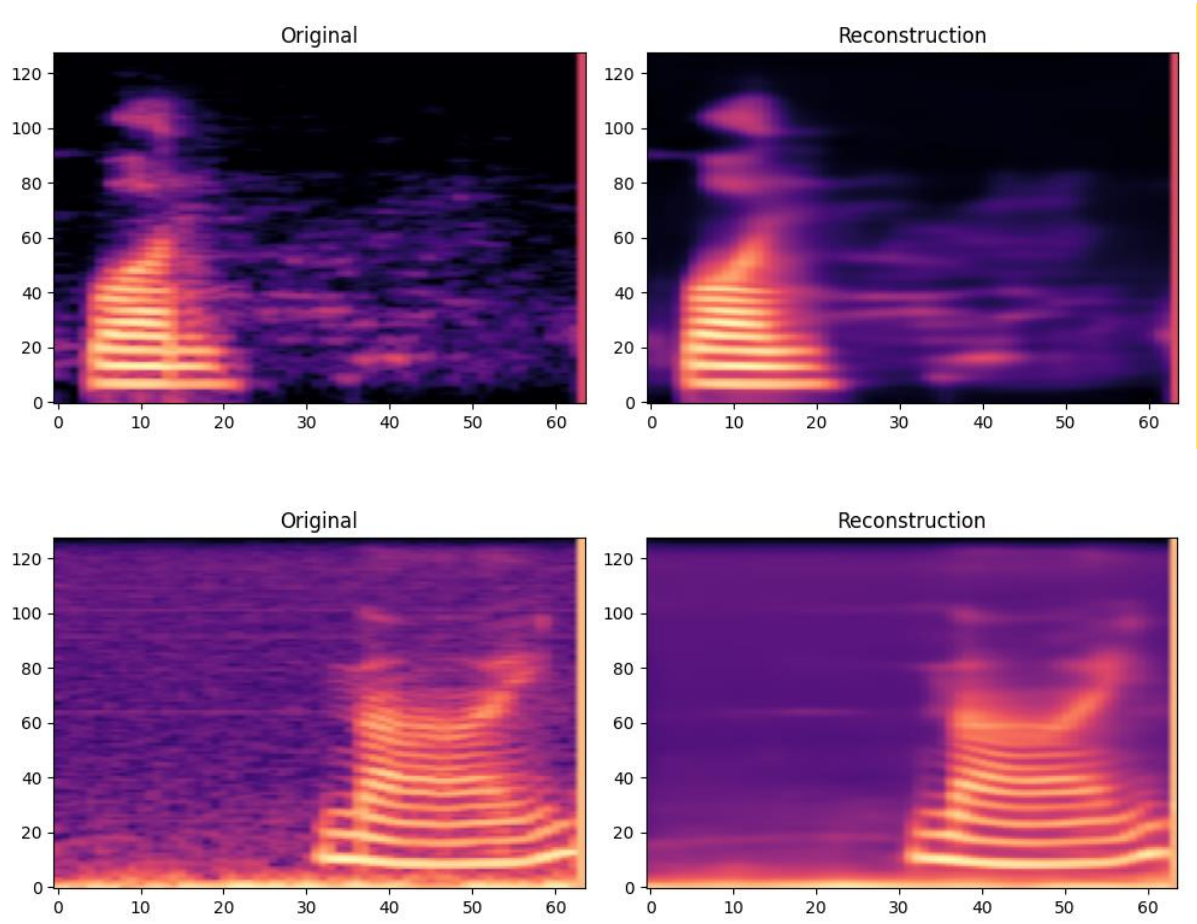

Figure 4.2: Log Mel-Spectrogram of real audio (left) and audio reconstructions (right) using pure autoencoder.

We slowly re-introduce the KL-divergence term with a β value of 0.3 which is gradually reached through 30 epochs of KL-annealing, to prevent overwhelming the reconstruction loss too quickly and halting learning through over regularisation of the latent space. Following previous experiments where even a very small β factor can overwhelm reconstruction loss, we warm-up the VAE by training as an autoencoder for 100 epochs.

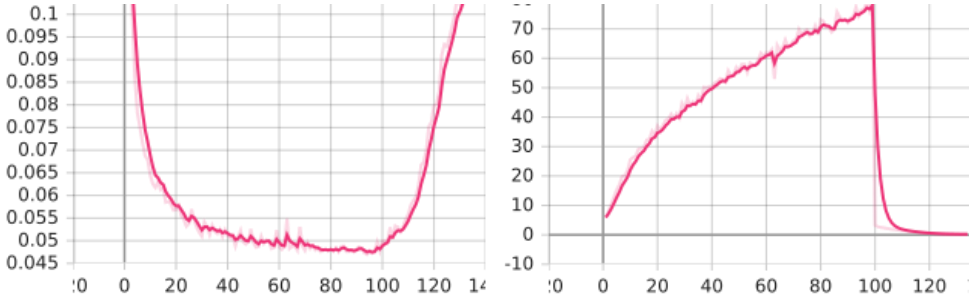

Figure 4.3: Loss curves for reconstruction (left) and KL-divergence (right).

Figure 4.3 shows a steady decrease in reconstruction loss and gradual improvement in mel-spectrogram quality until the β starts increasing from 0.0. Naturally, the KL-divergence loss gradually increased until epoch 100, indicating the VAE first learns fixed latent representations diverging from a continuous gaussian distribution. Before the maximum β value of 0.3 is even achieved, the KL-divergence loss is minimized to near 0 after 20 epochs of KL-annealing. In contrast, by the time the maximum β is reached, the MSE loss has more than doubled from its lowest value 0.047. The KL-divergence loss cannot be decreased further and with this regularisation imposed the reconstruction quality can't improve, leading to an end in training. The reconstruction in Figure 4.4 shows the temporal structure is preserved and finer details are starting to be captured, though there is significantly more smoothing in the lower frequency bands compared to our pure autoencoder reconstructions, which could be reduced through further autoencoder training.

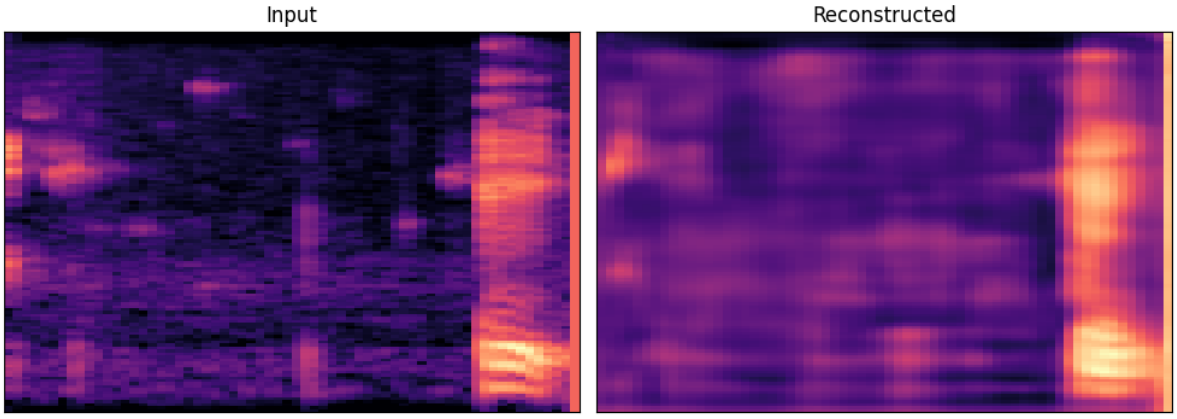

Figure 4.4: Real audio input (left) log mel-spectrogram vs the VAE reconstruction (right) with maximum β=0.3.

Using t-SNE (Maaten and Hinton, 2008) we visualise the latent space in Figure 4.5, which shows the latent space conforming to a 0 centred normal distribution with some partial clustering taking place.

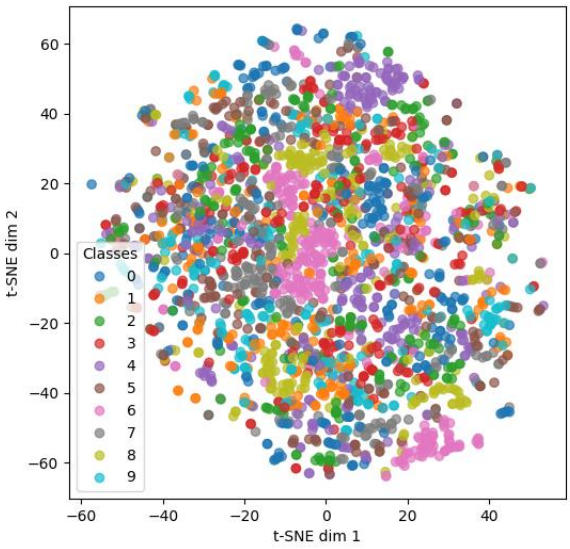

Figure 4.5: Encoded SC09 latent space visualisation via t-SNE for β=0.3.

We conducted similar experiments on the ESC50 dataset, with worse overall performance likely due to dataset complexity and fewer samples for generalisation but experiments still followed similar behaviour with reconstruction loss converging until β is increased, at which point the latent space is regularised. Visualising the latent space with t-SNE in figure 4.6, the 50 classes of ESC50 are partially clustered, but still with far too much variation and overlap within the overall distribution for accurate classification.

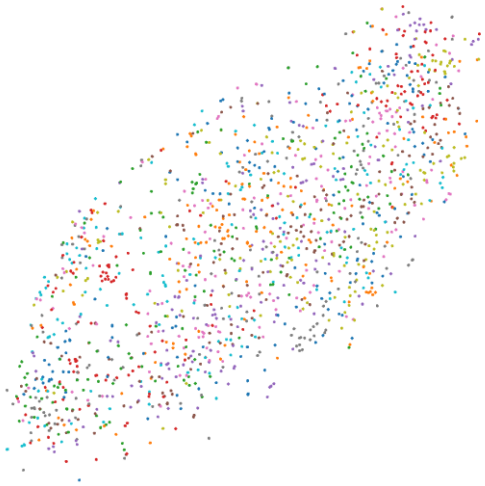

Figure 4.6: Encoded ESC50 latent space visualisation using t-SNE.

4.5.3 Supervised Classification

To determine whether the model has the capacity to differentiate between the classes, we add a classification term to the ELBO loss of the model. Specifically, we minimise the cross entropy between the logits, obtained through a linear classification layer in our model, and the ground truth labels for the input mel-spectrograms. We experiment with input to the classification head of both the latent encoded means and the latent vector produced through the sampling using the mean and standard deviation, with similar results for each. We weight the classification loss with a factor γ, to control the impact this has on overall training, giving us a new ELBO loss function:

$$\mathcal{L}_{ELBO} = MSE(x, \hat{x}) + \beta \times D_{KL}\big(q(z|x) \parallel p(z)\big) + \gamma \times Class(logits, y)$$

Training on SC09, we expected to see some improvement in clustering using this new supervised form of training, as the classification loss should encourage meaningful and distinct class means in the latent space. Early in training, even with γ values of 0.1, minimisation of class loss dominates training, leading to an extremely well clustered but fragmented latent space, as shown in Figure 4.7. As KL pressure increases, eventually the latent space is regularised to more closely match a normal distribution, at the cost of the clustering achieved earlier in training.

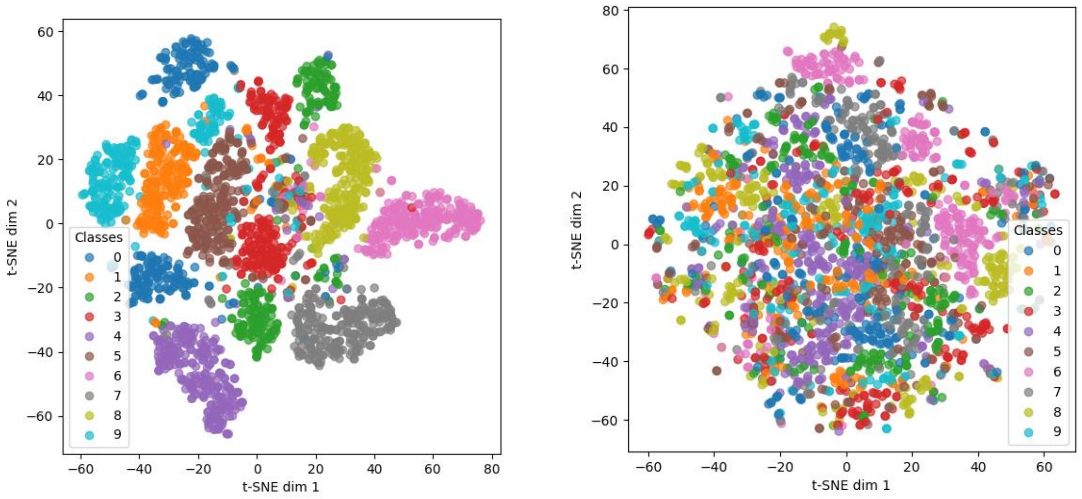

Figure 4.7: Latent space visualisations early in training (left) where class loss dominates vs later when KL loss is minimised (right).

We further test the classification ability of the encoder in our VAE by compressing the latent dimension much further to 10, equal to the number of clusters we need to capture the classes in SC09. We also weaken the encoder by replacing our double convolution blocks with single convolutions. After only 3 epochs, in Figure 4.8 we see that the encoder shifts the clusters of each class within the distribution to be equally and uniformly spaced for reliable classification on differing semantic audio content within the speech class, sharing similar characteristics and frequency patterns. Equally,

training our encoder to classify ESC50 results in similarly shifted and distributed class means within the latent space and high classification accuracy of 93%.

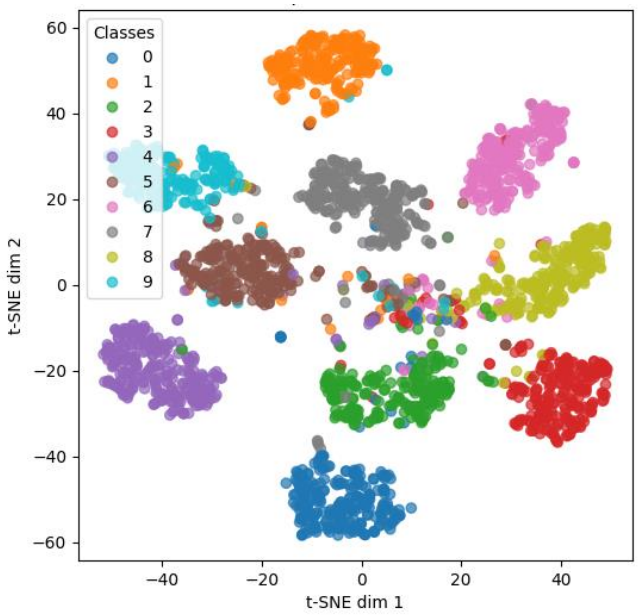

Figure 4.8: Latent space with latent dimension of 10 using class loss.

Based on this, if given a mixture of gaussian distributions containing clusters centred at different means representing classes, using these cluster ids as labels we could accurately classify new samples. Our clusters were generated using ground truth class labels from real log mel-spectrogram audio, but if these clusters can be learned implicitly using unlabelled data, this could facilitate the automatic detection and labelling of new samples, potentially extending to multi-labelling for audio mixtures.

4.5.4 Class Prediction

Using our best trained MelSpec VAE model using a latent dimension of 256 on the SC09 dataset, we compute the accuracy scores for predicting the class labels of audio samples by computing the KL-divergence between unknown encoded sample distributions and the class distributions using a small batch of labelled audio data for semi-supervised classification. We measure performance by taking the class label with the lowest KL-divergence value per class as a prediction for top-1, and the 3 lowest values for top-3, marking correct if the true label belongs to any of these classes. We vary the number of samples per class used to compute these distributions to observe the impact of increasing the amount of labelled data in our semi-supervised process, as a measure of how practically useful this method is based on the scarcity of well labelled data.

Table 4.2: Semi-supervised class-based KL-divergence classification accuracy with top-1 and top-3 predictions.

Number of Labelled Samples per Class	Overall Accuracy	
	Top-1	Top-3
5	0.097	0.312
10	0.228	0.450
25	0.318	0.501
50	0.201	0.691
100	**0.381**	0.622
200	0.344	0.610
500	0.299	**0.716**
1000	0.340	0.692

The results in Table 4.2 show that using a very small number of labelled samples such as 5 per class to compute our KL-similarity metric for class prediction results in essentially an equally random chance of correct prediction or roughly $\frac{1}{N_{classes}}$ for top-1 and $\frac{3}{N_{classes}}$ for top-3 accuracy. Increasing the number of samples per class shows a clear improvement in classification accuracy with this metric, achieving an accuracy of 0.381 for 100 samples per-class. Although this isn't extremely strong, the accuracy using the top-3 formulation becomes much closer to a feasible semi-supervised labelling measure, achieving almost 70% accuracy with only 50 known samples per class, computed over the whole validation set, despite only partial clustering observed in the latent space. After 50 samples, the improvements in accuracy show diminishing returns, indicating the specific samples chosen for computing the class distribution are more likely to impact accuracy rather than the quantity.

Further analysis of the per-class accuracy scores reveals the likely reason for this is due to stronger clustering of some classes but wide distribution of others. The accuracy results using 1000 labelled samples per class shown in Table 4.3 show a clear disparity in prediction accuracy between different classes. For top-1 accuracy, spoken digits 2 and 4 achieve around 90%+ accuracy, whereas our method is unable to accurately classify digits 0, 3, 6 and 8 at all, indicating they are distantly distributed in the latent space. This largely remains true for our top-3 classification scores, which in the case of digits 1, 2, 4 and 5 achieve almost perfect classification scores indicating the latent space is structured and at least partially clustering some class modes. Equally, most classes difficult to classify with top-1 scores remain difficult with top-3 label assignment predictions, digit 6 essentially being impossible to classify even with 1000 samples per class to define the class-specific distribution.

Table 4.3: Classification accuracy per class in SC09 for top-1 and top-3 predictions.

Class	Overall Accuracy	
	Top-1	Top-3
0	0.000	0.273
1	0.324	0.985
2	**0.954**	0.999
3	0.025	0.607
4	0.895	**1.000**
5	0.607	0.968
6	0.000	0.049
7	0.386	0.777
8	0.000	0.393
9	0.215	0.864

We compare this with classification accuracy on the encoded latent features using our supervised classifier head using ground truth class labels from the validation set of SC09, in addition to unsupervised clustering baselines using the KMeans algorithm (Lloyd, 1982) with 10 clusters and a gaussian mixture model (GMM) (Dempster et al., 1977) with 10 components, computing accuracy through majority mapping clusters to class labels. We also provide an upper bound of accuracy on SC09 provided by the ResNeXT network, achieving near perfect classification accuracy (Kong et al., 2021) in the fully supervised setting. As expected, our results in Table 4.4 show class prediction accuracy is almost perfect using ground truth labels, requiring minimal epochs for convergence, compared with much longer training on the VAE for structuring the latent space. However, in the absence of strongly labelled data this kind of supervised classification isn't possible, leading to our motivation in exploring our semi-supervised method. Even so, fully unsupervised clustering with KMeans achieves comparable classification accuracy to our semi-supervised top-1 KL-classification method, indicating uninformed clustering performance using our VAE currently doesn't provide huge benefit in comparison to traditional methods, which require no labelled samples.

Table 4.4: Classification accuracy using supervised classification and semi-supervised top-1 and top-3 classification.

Method	Overall Accuracy
KMeans (10 clusters)	0.242
GMM (10 components)	0.150
Top-1 KL-Classification (1000 per class)	0.340
Top-3 KL-Classification (1000 per class)	0.692
Encoder (Supervised Classification Head)	0.954
ResNeXT	**0.988**

4.6 Conclusion

Through training a VAE on log mel-spectrograms, we extensively tested the feasibility of constructing a disentangled and structured latent space for semi-supervised classification of new samples. Regardless of the weighting on the KL-divergence term or reconstruction quality, the latent space couldn't easily be disentangled without the aid of supervision or conditional information, likely due to data complexity between class samples and commonly shared frequency information between the classes. Using our semi-supervised classification scheme for prediction of class labels through the computation of KL-divergence, we achieve modest prediction accuracy with relatively small amounts of labelled data. However, this is poor in comparison to other baseline methods such as ResNeXT using simple classification where labelled data is available, and similar to the KMeans algorithm which requires no labelled samples. We also cannot be sure that clustering with our VAE would correlate exactly to semantic classes, as rather than clustering for semantic content, the model could instead cluster on the temporal location of the sound, or specific frequency ranges, which may correlate with many samples across different classes, making it sensitive to dataset specific features. For example, we may get clusters for audio contained at the beginning of audio clips, and others found towards the end, resulting in poorer classification performance. Our classification performance indicates that clustering seems to at least partially correspond to class, though whether this is due to semantic content or other biases in the data is unclear. Further work would be needed to investigate this problem and mitigate it, as currently there are no specific controls in place to prevent it, especially without classification guidance in the full unsupervised setting.

We believe it may be possible to structure the latent space more strongly for improved performance, perhaps with the introduction of semi-supervised learning using conditional embeddings or a deeper neural network, though the likelihood for this to exceed other simpler methods is unlikely. However, we've learned that our encoder can easily classify samples given class labels both on semantic content, such as the words spoken by a wide range of speakers, and more diverse content with high intra-class variability such as environmental sounds, provided the distribution of the latent space is

well clustered. Given this information, we explore a method to automatically capture these isolated clusters as a mix of gaussian distributions centred on equally distant means without the need for supervised training or ground truth class labels in Chapter 5.

Chapter 5

5 Infinite Audio GANs for Unsupervised Audio Clustering, Synthesis and Classification

5.1 Introduction

Although generation of synthetic data is a challenging task, deep learning models have achieved great strides in quality and efficiency in generation tasks across both images and audio. Several approaches have been taken to tackle the challenge including Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Normalizing Flows and more recently Diffusion models; each of these approaches have their respective strengths and weaknesses. VAEs excel at producing diverse samples of data, though this can often come at the cost of quality, leading to 'blurry' output (Bredell et al., 2023). Normalizing flows are exceptionally stable to train (Omray, 2021), though due to its bijective nature results in a very high dimensional latent space, lack of expressiveness and inflexible network architecture relative to other models (Li et al., 2025). Diffusion models are currently very popular, largely due to their high sample quality and diversity (Li et al., 2025), though often this requires large datasets and is computationally expensive and time consuming (Ulhaq and Akhtar, 2024). GANs are also popular, though much more prior to the mass adoption of diffusion models. This is because they are relatively quick to train with low computational costs, though like other approaches they have their disadvantages. The training process can often be very unstable, leading to serious issues such as mode collapse, where only a small number of samples or classes are captured, and all generated samples are very similar or in the worst case identical. Unlike the disadvantages posed by the other methods of synthesis, mode collapse can potentially be leveraged to improve sample quality and diversity, which at first glance seems to be counterintuitive.

GANs consist of a generator network and a discriminator network, where the generator attempts to generate samples indistinguishable from training data, and the discriminator attempts to detect whether the data it is provided is real or fake. This results in a training process where the two models compete, where the generator attempts to deceive the discriminator. This results in the generator increasing sample quality, which in turn forces the discriminator to be more critical of the quality of the sample, further motivating the generator to improve quality further. The problem with this process is that to improve at generating samples to fool the discriminator, the generator often collapses to focus on generating a small number of samples with very similar features that can reliably pass for

real data. When working with a multi-modal dataset this often means only one or a small number of classes are captured during training. In the case of audio data, an example could be only generating samples that sound like drums when trained from a dataset also containing bass, vocals, guitar etc. The purpose of many generative tasks is to capture multiple modes at high quality, meaning mitigating this weakness has been intensively investigated to make GANs more reliable.

Rather than seeking to eliminate the problem of mode collapse, in (Ying et al., 2021) the authors sought to harness it in an unsupervised framework, incorporating conditional embeddings not from prior knowledge such as a ground truth class label, but that could be learnt through training and applying a Chinese-Restaurant Process (CRP) (Aldous, 1985) to sort clusters of data into their respective modes whilst encouraging an increase in generative quality in successive training loops. Our work adapts this approach into the domain of audio synthesis, to investigate the potential for this framework to automatically learn the latent modes and map these to individual GANs, for audio synthesis and class discovery.

5.2 Related Work

5.2.1.1 Mode Collapse

Mode collapse is a common weakness (Che et al., 2017; Goodfellow et al., 2014; Jiralerspong and Gidel, 2022) in generative adversarial networks, which leads to the generator refusing to capture the overall distribution of the original dataset, focusing only a limited number of modes or samples. There are several theories as to why this occurs, including flaws in the learning procedure itself rather than seeking to minimize divergence, as shown in an experiment where minimizing the KL-divergence from the original data distribution was a training objective, and mode collapse still occurred (Goodfellow et al., 2014). Another theory is that the minmax formulation $min_G\ max_D\ V\ (D, G)$ is not necessarily favoured over maxmin $max_D min_G\ V\ (D, G)$, leading to the generator attempting to map most latent vector values to the most likely values to be identified as real by the discriminator, resulting in a low number of modes captured (Kossale et al., 2022b). Further research indicates a low-capacity discriminator could be at fault, failing to recognize the generator has only captured a low number of modes with a lack of diversity across generated samples and therefore not penalising it with predictions the content is fake (Arora et al., 2017). Another study found the use of non-saturating GAN loss may be a key cause of mode collapse, as this minimises mode-seeking divergence, the opposite to KL-divergence which is designed to ensure the latent space is well-structured and disentangled (Arjovsky and Bottou, 2017).

The introduction of the Wasserstein loss function is a common approach to avoiding this mode seeking behaviour (Arjovsky et al., 2017b), the networks which use it often referred to as WGANs. This differs from the traditional GAN loss function, which isn't implicitly affected by increases in synthesis quality and simply measures how effectively the generator can fool the discriminator,

leading to mode collapse. In contrast to GAN loss, WGAN loss doesn't output a confidence value between 0 and 1 for whether a sample is fake or real, but a real number between negative and positive infinity. This informs the generator of how much the distribution of fake samples needs to be 'moved' to match the real distribution, providing more stable propagation of gradients and stable learning.

Minibatch discrimination instructs the discriminator to stop treating synthesised samples as independent and actively compare their similarity before making a prediction on their authenticity (Salimans et al., 2016). This is facilitated by computing the distance between feature maps of each sample to calculate a score measuring their diversity across the batch via L1 distance between each sample's feature matrix, defined by the equation $c_b(x_i, x_j) = exp(-||M_{i,b}, - M_{j,b}||L_1) \in R$, where x_i is the current sample and x_j is the sample it is being compared to. These comparisons are summed across all comparisons per sample of the form $o(xi)b = \sum_{j-1}^{n} c_b(x_i, x_j) \in R$. This is repeated for all samples, the result of which is concatenated with the learnt features before samples are classified as real or fake. The generator is penalised for generating samples that score too highly in similarity, encouraging them to provide more varied samples and avoiding collapse. Though this is effective, it introduces additional computational costs and can hamper generation quality for the sake of diversity.

Other models address mode collapse using multi-generator architectures alongside a single discriminator. The MAD-GAN (Ghosh et al., 2018) architecture consists of a fixed number of generators to capture multiple modes enforced by the discriminator, which not only identifies whether the sample is fake, but also must assign the sample to the GAN that generated it or the original data distribution using softmax. Increasing the number of generators showed an increase in the ability to capture multiple modes avoiding total mode collapse, with overlapping in generation occurring when this number exceeded the original modes of the dataset. MGAN (Hoang et al., 2018) is similar, though they use a third classifier network that trains in conjunction with the original adversarial process to classify samples to the index of the generator they were created by. A key limitation of these networks is the assumption of prior information on the number of modes in the data, as the number of generators is static and cannot change throughout training. They also require instantiating multiple generator networks with independent weights, which can be costly and persist through training regardless of whether they are unnecessary for capturing all modes.

The introduction of additional prior information to inform structuring the latent space, such as conditions, is also often used to work around mode collapse. CGAN (Mirza and Osindero, 2014) introduced conditioning on ground truth class labels encoded as one-hot vectors combined with the noise vector in the generator and the MLP in the discriminator, which forces the GAN to produce samples from each class. This can lead to substantial increases in the diversity of samples and their quality by organising the latent space but heavily relies on strong and accurate prior knowledge of the dataset leading to a more supervised approach. Unsupervised approaches have also seen success,

such as InfoGAN (Chen et al., 2016) which incorporated mutual information discovery between conditional latent codes and the generated data to discover modes automatically. However, this also requires prior information through specifying a static number of latent codes to capture these modes. GANs have also applied more high-dimensional conditioning using images or descriptions, though this can lead the generator to ignore the noise and focus solely on the condition, causing a lack of intra-class diversity (Mao et al., 2019).

5.2.1.2 MICGANs and CRP

Rather than mitigate mode collapse, we seek to harness it to automatically capture the modes in the data without strong prior information such as class labels. Multi-generator architectures tackle a similar task, however they require prior knowledge of the number of classes in the dataset to effectively cover all modes, without instantiating a huge number of unnecessary GANs, causing mode overlapping and excessive computational cost. Ideally, the GAN should automatically learn the number of classes and how many generators are needed to capture, exploiting their tendency to collapse to a mode of similar samples. The MIC-GANs (Ying et al., 2021) model accomplished this through a mixture of infinite conditional GANs, which in reality is a single generator single discriminator network conditioned on a latent embedding vector similar to a conditional GAN. Unlike CGAN however, the discriminator doesn't receive this condition in the first stage of training, where the only goal is to produce realistic samples. The conditional embedding facilitates multiple GANs, with each index or cluster label representing a separate conditional generator network. The conditional input is sampled uniformly from a distribution of indexes, each related to a specific GAN. Their experiments show throughout this initial stage of training, each generator conditioned on this projected embedding of the cluster label would naturally collapse to producing either one or a small number of modes, effectively capturing the number of classes within the dataset. The number of cluster labels K must initially be estimated with a number, like multi-generators. This should be a number likely to exceed the modes in the real data distribution, but unlike other multi-generator GANs the number of generators is refined in the second stage of training.

MIC-GANs introduces their implementation of an Adversarial Chinese Restaurant Process (ACRP) (Ying et al., 2021), an algorithm which performs several functions. The first is training a classifier network, which maximises the log-likelihoods of the encoded generated samples with respect to a Gaussian with mean μ_c where c is the condition as part of an infinite Gaussian mixture model (IGMM) containing K distributions, each related to a conditioned generator. The dimensionality of the latent space for this encoded embedding is chosen to match the initial number of conditions, to reflect the similarity between each distribution. This initialised encoder is then used to compute log-likelihoods that real samples are likely to be generated by each individual GAN to perform density estimation across all clusters in the IGMM, and the most likely cluster per sample is computed. These assigned modes are then compared to the number of samples already assigned to each mode, where

modes with more samples assigned are more likely to have new samples assigned to them, with a rich-get-richer sampling. This is likened to the seating in a Chinese restaurant with infinite tables, where the probability a patron will sit at a new table or an occupied one is proportional to the number of people at each table, hence the name Chinese Restaurant Process or CRP (Aldous, 1985). If a particular cluster has too few samples assigned to it through the CRP sampling, the GAN it represents is effectively destroyed, reducing the number of generative networks in the next round of training. The remaining generators are fine-tuned through more training like the initialisation stage, but now conditional embeddings are sampled with a probability proportional to the likelihood of sampling each cluster in the categorical distribution after CRP. Successive iterations of ACRP cause the number of generators to converge to the number of modes in the dataset, whilst simultaneously refining the quality of synthesised samples for these modes, thus performing automatic unsupervised class discovery and synthesis. Inspired by the success of ACRP in the image domain, our work investigates how well this process translates to the audio domain using a similar process.

5.3 Infinite Audio GANs Framework

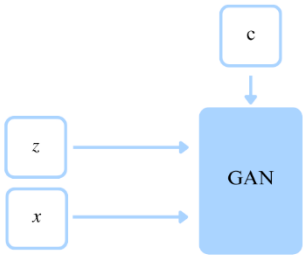

Figure 5.1: Stage 1 of the Infinite Audio GANs Framework consists of standard GAN training, with an embedding projected from cluster label c additively combined with the sampled noise vector z for use in the generator, and real audio data x used in training the discriminator.

The infinite audio GANs framework consists of two main stages: initial GAN training shown in Figure 5.1 and CRP refinement training shown in Figure 5.2, the latter of which follows the ACRP algorithm. In our framework, a given audio GAN model is trained on log-mel spectrograms (or HuBERT audio features in some ASGAN configurations) alongside a conditional embedding. The embedding is additively combined with the sampled latent noise vector, to facilitate a theoretically infinite number of conditional GANs, where each embedding corresponds to one GAN. The number of conditions or clusters is initialised to some value K, that is likely to be greater than the number of distinct modes in the data. For example, in a dataset that is likely to contain at most 10 classes, we may select a value of $K = 15$. A value between 0 and $K - 1$ is chosen randomly as a pseudo class label and used to create an embedding projected to a tensor of equal dimensionality to the random noise (e.g. 512). During initialisation, due to the random sampling of the conditions, each class label will attempt to generate samples from each distinct mode initially. Adversarial training is conducted without specific checks in place to prevent mode collapse, which should in theory result in the samples generated from each of these clusters to collapse to similar samples from the same mode. On the MNIST dataset, after successive epochs of training, each GAN generates samples from either one or few modes at reasonable quality. However, this automatic clustering behaviour can be further refined through ACRP, whilst simultaneously improving sample quality through further adversarial training in stage 2.

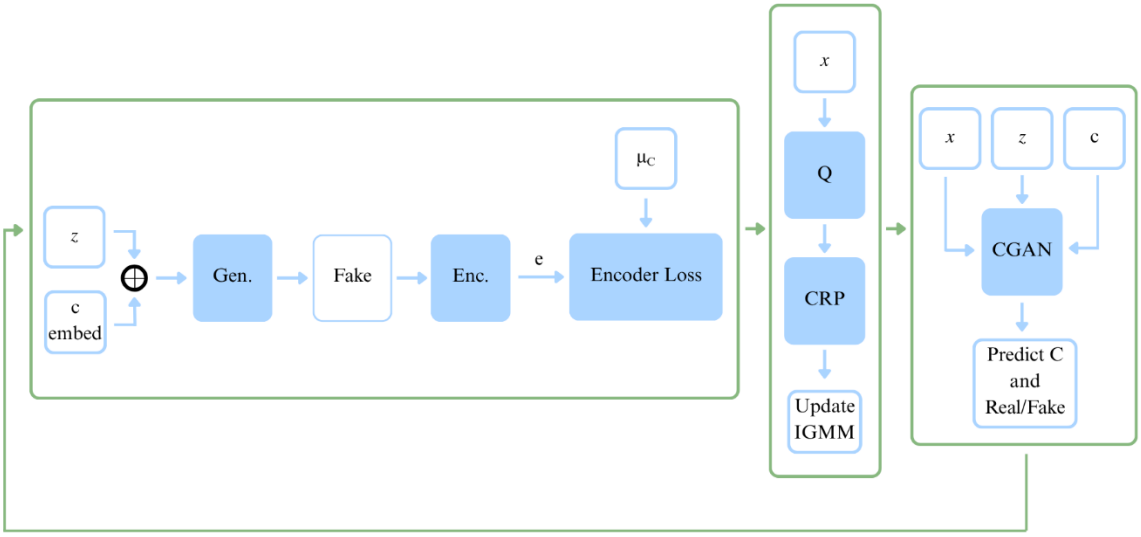

Figure 5.2: Stage 2 of the Infinite Audio GANs Framework begins with encoder training (left) on synthetic audio data using the GAN generators from stage 1 to compute cluster likelihoods with the IGMM, which should each now correspond to one or few modes mapping to classes in the dataset. This trained encoder is used to extract embeddings from real audio data x to compute likelihoods with the IGMM for the CRP process to assign samples to GAN clusters and update the IGMM (middle). Lastly, GAN training resumes in a conditional context, acting as a CGAN (right), sampling class labels from the remaining clusters following CRP to further improve sample quality. This entire stage repeats according to the number of CRP epochs set, theoretically leading to convergence of modes towards the number of classes in the dataset.

This next stage first involves training an encoder on density estimation to accurately output embeddings using synthesized data to produce likelihood estimates against the embeddings produced by an infinite gaussian mixture model (IGMM) using the same pseudo class label relating to a gaussian cluster. Provided the initialisation training resulted in mode collapse leading to biased synthesis in favour of one or few modes, the encoder should converge quickly. Real data is then fed into the trained encoder to output embeddings for computing the likelihoods that each sample belongs to a given Gaussian in the IGMM, which is used in the CRP process. Each sample is initially assigned to a given cluster, but the likelihoods are combined with a rich-get-richer sampling procedure to dynamically reassign samples based on cluster popularity and probability per sample. Essentially, this leads to clusters with many assignments gaining more assignments due to high confidence. Through successive CRP refinement loops, low-confidence clusters can eventually be destroyed or removed from sampling, allowing the number of active clusters to converge towards the true number of modes. Following this, conditional GAN training begins by activating the conditional final layer in the discriminator, but in contrast to initialisation, only class labels that CRP hasn't already removed are selected, meaning once the probability of a cluster is set to 0, it is effectively removed from the training process. This entire process is then repeated for the given number of epochs. We explain each stage with respect to our implementation in more detail in section 5.4.3, but for information on the original algorithm please review the original MICGANs paper (Ying et al., 2021) where it is explained in depth. We adopt their codebase to implement the ACRP algorithm, specifically their IGMM model, CRP logic and likelihood calculations, though we implement our own training process

for the audio GANs in both stages, and implement our own encoder for computing embeddings to facilitate the likelihood calculations used in CRP. We reuse their data preparation for computing the MNIST baseline, but our own for converting, processing, synthesizing and loading our audio data.

5.4 Implementation

The task consists of two main stages, initialisation and refinement with ACRP. In the initialisation stage, the GAN is trained to produce samples of sufficient quality to ensure the ACRP process can automatically assign samples to the correct clusters. This is important as the encoder computes the embeddings needed for log-likelihoods of each sample belonging to each GAN in the IGMM used in the ACRP algorithm, and heavily distorted or otherwise indistinguishable samples would result in poor performance and heavily inaccurate likelihood results. This is also the stage at which mode collapse should start to happen, where each GAN will take conditioning in the form of random pseudo-class labels which are projected to a latent-embedding vector which is combined with the latent noise the fake samples are synthesised from. As the class information is random, the GANs will struggle to distinguish between what makes one GAN belong to a specific conditioning vector over another, eventually nurturing mode-collapse towards clusters of data for each individual condition, to cover the distribution and achieve the best possible loss values.

5.4.1 Dataset

We utilize the SC09 spoken digits dataset, which is semantically the most similar to the MNIST dataset used in the image domain, which we compare our results to as a baseline between the image and audio tasks. There are several critical differences however: SC09 contains temporal data in the form of .wav files, waveforms of continuous audio data, whereas MNIST is comprised of images of hand-written digits, constituting discrete spatial data. SC09 is also much larger, as each MNIST image consists of a 28x28x1 grayscale image, containing a total of 784 datapoints which increases to 32x32x1 or 1024 datapoints after the image transform. For comparison, each audio file is sampled at a rate of 16KHz with a duration of 1 second, meaning each sound consists of around 16000 datapoints, making each sample from the dataset 16 times larger than an MNIST sample. There are also many more samples in the MNIST dataset which consists of 70000 images, compared to the much smaller SC09 dataset containing almost 24000 audio files. Although there are significant differences there are some strong similarities: the dataset consists of 10 classes, each sample is isolated (only a single example from a single class per sample), each sample is a human perceptive representation of a digit between 0-9, and the samples are created by a large number of different individuals to create a robust general representation of the data, in the audio case from a diverse set of voices with different accents and genders.

We also explore another dataset to compare performance and utility of the framework with more complicated and varied data. ESC50 is a dataset normally used for testing environmental audio classification or source separation systems with a large amount of variance between and within each class. This is a very small dataset containing only 2000 audio files, but this allows us to test the reliability of the clustering on classes which may have distinct differences not just from other classes but between their own samples, unlike those found in SC09. ESC50 is recorded with a 44.1KHz sampling rate, meaning the waveforms need to be downsampled to 16Khz to fit the current network architecture in line with SC09. This results in a noticeable decrease in quality, through the loss of some fine-detail contained in the higher frequency components of the audio, due to no longer satisfying the Nyquist-Shannon sampling theorem, but for our purposes the quality is still more than sufficient for distinguishability between different audio classes and individual samples.

Due to the complexity and size of the data, some preprocessing is necessary. As is often standard practice in the literature to reduce dimensionality and achieve greater convergence in training, the audio data is first transformed from a one-dimensional waveform to a two-dimensional spectrogram representation using the short-time Fourier transform. This is then mapped to the mel-scale for a logarithmic representation that better coincides with the sensitivity of human hearing. This visual format of an audio signal can be used to train the model similarly to an image-based GAN, allowing us to utilize a lower dimensional 2D deep learning architecture and visual inspection of the output during training. The dimensions of the mel-spectrogram depend on the options chosen during preprocessing, but to preserve the quality and intelligibility of the signals, we chose to use group the frequency information into 128 mel bands contained in the y-axis of the data, and parameters and padding/cropping that leads to 128 time windows or frames of audio contained in the x-axis, which is achieved using a hop-length of 512 and n_fft of 2048. The resultant mel-spectrogram is scaled from power to dB, which we found made training much more stable.

Due to the much smaller size of the ESC50 dataset, we artificially boost the number of samples per class using audio augmentation techniques on the original samples and save these for training. This includes pitch shifting up or down by up to 2 semi-tones and time-stretching between 0.8 to 1.2 times the duration of the original signal using the Librosa library, which is cropped or padded to the correct size during the audio transformation to a mel-spectrogram image. We find that although the audio may be obviously perceptibly different from the original source waveform, the class is still just as clear to a human listener and the quality of the signal isn’t significantly degraded. We ignore other augmentations that either partially destroy or deliberately degrade the signal to both prevent the model from learning to synthesize poor quality samples and make analysis of generated audio quality clearer and more objective.

5.4.2 Model Architecture

For the generative architecture of the overall network, we adopt models based on the foundational frameworks of DCGAN and StyleGAN, with modified versions of SpecGAN (Donahue et al., 2019a) and ASGAN (Baas and Kamper, 2022). We chose these models due to their ease of implementation with mel-spectrogram data and their differences in generation quality and training speed. This would allow us to analyse the trade-off between generated audio quality and model complexity when incorporated into the infinite GANs framework to observe the effects and inform which models can benefit from it the most. We explore each model in more depth and compare their benefits and drawbacks in the Appendix. To implement the training of an infinite number of GANs, we must also introduce a conditional variable in the form of an embedding. This allows us to train as many GANs as we have different conditions without having to instantiate entirely new GAN models for each cluster. The original StyleGAN already has an implementation for conditional embeddings in the form of ground-truth class labels, but we will modify all the networks to use a specific and consistent conditional embedding to fine-tune the weights for each GAN depending on their cluster number which acts as a pseudo-class label.

This conditional information is incorporated into the latent noise z, used as input for the generator. We explored both multiplicative and additive conditioning on incorporating the latent embedding into the noise vector in the original MICGANs experiments with MNIST, and found that the automatic clustering of modes during the initialisation stage of training only occurred with additive conditioning. This results in $z_e = z + e_c$, where z is drawn from a normal distribution $z \sim N(0, 1)$ and e is the projected embedding of GAN labels sampled from the indexed list of all GANs proportional to the probability of sampling each GAN, such that $c \sim Clusters(\alpha_1, \ldots, \alpha_K)$ where α is the probability. During initialisation, the probability of sampling any one label, and thereby the GAN itself during generation, is uniform across all GANs where $\alpha_n = \frac{1}{K}$ for K total conditions. During the CRP stage, these probabilities are refined, where the probability of sampling GANs that capture whole modes is increased and those that don't are gradually reduced towards 0, effectively destroying the GAN. Note that the model isn't a conditional GAN as the discriminator doesn't receive any conditional information during the initialisation stage of training, only the generator to encourage mode collapse and facilitate the infinite GANs formulation.

5.4.3 Training Procedure

We train SpecGAN model with a learning rate of 0.0001 using the Adam optimiser with betas [0.5, 0.9]. We train ASGAN with a learning rate of 0.002 also with the Adam optimiser with betas [0.0, 0.99]. We train both models with a batch size of 16, though with experiments in ESC50 with we increase the batch size to 64 for a better chance at synthesizing samples from more classes in each

batch. The size of the sampled noise vector z is 256 for SpecGAN and 512 for ASGAN, the dimensionality of which remains the same after incorporating the additive conditional cluster embedding. Libraries used include PyTorch for facilitating the neural networks and training, Numpy for numerical operations on data and results and Librosa for converting data between waveforms and mel-spectrograms, performing augmentation and visualisation of spectra through MatPlotLib. A typical training run ranges from 12 hours to 3 days depending on the depth and complexity of the architecture, number of epochs/iterations in initialisation and CRP training stages, dataset size and dimensionality.

Originally the creators of SpecGAN used Wasserstein loss to train generator and discriminator. In addition to providing more stable training and meaningful gradients even when distributions are far apart, a key benefit for this loss function is helping to prevent mode collapse, as in order to minimize the distance between the generated and real distributions effectively, the generator must learn to cover all the modes rather than focusing on producing high quality samples with very little diversity or mode coverage. However, we seek to exploit the problem of mode collapse, meaning that in our use case we will forego the use of Wasserstein GAN loss and use non-saturating GAN loss for both models, a slight edit on traditional min-max GAN loss whereby instead of the generator attempting to minimise the likelihood of its output being predicted as fake, it maximises the likelihood of is output being predicted as real, preventing model saturation. This modification helps to prevent the generator from being overwhelmed during early stages of training by the discriminator, allowing it to learn enough that fooling the discriminator is possible. Unlike loss functions such as Wasserstein GAN loss, non-saturating GAN loss does not help prevent mode-collapse, making it suitable for our purposes.

The task consists of two main stages, initialisation and refinement with ACRP. In the initialisation stage, the GAN is trained to produce samples of sufficient quality to ensure the ACRP process can automatically assign samples to the correct clusters. This is important as the encoder computes the embeddings needed for log-likelihoods of each sample belonging to each GAN in the IGMM used in the ACRP algorithm, and heavily distorted or otherwise indistinguishable samples would result in poor performance and heavily inaccurate likelihood results. This is also the stage at which mode collapse should start to occur, where each generator is conditioned on a randomly assigned cluster ID embedding. As the class information is random, the generators learn to synthesise similar samples to satisfy generating diverse samples between each condition, nurturing mode-collapse towards covering different classes in the data distribution.

5.4.3.1 Initialisation Stage

In each epoch, we iteratively sample the prior distribution $p(\phi|\Phi)$ with samples from dataset $x{\sim}X$ and a normal distribution for the noise vector $z{\sim}N(0,1)$. We also randomly sample a cluster label from a categorical distribution $c{\sim}Categorical(\alpha_1, \alpha_2, \ldots, \alpha_K)$, where $\alpha = \frac{1}{K}$ and K is the initial

estimate of clusters, leading to a uniform sampling across all generators. This is projected to an embedding matching the dimensionality of noise z. The generator G takes the noise and conditional embedding as input, which are additively combined to generate fake samples $x' = G(z, c)$, maximising the probability discriminator D classifies the samples as real, of the form $\max D(G(z, c))$. D is trained to maximise the probability samples x are classified real and generated samples x' are fake, of the form $\max D(x) + (1 - D(G(z, c))$. These functions are optimised to adversarially train the GAN until samples of sufficient quality are produced. This should also cause significant mode collapse, as there are no specific checks in place to counter it and conditions imposed on the generator to encourage it.

5.4.3.2 ACRP Stage

Following GAN initialisation is the ACRP stage of training. This consists of 3 key steps: training an encoder for log-likelihood calculation, performing CRP sampling to update the IGMM for cluster sampling probabilities, and further fine-tuning of the GAN using these new weights for the conditional embeddings.

As GANs cannot compute log-likelihoods directly, we incorporate a surrogate classifier network to facilitate this for density estimation. The architecture of the classifier encoder is a simple stack of 5 Conv2D layers with LeakyReLU activations, leading into a network head that outputs a latent embedding of the input. The pre-trained GANs are sampled uniformly in the initial training run of the encoder as the probability each generator is sampled during initialisation is equal. To train the encoder network, we iteratively sample $x' = G(z, c)$ which the encoder uses to output an embedding vector $e = E(x')$ of size K. The log-likelihood of this embedding with respect to the Gaussian in the IGMM, containing K Gaussian distributions, with mean μ_c representing a cluster, is maximised to train the encoder using MSE loss. This teaches the encoder which generator a sample is most likely to be drawn from. Initially the means of these Gaussians in the IGMM are equidistant to ensure they are distinct, but over the course of ACRP these means are adjusted.

First, latent embeddings e_i are output by the trained encoder network for batch of real mel-spectrograms x_i. The likelihood $l_{i,k}$ these embeddings are drawn from a gaussian with mean μ_k and covariance Σ_k for each cluster k in the IGMM is then calculated such that $l_{i,k} = Gauss(e_i|\mu_k, \Sigma_k)$ for all K. For each sample $x \sim X$, the cluster with the highest likelihood $c_i = argmax(\{l_{i,k}\}_{k=1}^{K})$ is computed. This sample is intermediately assigned to this cluster and the total number of assignments to that cluster N_{c_i} is incremented. The likelihoods per cluster $l_{i,k}$ are then multiplied with the number of assignments per cluster N_{c_i} to determine the weight towards assignment of the sample in each GAN β_k where the sum of all weights is equal to 1. With these weights, the final cluster assignment $c_i \sim Categorical(\beta_1, \beta_2, \ldots \beta_K)$ is sampled and their total assignments updated accordingly.

Embeddings e_i and cluster assignment c_i are used to update the cluster means μ_k in the IGMM and remove any GANs with a low number of assignments for convergence towards the true number of modes.

Following this step, the GANs are trained further to fine-tune their generation using the new categorical distribution of clusters labels. Their weighting α is no longer $\alpha = \frac{1}{K}$ after the first CRP step, as this sampling is now tied to the weight of each cluster determined by the number of samples assigned to it previously. Training is similar to the initialisation stage, with the caveat that the discriminator is now trained to predict whether the audio is real or fake and which cluster it was generated from, meaning conditional GAN loss in the discriminator of the form $\max D(x, c) + (1 - D(x', c))$ is now optimised. This is repeated until convergence of the number of GANs matches the modes in the dataset and high-quality synthesis is achieved.

5.5 Results and Discussion

To measure unsupervised clustering performance, we evaluate how well the model captured the modes using the purity score (Manning et al., 2009), formulated as

$$Purity = \frac{1}{N}\sum_{i=1}^{K} max_j |c_i \cap t_j|$$

for all mel-spectrograms N and K clusters, computed from the maximum overlapping class t_j for each cluster c_i. The purity of each cluster for their class with the most overlap is used to compute the total purity score across all clusters, where a score of 1 represents a perfect mapping of each individual class to an individual GAN and 0 is the total inability to perform this clustering successfully. For each dataset we choose a different estimate of the number of clusters K needed.

5.5.1 SC09

Table 5.1: Purity scores for experiments with MNIST in the image domain using DCGAN, and SC09 in the audio domain using SpecGAN and ASGAN, where K=15.

Model	Dataset	Purity
DCGAN	MNIST	**0.8973**
SpecGAN	SC09	0.1016
ASGAN	SC09	0.1023

Initially we experimented with the entire SC09 dataset of 10 classes using $K = 15$ trained with SpecGAN and ASGAN for a direct comparison with our baseline experiments on MNIST, which is semantically similar to SC09 in the image-domain, using the DCGAN to replicate and verify the results obtained in (Ying et al., 2021). As the results in Table 5.1 show, both the SpecGAN and

ASGAN models are unable to capture these clusters automatically, resulting in very low purity scores. In contrast, the results using DCGAN on MNIST for clustering purity are very high, showing the majority of samples from each written digit class mode are captured within individual GANs.

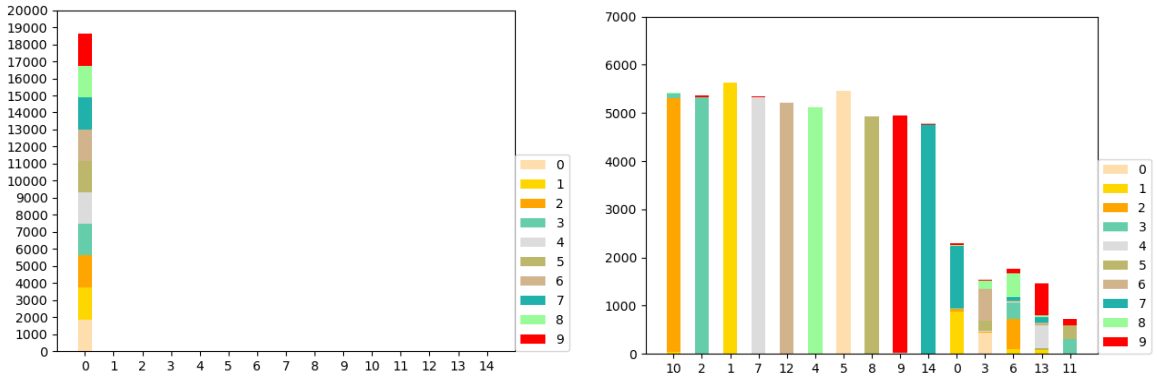

Figure 5.3: SC09 clustering results (left) and MNIST clustering results (right) after 10 ACRP epochs..

As we can see in Figure 5.3, the CRP process has filtered out all but one of the clusters for sampling the SC09 dataset (the same results for both SpecGAN and ASGAN), resulting in a single GAN that attempts to synthesize the entire dataset. Expected behaviour should be similar to the chart for our experiments on MNIST, where 10 of the clusters synthesise almost entirely only one mode from the original dataset, whilst the rest containing overlapping modes are sampled much less. To explore the reasons for this behaviour, we first analyse the quality of the synthesised speech more deeply to determine whether it is too unintelligible or similar for the classifier to distinguish between each class, using the ASGAN model. We measure the modified inception score (mIS) to assess both quality and intra-class diversity of generated samples (Gurumurthy et al., 2017), activation maximization (AM) to measure audio quality by comparing the KL-Divergence between real and fake data distributions (Zhou et al., 2018), and both Fréchet inception distance (FID) and Fréchet audio distance (FAD) (Heusel et al., 2018) to measure the perceptual quality and similarity between distributions of real and generated audio samples (Heusel et al., 2018; Kilgour et al., 2019). Our best results from training ASGAN using mel-spectrogram or HuBERT audio features unconditionally, conditioned on random cluster ID labels, and conditioned on ground truth class labels are shown in Table 5.2.

Table 5.2: Quantitative evaluation of audio synthesis quality and diversity across training ASGAN unconditionally, conditionally on cluster ids and conditionally on ground truth class labels, measured by FAD, AM and MIS.

ASGAN Type	FAD	AM	MIS	FID
Unconditional	15.95	0.82	197.35	0.71
Conditional (Clusters)	14.08	**0.13**	214.42	5.41
Conditional (Ground Truth)	**12.52**	0.42	**308.66**	**0.29**

Using the same hyperparameters, unconditional training achieved the worst overall scores in FAD, AM and MIS. Using pseudo-class labels as conditions for the random clustering in training, we achieved a lower FAD score indicating greater perceptual quality and higher MIS score indicating higher intra-class diversity, with the best AM score out of the three comparisons, showing low KL-divergence between the fake and real data distributions. Overall, using ground truth labels achieved the best FAD, MIS and FID scores which is to be expected, as no clustering needs to take place for the GANs to generate samples from a single mode of data, ignoring all others. Subjective assessment of the synthesised audio also clearly shows the model can produce not only audio with easily distinguishable classes (spoken digits), but of relatively high-quality in comparison to the original data across all training schemes. Interestingly, the FID scores of using ground truth label conditioning and unconditional training are relatively strong, whereas training for random clustering produces much worse scores. Subjective analysis shows this model still produced samples from all classes across different speakers, indicating this score may be indicative of some collapse, though not due to semantic content or speaker id/gender.

We perform more tests on a simplified version of SC09, containing only 3 classes: ‘zero’, ‘one’ and ‘two’. Hyperparameters are identical with a modified estimate for $K = 5$ to reflect the smaller number of classes for less redundancy. We train both SpecGAN and ASGAN on this subset of the data and compute the initial sample assignments at the beginning of the ACRP stage and once ACRP has concluded. The results for the ASGAN experiment in Figure 5.4 show GANs with id 0, 1 and 4 are significantly more likely to be sampled than ids 2 and 3. However due to the almost uniform ratios of assigned samples from each class per GAN it seems the only factor that effects this assignment is sample quality, as each GAN seems to generate all classes with equal probability, leading ACRP to eventually filter the sampling of all classes into a single cluster, with id 3. The results in the SpecGAN experiment shown in Figure 5.5 are similar, where each GAN produces each class, however the probability each GAN samples from each class is much less evenly distributed, indicating some

partial collapse towards the class modes. However, this collapse isn't significant enough for the CRP process to refine the GANs into sampling from individual modes, leading to destroying all but a single GAN. The purity results in Table 5.3 reflect these assignments, as SpecGAN's initial purity is greater than ASGAN's, reflecting this partial collapse and bias towards individual modes, but through ACRP filters into a single GAN for all modes.

Table 5.3: Initial purity at start of ACRP stage and final purity through application of ACRP procedure, for both SpecGAN and ASGAN models on SC09 subset of classes 'Zero', 'One' and 'Two'.

SC09 Digits 0-2 (Mel-Spectrograms 128x128)	SpecGAN			ASGAN		
	K	Initial Purity	Final Purity	K	Initial Purity	Final Purity
	5	0.4253	0.3362	5	0.3549	0.3312

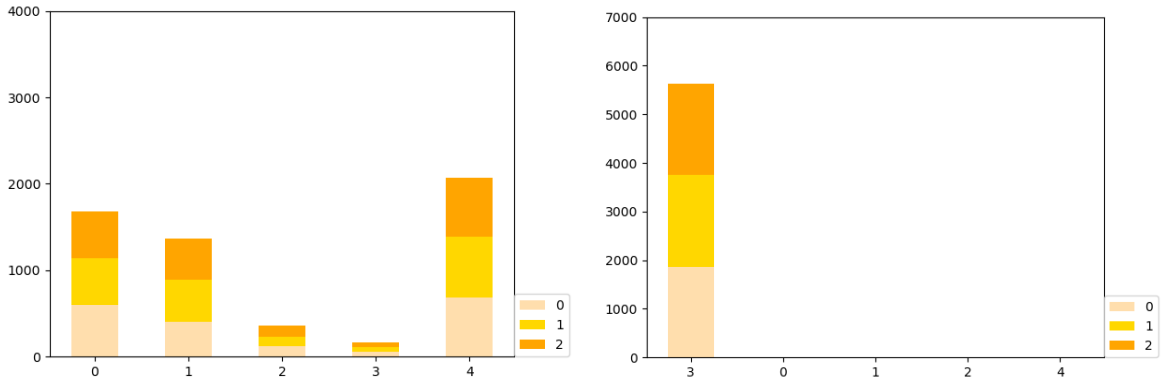

Figure 5.4: ASGAN initial clustering results (left) and final clustering after ACRP (right) for subset of SC09 digits 0-2.

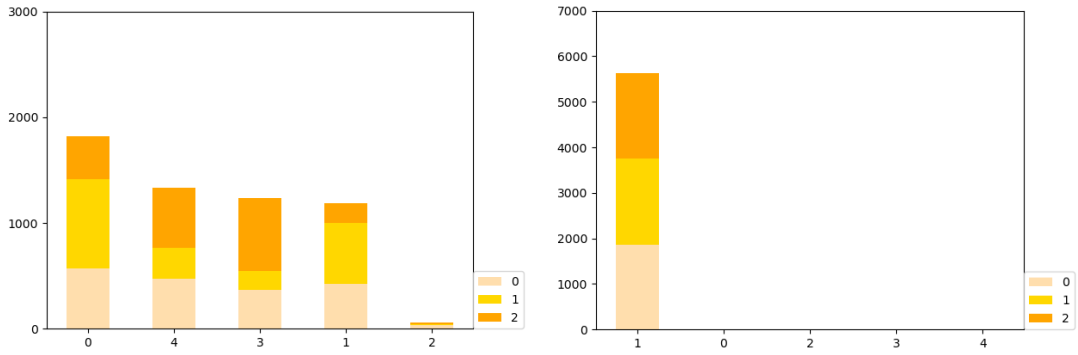

Figure 5.5: SpecGAN initial clustering results (left) and final clustering after ACRP (right) for subset of SC09 digits 0-2.

We then assess the degree to which mode-collapse occurs across the clusters, by randomly generating a large number of samples and classifying them using a ResNeXT (Xie et al., 2017) classifier pre-trained on the entire SC09 dataset, which we used for computing our audio quality metrics. We

perform this for $K = 1$ (unconditional), $K = 15$ and ASGAN conditioned on ground truth labels as a control. The number of samples generated belonging to spoken digit classes 0-9 are shown in Table 5.4. The confidence of these classifications was evaluated both via the accuracy score of the classification results and human-subjective listening tests, showing the labelling to be accurate. As expected, using ground-truth labels produces only samples from the target class, as confirmed by the classifier, giving more reliability to our other results. Unconditional training resulted in some modes being strongly captured and others being almost if not totally ignored, where over a quarter of samples belong to digit class 4 and classes 1 and 9 are barely generated if at all. We observe similar behaviour in our cluster GANs, where digit classes 2 and 9 are much more likely to be generated than all other classes, and classes 6 and 7 are rarely sampled. However, no classes are entirely ignored with clusters having different weighting for different classes, indicating our conditioning is encouraging the model to generate all classes between the clusters, but still suffering from some bias towards a few modes without totally collapsing. This could be indicative of partial collapse, or more likely that mode collapse is occurring but not between individual classes within the data but between similar sounding samples despite their semantic content (the spoken digit).

Table 5.4: Number of generated samples per class from ASGAN across unconditional, conditional (clusters) and conditional (ground truth) variants.

ASGAN Type	Total Samples Per Class (1000 total)									
	0	1	2	3	4	5	6	7	8	9
Unconditional	80	3	124	126	256	125	78	98	110	0
Conditional (Clusters)	123	47	187	90	137	80	26	20	92	198
Conditional (Ground Truth)	100	100	100	100	100	100	100	100	100	100

Through these observations and other experiments with MNIST where the conditioning was modified to be multiplicative, we found that prior to the CRP stage the training during the initialisation stage didn't result in severe collapse of any individual modes into conditional clusters. For the encoder to successively initialise and facilitate computing likelihoods for sampling from each GAN, these pre-trained GANs need to be heavily biased towards one or few modes in the data. If the initialisation stage doesn't collapse, each class is essentially equally as likely to be sampled from all GANs, leading the CRP process to eventually destroy all but one GAN. We believed the architecture of the generator itself could be a potential cause, in that it may be too powerful and robust for the mode collapse problem. This seemed relatively unlikely given the strong results for the MNIST dataset using a StyleGAN architecture, but we explored several modifications to the original design to encourage mode collapse such as reducing the depth of style blocks in the generator and removing the original mapping network taking a latent noise sampled from a normal distribution as input. We train an unconditional ASGAN with these modifications on a subset of the data containing only the digits 0, 5

and 8, based on their differences in pronunciation being more distinct than some of the other classes, ranking how often these classes are generated after sampling the generator 1000 times. We also experiment with and without injecting noise, though this only degraded the quality of the audio with no bearing on collapse, hence the results from this experiment are omitted from the table.

Table 5.5: Ranking of most dominant mode sampled from ASGAN trained with various architecture changes on subset of SC09 containing classes 'zero', 'five' and 'eight', selected for their semantic differences.

ASGAN Modification	Most Dominant Modes Sampled (out of 1000 samples)		
	1st	2nd	3rd
Shallow Gen Network Depth	'zero': 445	'eight': 278	'five': 259
Removed Mapping Network	'five': 651	'eight': 216	'zero': 125
Both	'five': 444	'zero': 350	'eight': 205

Our results in Table 5.5 indicate that reducing the depth and therefore learning capacity of the generator does not significantly affect mode collapse, with similar bias towards one class over others as seen in previous experiments with many more classes. Replacing the latent code w created using the mapping network with an unmapped latent noise vector z did appear to have a significant effect on mode collapse, resulting in the digit class 5 being sampled almost twice as often as both 0 and 8 combined. However, this mapping network is a key and common component amongst all StyleGAN architectures, meaning this framework wouldn't accept these models as they are and would require significant modifications to be appropriate. Therefore, we explore the simpler WaveGAN which uses a latent noise z by default in our other experiments on the ESC50 dataset which may be more likely to collapse due to the stark contrast in content between classes.

5.5.2 ESC50

Although semantically SC09 is very similar to MNIST as both consist of representations of the digits 0-9 created by different individuals, this doesn't mean distinguishing between these audio modes is necessarily easy. The mel-spectrogram representation of a spoken digit is much more complex than a visual representation of a written digit. Not only is it larger in size, but it is composed of many overlapping frequencies that extend across the whole sample, rather than simple lines and curves constituting a single easily recognizable shape. The model may interpret all spoken digits as a single class such as speech, or each speaker as a separate class, leading training to behave as an unconditional GAN with mode collapse not providing a trivial solution to fooling the discriminator, as many samples appear very similar across all modes. Several examples of mel-spectrogram representation of digits 0-3 for two separate male and female speakers are provided in Figure 5.6. Visually, spoken digits 0-3 from 'woman 1' and 'man 1' are similar to each other, contained in short

bursts of the audio. However, temporally whilst most utterances for ‘woman 1’ are contained in the latter half of the audio, the utterances in ‘man 1’ are spread across the whole sample. In this sense, the model could collapse to generating these similar samples between multiple speakers, the same speaker, or within the same temporal portion of the audio. In contrast, the samples from speakers ‘woman 2’ and ‘man 2’ are visually distinct from other speakers and themselves, with great variation between classes, but occur over a much greater portion of the temporal dimension. Whilst there is some similarity in shape and energy between the samples from each class from speakers ‘woman 2’ and ‘man 2’, this is not shared across most speakers, and the level of distortion between the frequency bands introduces another distinct variation within the modes. This indicates whilst audibly these speakers and their pronunciations of these digits are distinct and each spoken digit class is obvious, this may not be trivial for the generators to capture through mode collapse.

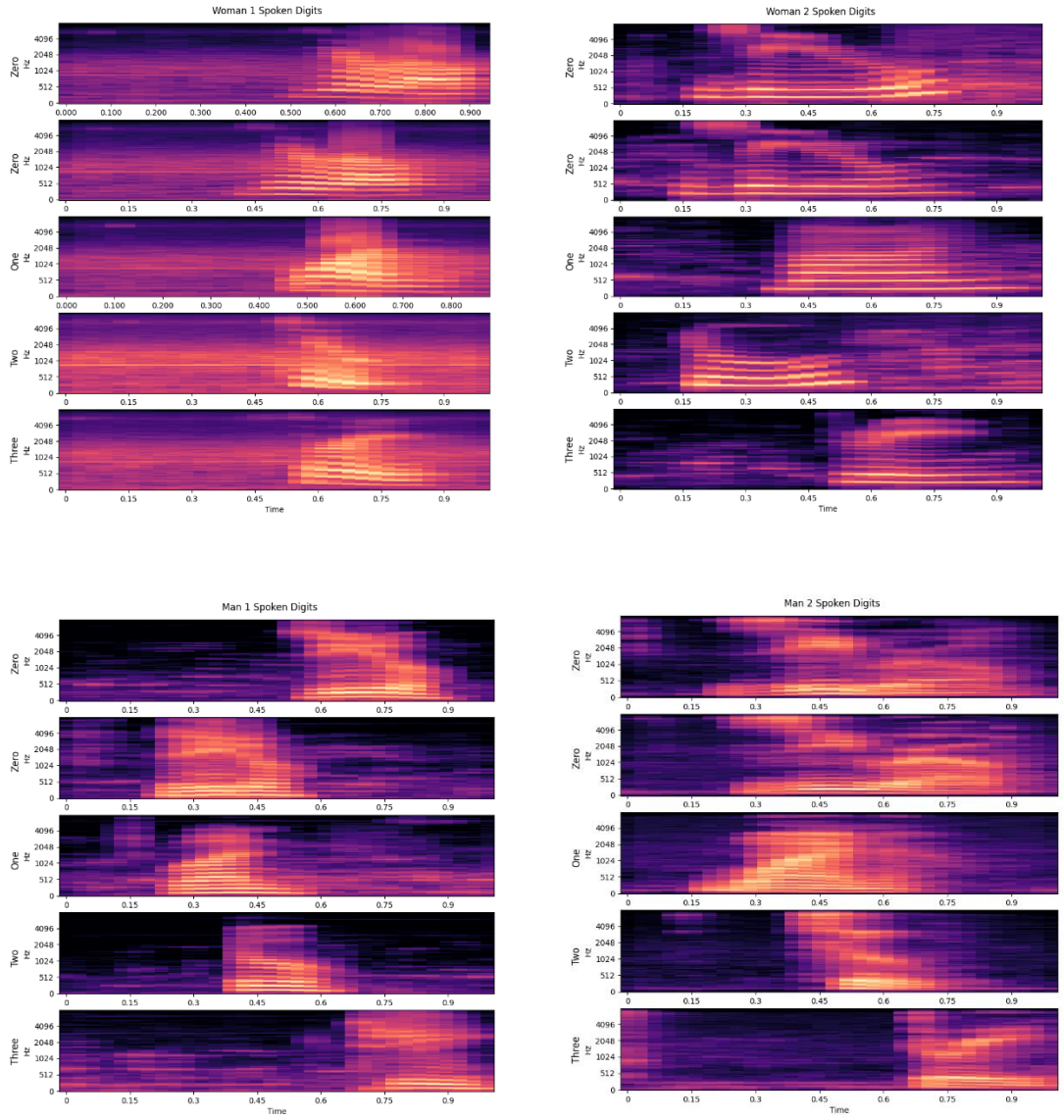

Figure 5.6: Mel-spectrogram examples of spoken digits 0-3 between 4 speakers: woman 1 (top left), woman 2 (top right), man 1 (bottom left) and man 2 (bottom right).

Therefore, we experiment with the ESC50 dataset, containing more distinct audio classes, composed of dramatically different combinations of frequencies, amplitudes and semantic content which is more visually obvious. For example, samples from class 'Sneeze' are short audio bursts occupying most of the frequency range, 'Siren' samples occupy the entire temporal axis with repeating or constant shape across frequency bands, and 'Handsaw' samples are constant repetitive bursts spanning both the temporal and frequency axes. However, there is still significant intra-class diversity, especially across certain classes such as 'Alarm Clock' and 'Laughing' which could pose difficulties for the infinite GANs mode-collapse framework. As a result of earlier experiments with StyleGAN based architectures indicating them as potentially unsuitable for this unsupervised clustering, we attempt to train the framework using the less complex SpecGAN architecture across all 50 classes, including both the Initialisation and CRP stages, with an initial cluster estimation of $K = 60$, where refinement

to $K = 50$ is the target with high purity. Although there is significant mode collapse in most of the clusters, the resulting purity is still extremely poor, indicating that although specific shapes and forms in the mel-spectrogram images are being identified as easy to replicate by the generator to fool the discriminator, the actual modes being captured don't reflect the general content of each of the classes. This is likely due to total collapse to a single sample per GAN caused by the complexity of covering so many modes with such high inter-class and intra-class diversity.

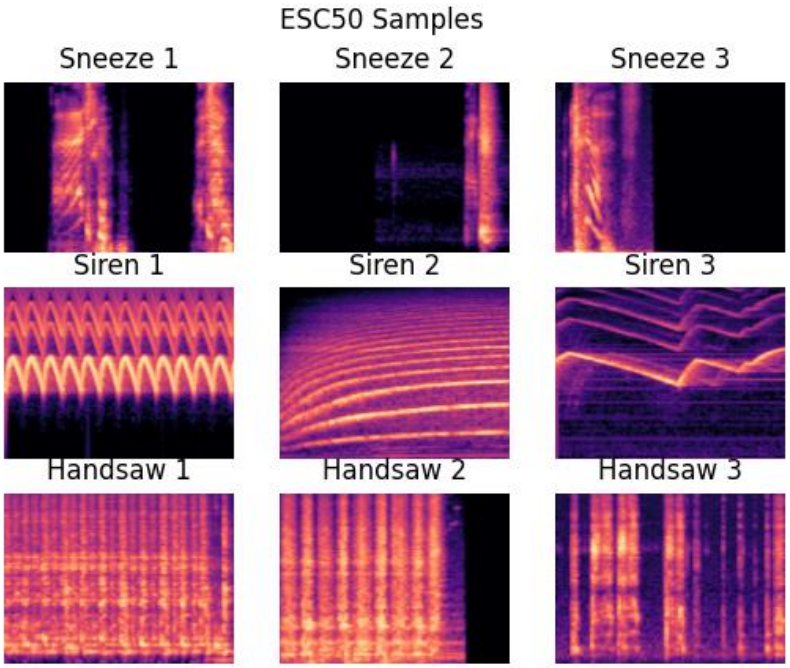

Figure 5.7: Samples from ESC50 dataset from the Sneeze (top), Siren (middle) and Handsaw (bottom) classes.

To simplify the problem further to give the model the best chance of capturing each individual mode, we focus only on 3 specific classes which are visually distinct to the human observer and easily placed in different classes: sneezing, sirens and handsaws. Examples of each of these classes are shown in Figure 5.7. We retrain with a cluster estimate of $K = 5$, which should reduce most of the sampling to 3 GANs capturing the 3 modes if successful. As shown by results in Table 5.6, initially the cluster purity is relatively high compared to previous experiments at 0.5625, though one GAN is dominant for synthesising all 3 classes. This results in the CRP process assigning more samples to this mode and thereby eliminating the others through progressive CRP epochs, leading to the lower purity score of 0.3906, more in line with our previous experiments. Figure 5.8 shows the generated samples from each cluster, where GANs with cluster id 1 and 4 produce a 'Sneeze' sample, 3 and 4 produce 'Handsaw' samples and GAN 0 produces 'Siren' samples. However, once again mode collapse appears too severe to capture the intra-class diversity, focusing only on a very specific example of the mode and failing to capture general features of the mode across all samples within the class.

Table 5.6: Initial and final purity scores on experiments on full ESC50 dataset of 50 classes, and ESC50 subset of 'Sneeze', 'Siren' and 'Handsaw' classes.

SpecGAN			
ESC50 – All classes (50 modes) (Mel-Spectrograms 128x128)	K	Initial Purity	Final Purity
	60	0.02	0.02
ESC50 – Sneeze, Siren and Handsaw (3 modes) (Mel-Spectrograms 128x128)	5	**0.5625**	0.3906

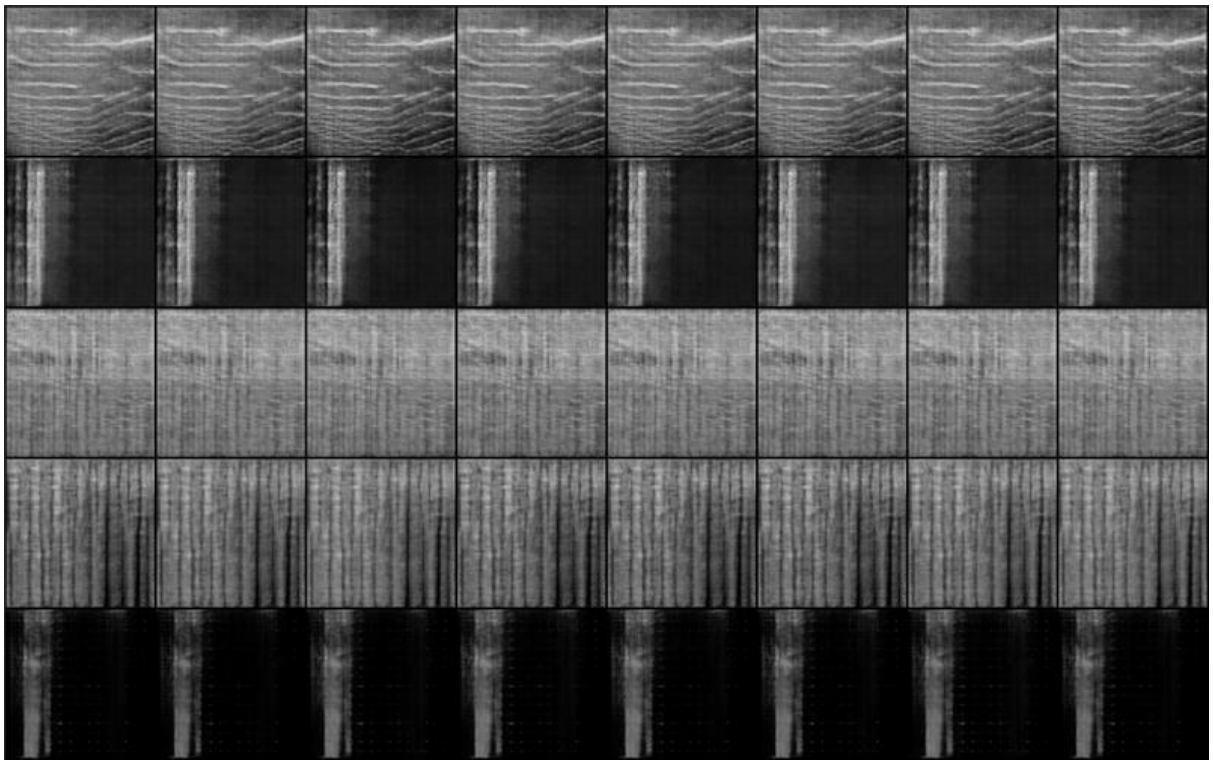

Figure 5.8: Generated samples across 5 GANs for classes 'Sneeze', 'Siren' and 'Handsaw'. From top (cluster id 0) to bottom (cluster id 4), the GANs produce very similar samples of: 'Siren', 'Sneeze', 'Handsaw', 'Handsaw' and 'Sneeze'.

5.5.3 Synthesised Dataset

To fully eliminate this intra-class and inter-class diversity problem to fully evaluate the feasibility of the ACRP framework in the audio context, we synthesise our own dataset of 10 classes. Each class contains 6000 samples, which when converted to log-mel spectrograms are visually similar to each other, whilst being extremely distinct from all other classes. These log-mel spectrograms are created with the following parameters: n_mels=32, n_fft=1024, hop_size=512. This results in audio data of shape 32x32, comparable to MNIST at 28x28, which also eliminates the size of the data potentially impacting mode collapse. Samples of each class are shown in Figure 5.9.

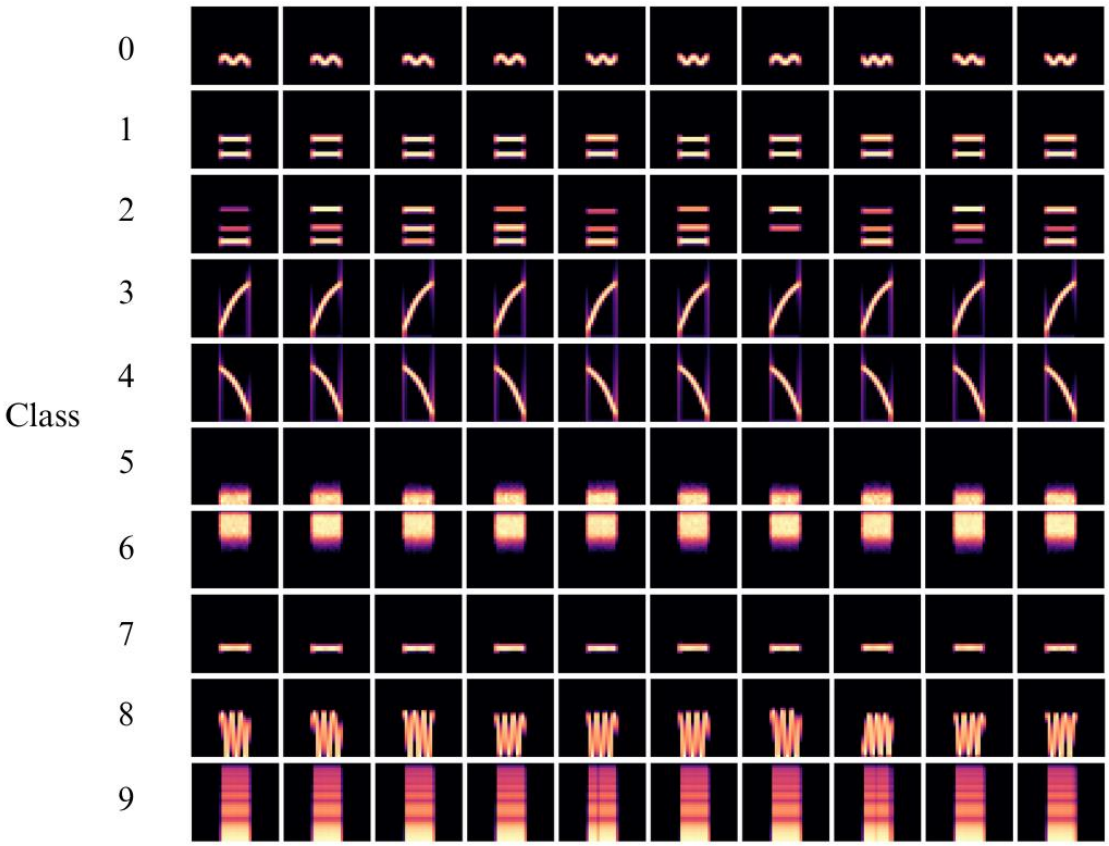

Figure 5.9: Samples of our synthetic dataset of distinct classes 0-9 (top to bottom rows).

We train our SpecGAN based model with $K = 10$ for 5 epochs in the initialisation stage, which results in strong mode collapse to specific samples, as it did in our ESC50 experiment. However, due to the low intra-class diversity and high inter-class diversity, the initial purity score following 1 epoch of CRP is very similar to the MNIST baseline experiments at 0.8939, illustrated by the high correlation of each classes' sample assignments to single GANs or clusters, shown in Figure 5.10.

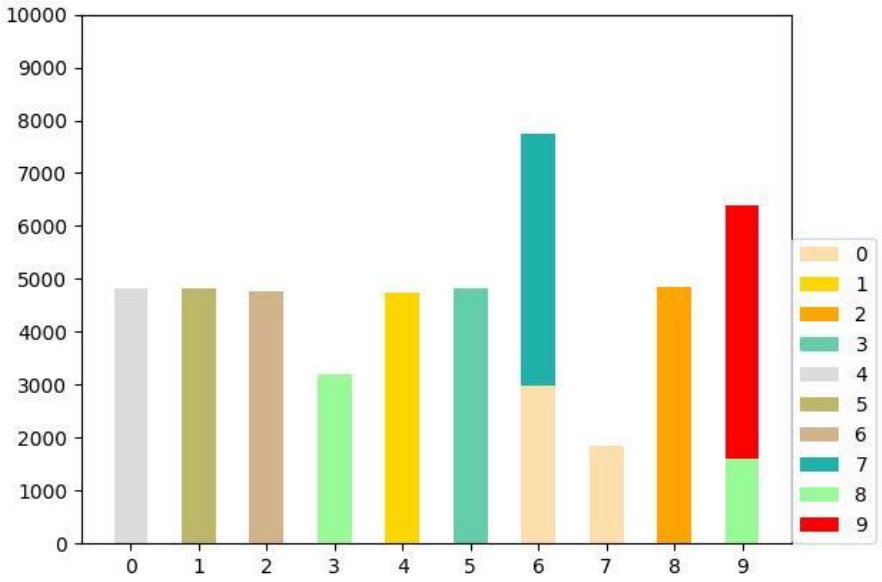

Figure 5.10: Number of samples (left) assigned to each cluster 0-9 (bottom).

However, after the next CRP epoch, the purity plummets to 0.2115, and by the third epoch the purity plateaus at 0.1010, due to all clusters but one being removed through the CRP process.

We investigated the synthesised samples per cluster, which made apparent the problem with applying the ACRP algorithm in our context. Unlike with the MNIST training, our audio GANs were mode switching. Specifically, between iterations, clusters could swap which class they would collapse to, whereas training with MNIST the clusters consistently collapsed to the same class/es throughout both stages of training. Initially this isn't a problem, as clustering is strong through mode collapse by itself, resulting in high classification accuracy through the IGMM and encoder computed likelihoods on real data. However, by the next epoch, the trained encoder is provided with completely different data per cluster that it was previously trained on, meaning their embeddings poorly correlate with the original class and the computed likelihood that each sample belongs to their correct cluster falls dramatically. This in turn results in extremely low likelihoods for real samples, and the CRP to therefore remove most, if not all but one, cluster or GAN. In Figure 5.11, each row corresponds to generated samples for an individual GAN after initialisation training, which in the left image shows clear mode collapse in each of the GANs. However, in the right image we can see some of the GANs have swapped the modes they have collapsed to, resulting in the problems we have described.

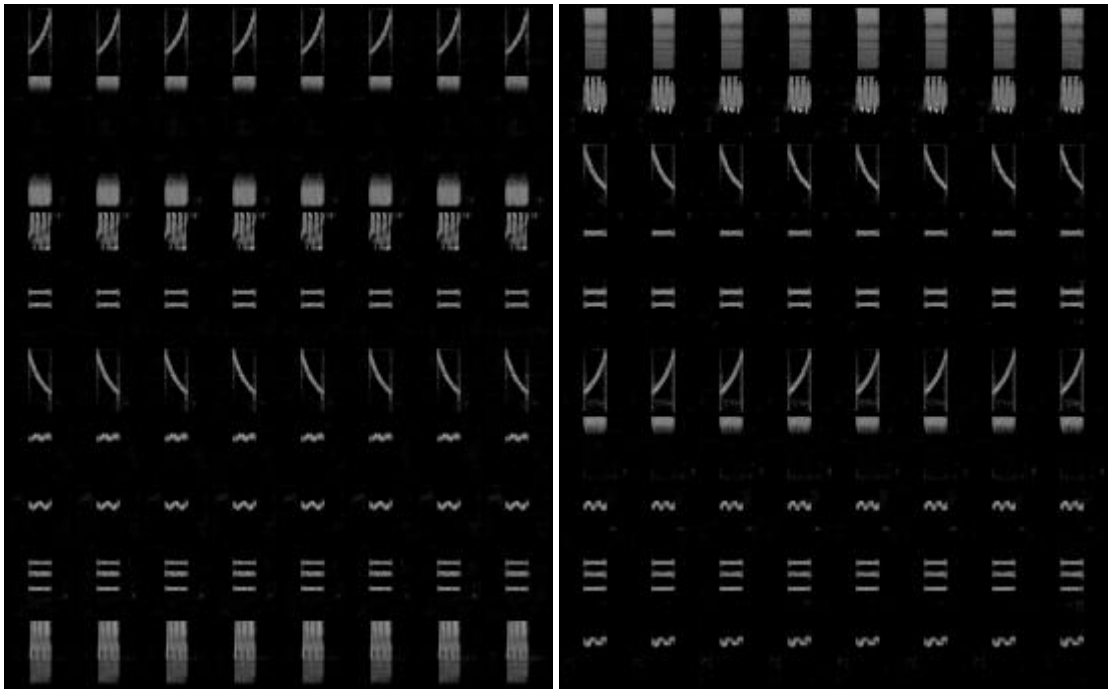

Figure 5.11: Synthesised samples for clusters 0-9 (rows top to bottom) following initial GAN training used in CRP epoch 1 (left), and synthesised samples for the same clusters after further GAN training during the CRP process has caused mode switching (right).

5.6 Conclusion

In conclusion, we have explored the potential for applying a mixture of infinite conditional GANs to the task of automatic audio clustering, classification and synthesis with both a DCGAN based architecture 'SpecGAN' and a StyleGAN based architecture 'ASGAN' through leveraging mode-collapse. Although mode collapse does occur in some experiments with SC09, the degree to which this happens is not enough for the encoder to assign high likelihood to individual modes of the data representing classes within the dataset to each individual GAN. We have verified the quality of synthesised samples is high enough for intelligibility and understanding both objectively and subjectively from the human perspective. The tendency for the GANs to produce similar samples through mode collapse occurs much more clearly when training with ESC50, but not enough to totally capture the individual classes. This is likely due to the high variability in the data making mode collapse more rewarding to the generator, but still too complex for there to be simple replicable patterns amongst the audio to reliably produce a variety of samples from the same class through collapse. Using our synthesised dataset, we show given audio data with high inter-class diversity and low intra-class diversity, similar to MNIST in the audio domain, mode collapse can capture class modes given a conditional embedding without any labelled data or supervision, resulting in high classification accuracy. However, due to the mode switching behaviour during audio GAN training, the CRP process cannot reliably refine the data into clusters, preventing the automatic conversion towards the true number of classes impossible in its current state. Due to these results, we believe this framework to be unsuitable for automatic clustering and classification by exploiting mode collapse on

the datasets and models we have tested. Further work could explore methods of constraining or punishing dramatic changes in the variety of samples produced by each individual GAN to mitigate this issue, which may then see utility with the ACRP algorithm, though this would still rely on very distinct classes such as those in our synthesised dataset.

Chapter 6

6 Conclusion

This thesis represents our work addressing the task of universal sound separation and synthesis using unsupervised and unconditional methods. We proposed a model with which to incorporate the learning of long-term dependencies and high-detail frequency features within sound mixtures utilizing transformers without drastically increasing computational cost through axial attention in Chapter 3. We then address the improvement that can be gained through class conditioning in universal sound separation and the lack of labelled data available through the use of unsupervised generative VAE models to automatically structure latent class encodings for use in semi-supervised classification through KL-divergence computation in Chapter 4. Finally, we propose a truly unsupervised classification solution using GANs conditioned on cluster labels discovered through leveraging mode-collapse with an infinite Gaussian mixture model and an adversarial Chinese Restaurant Process for clustering, classification and generation of audio classes in Chapter 5.

Although we see potential in our solutions, there is room to push them further. Our U-AxialNet showed slight improvements over the vanilla U-Net with the incorporation of an efficient axial-transformer bottleneck to incorporate attention whilst keeping computational costs low. However, the model is too low capacity for effective separation in its current state. Further work could incorporate the axial transformer in a much deeper network with the axial cross-attention skip connections we proposed, though axial attention itself may be too sparse for effective application in the audio domain. Through training VAEs to structure and disentangle the latent space of a given audio dataset, we showed through computation of KL-divergence and semi-supervised classification using a relatively small number of labelled samples that unknown in-domain audio clips could be sampled with moderately high-accuracy, though this is comparable with traditional fully unsupervised methods such as the KMeans algorithm. Our classification through the latent encodings with VAEs could likely be improved with much deeper models and the incorporation of training for specific clustering behaviour to more strongly structure the latent space. Lastly, our generative model using infinite GANs shows excellent generative performance, but limited classification and mode capture due to the intrinsically high intra-class diversity of audio data. We observe that eliminating this intra-class diversity and ensuring each class is distinct from all others, classification by leveraging mode collapse is certainly possible, though the GANs can mode switch, resulting the ACRP algorithm being unapplicable in the current framework for discovering the true number of class modes in an audio dataset.

References

Agrawal, J., Gupta, M., Garg, H., 2023. A review on speech separation in cocktail party environment: challenges and approaches. Multimed. Tools Appl. 82, 31035–31067. https://doi.org/10.1007/s11042-023-14649-x

Aldous, D.J., 1985. Exchangeability and related topics, in: Aldous, D.J., Ibragimov, I.A., Jacod, J., Hennequin, P.L. (Eds.), École d'Été de Probabilités de Saint-Flour XIII — 1983. Springer, Berlin, Heidelberg, pp. 1–198. https://doi.org/10.1007/BFb0099421

Amorim, L.B.V. de, Cavalcanti, G.D.C., Cruz, R.M.O., 2023. The choice of scaling technique matters for classification performance. Appl. Soft Comput. 133, 109924. https://doi.org/10.1016/j.asoc.2022.109924

Anggoro, D., Supriyanti, W., 2019. Improving Accuracy by applying Z-Score Normalization in Linear Regression and Polynomial Regression Model for Real Estate Data. Int. J. Emerg. Trends Technol. Comput. Sci. 549–555. https://doi.org/10.30534/ijeter/2019/247112019

Arjovsky, M., Bottou, L., 2017. Towards Principled Methods for Training Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1701.04862

Arjovsky, M., Chintala, S., Bottou, L., 2017a. Wasserstein GAN. https://doi.org/10.48550/arXiv.1701.07875

Arjovsky, M., Chintala, S., Bottou, L., 2017b. Wasserstein GAN. https://doi.org/10.48550/arXiv.1701.07875

Arora, S., Ge, R., Liang, Y., Ma, T., Zhang, Y., 2017. Generalization and Equilibrium in Generative Adversarial Nets (GANs). https://doi.org/10.48550/arXiv.1703.00573

Asperti, A., Trentin, M., 2020a. Balancing reconstruction error and Kullback-Leibler divergence in Variational Autoencoders. https://doi.org/10.48550/arXiv.2002.07514

Asperti, A., Trentin, M., 2020b. Balancing Reconstruction Error and Kullback-Leibler Divergence in Variational Autoencoders. IEEE Access 8, 199440–199448. https://doi.org/10.1109/ACCESS.2020.3034828

Baas, M., Kamper, H., 2022. GAN You Hear Me? Reclaiming Unconditional Speech Synthesis from Diffusion Models. https://doi.org/10.48550/arXiv.2210.05271

Bai, J., Kong, S., Gomes, C.P., 2022. Gaussian Mixture Variational Autoencoder with Contrastive Learning for Multi-Label Classification. https://doi.org/10.48550/arXiv.2112.00976

Bitton, A., Esling, P., Chemla-Romeu-Santos, A., 2018. Modulated Variational auto-Encoders for many-to-many musical timbre transfer. https://doi.org/10.48550/arXiv.1810.00222

Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S., 2016. Generating Sentences from a Continuous Space. https://doi.org/10.48550/arXiv.1511.06349

References

Božić, M., Horvat, M., 2024. A Survey of Deep Learning Audio Generation Methods. https://doi.org/10.48550/arXiv.2406.00146

Bredell, G., Flouris, K., Chaitanya, K., Erdil, E., Konukoglu, E., 2023. Explicitly Minimizing the Blur Error of Variational Autoencoders. https://doi.org/10.48550/arXiv.2304.05939

Caillon, A., Esling, P., 2021. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. https://doi.org/10.48550/arXiv.2111.05011

Chang, X., Zhang, W., Qian, Y., Roux, J.L., Watanabe, S., 2019. MIMO-Speech: End-to-End Multi-Channel Multi-Speaker Speech Recognition, in: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). Presented at the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 237–244. https://doi.org/10.1109/ASRU46091.2019.9003986

Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W., 2017. Mode Regularized Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1612.02136

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P., 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. https://doi.org/10.48550/arXiv.1606.03657

Chen, Z., Luo, Y., Mesgarani, N., 2017. Deep attractor network for single-microphone speaker separation, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 246–250. https://doi.org/10.1109/ICASSP.2017.7952155

Cheng, B., Schwing, A.G., Kirillov, A., 2021. Per-Pixel Classification is Not All You Need for Semantic Segmentation. https://doi.org/10.48550/arXiv.2107.06278

Cherry, E.C., 1953. Some Experiments on the Recognition of Speech, with One and With Two Ears. J. Acoust. Soc. Am. 25.

Child, R., Gray, S., Radford, A., Sutskever, I., 2019. Generating Long Sequences with Sparse Transformers. https://doi.org/10.48550/arXiv.1904.10509

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., Weller, A., 2022. Rethinking Attention with Performers. https://doi.org/10.48550/arXiv.2009.14794

Cosentino, J., Pariente, M., Cornell, S., Deleforge, A., Vincent, E., 2020. LibriMix: An Open-Source Dataset for Generalizable Speech Separation. https://doi.org/10.48550/arXiv.2005.11262

Dauphin, Y.N., Fan, A., Auli, M., Grangier, D., 2017. Language Modeling with Gated Convolutional Networks. https://doi.org/10.48550/arXiv.1612.08083

Défossez, A., 2022. Hybrid Spectrogram and Waveform Source Separation. https://doi.org/10.48550/arXiv.2111.03600

References

Défossez, A., Usunier, N., Bottou, L., Bach, F., 2019. Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed. ArXiv190901174 Cs Eess Stat.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. J. R. Stat. Soc. Ser. B Methodol. 39, 1–38.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Presented at the 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dhariwal, P., Jun, H., Payne, C., Kim, J.W., Radford, A., Sutskever, I., 2020. Jukebox: A Generative Model for Music. https://doi.org/10.48550/arXiv.2005.00341

Donahue, C., McAuley, J., Puckette, M., 2019a. Adversarial Audio Synthesis. https://doi.org/10.48550/arXiv.1802.04208

Donahue, C., McAuley, J., Puckette, M., 2019b. Adversarial Audio Synthesis. https://doi.org/10.48550/arXiv.1802.04208

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N., 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://doi.org/10.48550/arXiv.2010.11929

Drgas, S., 2023. A Survey on Low-Latency DNN-Based Speech Enhancement. Sensors 23, 1380. https://doi.org/10.3390/s23031380

Engel, J., Hoffman, M., Roberts, A., 2017. Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models. https://doi.org/10.48550/arXiv.1711.05772

Ephrat, A., Mosseri, I., Lang, O., Dekel, T., Wilson, K., Hassidim, A., Freeman, W.T., Rubinstein, M., 2018. Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. ACM Trans. Graph. 37, 1–11. https://doi.org/10.1145/3197517.3201357

Fil, M., Mesinovic, M., Morris, M., Wildberger, J., 2021. Beta-VAE Reproducibility: Challenges and Extensions. https://doi.org/10.48550/arXiv.2112.14278

Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., Carin, L., 2019. Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. https://doi.org/10.48550/arXiv.1903.10145

Ganesh, P., 2019. Types of Convolution Kernels : Simplified [WWW Document]. Data Sci. URL https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37/ (accessed 2.28.25).

Gemmeke, J.F., Ellis, D.P.W., Freedman, D., Jansen, A., Lawrence, W., Moore, R.C., Plakal, M., Ritter, M., 2017. Audio Set: An ontology and human-labeled dataset for audio events, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the 2017 IEEE International Conference on Acoustics, Speech and Signal

Processing (ICASSP), IEEE, New Orleans, LA, pp. 776–780. https://doi.org/10.1109/ICASSP.2017.7952261

Gharehpetian, G.B., Karami, H., 2023. Power Transformer Online Monitoring Using Electromagnetic Waves, in: Power Transformer Online Monitoring Using Electromagnetic Waves. pp. 89–113.

Ghose, S., Prevost, J.J., 2021. AutoFoley: Artificial Synthesis of Synchronized Sound Tracks for Silent Videos with Deep Learning. IEEE Trans. Multimed. 23, 1895–1907. https://doi.org/10.1109/TMM.2020.3005033

Ghosh, A., Kulharia, V., Namboodiri, V., Torr, P.H.S., Dokania, P.K., 2018. Multi-Agent Diverse Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1704.02906

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1406.2661

Goyal, D., Pabla, B.S., 2015. Condition based maintenance of machine tools—A review. CIRP J. Manuf. Sci. Technol. 30–31.

Griffin, D., Lim, J., 1984. Signal estimation from modified short-time Fourier transform. IEEE Trans. Acoust. Speech Signal Process. 32, 236–243. https://doi.org/10.1109/TASSP.1984.1164317

Gurumurthy, S., Sarvadevabhatla, R.K., Radhakrishnan, V.B., 2017. DeLiGAN : Generative Adversarial Networks for Diverse and Limited Data. https://doi.org/10.48550/arXiv.1706.02071

Hennequin, R., Khlif, A., Voituret, F., Moussallam, M., 2019. SPLEETER: A FAST AND STATE-OF-THE ART MUSIC SOURCE SEPARATION TOOL WITH PRE-TRAINED MODELS.

Hershey, J.R., Chen, Z., Roux, J.L., Watanabe, S., 2015. Deep clustering: Discriminative embeddings for segmentation and separation. https://doi.org/10.48550/arXiv.1508.04306

Hershey, S., Chaudhuri, S., Ellis, D.P.W., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B., Slaney, M., Weiss, R.J., Wilson, K., 2017. CNN Architectures for Large-Scale Audio Classification. https://doi.org/10.48550/arXiv.1609.09430

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S., 2018. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. https://doi.org/10.48550/arXiv.1706.08500

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A., 2017. β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK.

Ho, J., Kalchbrenner, N., Weissenborn, D., Salimans, T., 2019. Axial Attention in Multidimensional Transformers. ArXiv191212180 Cs.

Hoang, Q., Nguyen, T.D., Le, T., Phung, D., 2018. MGAN: TRAINING GENERATIVE ADVERSARIAL NETS WITH MULTIPLE GENERATORS.

References

Hsu, W.-N., Bolte, B., Tsai, Y.-H.H., Lakhotia, K., Salakhutdinov, R., Mohamed, A., 2021. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. https://doi.org/10.48550/arXiv.2106.07447

Hsu, W.-N., Zhang, Y., Glass, J., 2017. Learning Latent Representations for Speech Generation and Transformation. https://doi.org/10.48550/arXiv.1704.04222

Isik, Y., Roux, J.L., Chen, Z., Watanabe, S., Hershey, J.R., 2016. Single-Channel Multi-Speaker Separation using Deep Clustering. https://doi.org/10.48550/arXiv.1607.02173

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., Carreira, J., 2021. Perceiver: General Perception with Iterative Attention. https://doi.org/10.48550/arXiv.2103.03206

Jiralerspong, M., Gidel, G., 2022. Generating Diverse Vocal Bursts with StyleGAN2 and MEL-Spectrograms. https://doi.org/10.48550/arXiv.2206.12563

Jolliffe, I.T., Cadima, J., 2016. Principal component analysis: a review and recent developments. https://doi.org/10.1098/rsta.2015.0202

Karras, T., Laine, S., Aila, T., 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1812.04948

Khanam, F., Munmun, F.A., Ritu, N.A., Saha, A.K., Mridha, M.F., 2022. Text to Speech Synthesis: A Systematic Review, Deep Learning Based Architecture and Future Research Direction. J. Adv. Inf. Technol. 13. https://doi.org/10.12720/jait.13.5.398-412

Kilgour, K., Zuluaga, M., Roblek, D., Sharifi, M., 2019. Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms. https://doi.org/10.48550/arXiv.1812.08466

Kim, Y.-S., Kim, M.K., Fu, N., Liu, J., Wang, J., Srebric, J., 2025. Investigating the impact of data normalization methods on predicting electricity consumption in a building using different artificial neural network models. Sustain. Cities Soc. 118, 105570. https://doi.org/10.1016/j.scs.2024.105570

Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980

Kingma, D.P., Welling, M., 2022. Auto-Encoding Variational Bayes. https://doi.org/10.48550/arXiv.1312.6114

Kingma, D.P., Welling, M., 2019. An Introduction to Variational Autoencoders. Found. Trends® Mach. Learn. 12, 307–392. https://doi.org/10.1561/2200000056

Kong, J., Kim, J., Bae, J., 2020. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. https://doi.org/10.48550/arXiv.2010.05646

Kong, Q., Chen, K., Liu, H., Du, X., Berg-Kirkpatrick, T., Dubnov, S., Plumbley, M.D., 2023. Universal Source Separation with Weakly Labelled Data. https://doi.org/10.48550/arXiv.2305.07447

Kong, Z., Ping, W., Huang, J., Zhao, K., Catanzaro, B., 2021. DiffWave: A Versatile Diffusion Model for Audio Synthesis. https://doi.org/10.48550/arXiv.2009.09761

References

Kossale, Y., Airaj, M., Darouichi, A., 2022a. Mode Collapse in Generative Adversarial Networks: An Overview, in: 2022 8th International Conference on Optimization and Applications (ICOA). Presented at the 2022 8th International Conference on Optimization and Applications (ICOA), pp. 1–6. https://doi.org/10.1109/ICOA55659.2022.9934291

Kossale, Y., Airaj, M., Darouichi, A., 2022b. Mode Collapse in Generative Adversarial Networks: An Overview, in: 2022 8th International Conference on Optimization and Applications (ICOA). Presented at the 2022 8th International Conference on Optimization and Applications (ICOA), pp. 1–6. https://doi.org/10.1109/ICOA55659.2022.9934291

Kullback, S., Leibler, R.A., 1951. On Information and Sufficiency. Ann. Math. Stat. 22, 79–86. https://doi.org/10.1214/aoms/1177729694

Kumar, K., Kumar, R., Boissiere, T. de, Gestin, L., Teoh, W.Z., Sotelo, J., Brebisson, A. de, Bengio, Y., Courville, A., 2019. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. https://doi.org/10.48550/arXiv.1910.06711

Langlotz, C.P., Allen, B., Erickson, B.J., Kalpathy-Cramer, J., Bigelow, K., Cook, T.S., Flanders, A.E., Lungren, M.P., Mendelson, D.S., Rudie, J.D., Wang, G., Kandarpa, K., 2019. A Roadmap for Foundational Research on Artificial Intelligence in Medical Imaging: From the 2018 NIH/RSNA/ACR/The Academy Workshop. Radiology 291, 781–791. https://doi.org/10.1148/radiol.2019190613

Lavault, A., 2023. Generative Adversarial Networks for Synthesis and Control of Drum Sounds (phdthesis). Sorbonne Université.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2324. https://doi.org/10.1109/5.726791

Lee, D.D., Seung, H.S., 2001. Algorithms for Non-negative Matrix Factorization.

Lee, J., Jeon, Y., Lee, W., Kim, Y., Lee, G.G., 2023. Exploring the Viability of Synthetic Audio Data for Audio-Based Dialogue State Tracking. https://doi.org/10.48550/arXiv.2312.01842

Lee, J., Lee, Y., Kim, J., Kosiorek, A.R., Choi, S., Teh, Y.W., 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. https://doi.org/10.48550/arXiv.1810.00825

Li Deng, 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. IEEE Signal Process. Mag. 29, 141–142. https://doi.org/10.1109/MSP.2012.2211477

Li, J., Yang, H., 2021. The underwater acoustic target timbre perception and recognition based on the auditory inspired deep convolutional neural network. Appl. Acoust. 182, 108210. https://doi.org/10.1016/j.apacoust.2021.108210

Li, P., Pei, Y., Li, J., 2023. A comprehensive survey on design and application of autoencoder in deep learning. Appl. Soft Comput. 138, 110176. https://doi.org/10.1016/j.asoc.2023.110176

References

Li, Y., Shao, X., Zhang, J., Wang, H., Brunswic, L.M., Zhou, K., Dong, J., Guo, K., Li, X., Chen, Z., Wang, J., Hao, J., 2025. Generative Models in Decision Making: A Survey. https://doi.org/10.48550/arXiv.2502.17100

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021a. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. https://doi.org/10.48550/arXiv.2103.14030

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021b. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. https://doi.org/10.48550/arXiv.2103.14030

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Trans. Inf. Theory 28, 129–137. https://doi.org/10.1109/TIT.1982.1056489

Lu, H., Wang, D., Wu, X., Wu, Z., Liu, X., Meng, H., 2022. Disentangled Speech Representation Learning for One-Shot Cross-lingual Voice Conversion Using $β$-VAE. https://doi.org/10.48550/arXiv.2210.13771

Lu, W.-T., Wang, J.-C., Kong, Q., Hung, Y.-N., 2023. Music Source Separation with Band-Split RoPE Transformer. https://doi.org/10.48550/arXiv.2309.02612

Luo, Y., Mesgarani, N., 2019. Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation. IEEEACM Trans. Audio Speech Lang. Process. 27, 1256–1266. https://doi.org/10.1109/TASLP.2019.2915167

Luo, Y., Yu, J., 2022. Music Source Separation with Band-split RNN. https://doi.org/10.48550/arXiv.2209.15174

Maaten, L. van der, Hinton, G., 2008. Visualizing Data using t-SNE. J. Mach. Learn. Res. 9, 2579–2605.

Maciejewski, M., Wichern, G., McQuinn, E., Roux, J.L., 2020. WHAMR!: Noisy and Reverberant Single-Channel Speech Separation. https://doi.org/10.48550/arXiv.1910.10279

Manning, C., Raghavan, P., Schuetze, H., 2009. Introduction to Information Retrieval.

Mao, Q., Lee, H.-Y., Tseng, H.-Y., Ma, S., Yang, M.-H., 2019. Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis. https://doi.org/10.48550/arXiv.1903.05628

Mao, X., Li, Q., Xie, H., Lau, R.Y.K., Wang, Z., Smolley, S.P., 2017a. Least Squares Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1611.04076

Mao, X., Li, Q., Xie, H., Lau, R.Y.K., Wang, Z., Smolley, S.P., 2017b. Least Squares Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1611.04076

Marrinan, T., Akram, P., Gurmessa, O., Shishkin, A., 2024. Leveraging AI to Generate Audio for User-generated Content in Video Games. https://doi.org/10.48550/arXiv.2404.17018

Matevosyan, G., 2023. Audio Source Separation. Unleashing the Power of Non-Negative Matrix Factorization: A Python Implementation. [WWW Document]. URL

https://gormatevosyan.com/audio-source-spearation-with-non-negative-matrix-factorization/ (accessed 2.26.25).

Meier, B., 2017. A Survey on Voice Conversion using Deep Learning.

Mirbeygi, M., Mahabadi, A., Ranjbar, A., 2022. Speech and music separation approaches - a survey. Multimed. Tools Appl.

Mirza, M., Osindero, S., 2014. Conditional Generative Adversarial Nets. https://doi.org/10.48550/arXiv.1411.1784

Murel, J., 2024. What is a feature engineering? | IBM [WWW Document]. URL https://www.ibm.com/think/topics/feature-engineering (accessed 2.27.25).

Nath, K., Sarma, K.K., 2024. Separation of overlapping audio signals: A review on current trends and evolving approaches. Signal Process. 221, 109487. https://doi.org/10.1016/j.sigpro.2024.109487

Nguyen, T.Q., 1994. Near-perfect-reconstruction pseudo-QMF banks. IEEE Trans. Signal Process. 42, 65–76. https://doi.org/10.1109/78.258122

Noor, M.H.M., Ige, A.O., 2024. A Survey on State-of-the-art Deep Learning Applications and Challenges. https://doi.org/10.48550/arXiv.2403.17561

Omray, A., 2021. Introduction to Normalizing Flows [WWW Document]. Data Sci. URL https://towardsdatascience.com/introduction-to-normalizing-flows-d002af262a4b/ (accessed 4.2.25).

Oord, A. van den, Vinyals, O., Kavukcuoglu, K., 2018. Neural Discrete Representation Learning. https://doi.org/10.48550/arXiv.1711.00937

Otten, N.V., 2023. Non-Negative Matrix Factorization Explained & Practical How To Guide In Python [WWW Document]. Spot Intell. URL https://spotintelligence.com/2023/09/08/non-negative-matrix-factorization/ (accessed 2.26.25).

Oxenham, A.J., 2018. How We Hear: The Perception and Neural Coding of Sound. Annu. Rev. Psychol. 69, 27–50. https://doi.org/10.1146/annurev-psych-122216-011635

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training Recurrent Neural Networks. https://doi.org/10.48550/arXiv.1211.5063

Pedersen, P., 1965. The Mel Scale. J. Music Theory 9, 295–308.

Petit, O., Thome, N., Rambour, C., Soler, L., 2021. U-Net Transformer: Self and Cross Attention for Medical Image Segmentation. https://doi.org/10.48550/arXiv.2103.06104

Pimpale, M.R., Therese, S., Shinde, V., 2016. A Survey on: Sound Source Separation Methods 5.

Platkiewicz, J., Brette, R., 2010. A Threshold Equation for Action Potential Initiation. PLoS Comput. Biol. 6, e1000850. https://doi.org/10.1371/journal.pcbi.1000850

Radford, A., Metz, L., Chintala, S., 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1511.06434

References

Ramachandran, P., Zoph, B., Le, Q.V., 2017. Searching for Activation Functions. https://doi.org/10.48550/arXiv.1710.05941

Razzaq, A., 2020. Google Open-Sources FUSS: The Free Universal Sound Separation Dataset. MarkTechPost. URL https://www.marktechpost.com/2020/04/09/google-open-sources-fuss-the-free-universal-sound-separation-dataset/ (accessed 11.2.20).

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., Eck, D., 2019. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. https://doi.org/10.48550/arXiv.1803.05428

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.48550/arXiv.1505.04597

Rouard, S., Massa, F., Défossez, A., 2022. Hybrid Transformers for Music Source Separation. https://doi.org/10.48550/arXiv.2211.08553

Roux, J.L., Wisdom, S., Erdogan, H., Hershey, J.R., 2018. SDR - half-baked or well done? https://doi.org/10.48550/arXiv.1811.02508

Saijo, K., Wichern, G., Germain, F.G., Pan, Z., Roux, J.L., 2024. TF-Locoformer: Transformer with Local Modeling by Convolution for Speech Separation and Enhancement. https://doi.org/10.48550/arXiv.2408.03440

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved Techniques for Training GANs. https://doi.org/10.48550/arXiv.1606.03498

Sandu, C., 2024. Sequence-to-Sequence Models. Medium. URL https://medium.com/@calin.sandu/sequence-to-sequence-models-603920ce9e96 (accessed 2.28.25).

Sharma, J., 2017. Swish in depth: A comparison of Swish & ReLU on CIFAR-10. Medium. URL https://medium.com/@jaiyamsharma/swish-in-depth-a-comparison-of-swish-relu-on-cifar-10-1c798e70ee08 (accessed 2.28.25).

Shin, U.-H., Lee, S., Kim, T., Park, H.-M., 2025. Separate and Reconstruct: Asymmetric Encoder-Decoder for Speech Separation. https://doi.org/10.48550/arXiv.2406.05983

Shu, L., Xu, H., Liu, B., 2018. Unseen Class Discovery in Open-world Classification. https://doi.org/10.48550/arXiv.1801.05609

Slizovskaia, O., Kim, L., Haro, G., Gomez, E., 2019. End-to-end Sound Source Separation Conditioned on Instrument Labels, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 306–310. https://doi.org/10.1109/ICASSP.2019.8683800

Sohn, K., Lee, H., Yan, X., 2015. Learning Structured Output Representation using Deep Conditional Generative Models.

Stöter, F.-R., Liutkus, A., Ito, N., 2018. The 2018 Signal Separation Evaluation Campaign. https://doi.org/10.48550/arXiv.1804.06267

References

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., Liu, Y., 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding. https://doi.org/10.48550/arXiv.2104.09864

Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., Zhong, J., 2021a. Attention is All You Need in Speech Separation. https://doi.org/10.48550/arXiv.2010.13154

Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., Zhong, J., 2021b. Attention is All You Need in Speech Separation. https://doi.org/10.48550/arXiv.2010.13154

Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., Zhong, J., 2021c. Attention is All You Need in Speech Separation. https://doi.org/10.48550/arXiv.2010.13154

Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R., 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. https://doi.org/10.48550/arXiv.2006.10739

Tay, Y., Dehghani, M., Bahri, D., Metzler, D., 2023. Efficient Transformers: A Survey. ACM Comput. Surv. 55, 1–28. https://doi.org/10.1145/3530811

Tong, W., Zhu, J., Chen, J., Kang, S., Jiang, T., Li, Y., Wu, Z., Meng, H., 2024. SCNet: Sparse Compression Network for Music Source Separation. https://doi.org/10.48550/arXiv.2401.13276

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H., 2021. Going deeper with Image Transformers. https://doi.org/10.48550/arXiv.2103.17239

Tripathi, A.M., Mishra, A., 2021. Environment sound classification using an attention-based residual neural network. Neurocomputing 460, 409–423. https://doi.org/10.1016/j.neucom.2021.06.031

Turner, R.E., 2024. An Introduction to Transformers. https://doi.org/10.48550/arXiv.2304.10557

Tzinis, Efthymios, Wang, Z., Smaragdis, P., 2020a. Sudo rm -rf: Efficient Networks for Universal Audio Source Separation, in: 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP). pp. 1–6. https://doi.org/10.1109/MLSP49062.2020.9231900

Tzinis, E., Wang, Z., Smaragdis, P., 2020. Sudo RM -RF: Efficient Networks for Universal Audio Source Separation, in: 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP). Presented at the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6. https://doi.org/10.1109/MLSP49062.2020.9231900

Tzinis, Efthymios, Wisdom, S., Hershey, J.R., Jansen, A., Ellis, D.P.W., 2020b. Improving Universal Sound Separation Using Sound Classification, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 96–100. https://doi.org/10.1109/ICASSP40776.2020.9053921

Ulhaq, A., Akhtar, N., 2024. Efficient Diffusion Models for Vision: A Survey. https://doi.org/10.48550/arXiv.2210.09292

References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2023. Attention Is All You Need. https://doi.org/10.48550/arXiv.1706.03762

Vechtomova, O., Sahu, G., Kumar, D., 2021. LyricJam: A system for generating lyrics for live instrumental music. https://doi.org/10.48550/arXiv.2106.01960

Vechtomova, O., Sahu, G., Kumar, D., 2020. Generation of lyrics lines conditioned on music audio clips. https://doi.org/10.48550/arXiv.2009.14375

Wang, A., Blair, N., Belkhale, S., n.d. Encouraging Categorical Meaning in the Latent Space of a VAE.

Wang, S., Li, B.Z., Khabsa, M., Fang, H., Ma, H., 2020. Linformer: Self-Attention with Linear Complexity. https://doi.org/10.48550/arXiv.2006.04768

Wichern, G., Antognini, J., Flynn, M., Zhu, L.R., McQuinn, E., Crow, D., Manilow, E., Roux, J.L., 2019. WHAM!: Extending Speech Separation to Noisy Environments. https://doi.org/10.48550/arXiv.1907.01160

Wisdom, S., Erdogan, H., Ellis, D., Serizel, R., Turpault, N., Fonseca, E., Salamon, J., Seetharaman, P., Hershey, J., 2020. What's All the FUSS About Free Universal Sound Separation Data? ArXiv201100803 Cs Eess.

Woodruff, J., Li, Y., Wang, D., 2008. RESOLVING OVERLAPPING HARMONICS FOR MONAURAL MUSICAL SOUND SEPARATION USING PITCH AND COMMON AMPLITUDE MODULATION.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated Residual Transformations for Deep Neural Networks. https://doi.org/10.48550/arXiv.1611.05431

Ying, H., Wang, H., Shao, T., Yang, Y., Zhou, K., 2021. Unsupervised Image Generation with Infinite Generative Adversarial Networks. https://doi.org/10.48550/arXiv.2108.07975

Yu, D., Kolbæk, M., Tan, Z.-H., Jensen, J., 2017a. Permutation Invariant Training of Deep Models for Speaker-Independent Multi-talker Speech Separation. https://doi.org/10.48550/arXiv.1607.00325

Yu, D., Kolbæk, M., Tan, Z.-H., Jensen, J., 2017b. Permutation Invariant Training of Deep Models for Speaker-Independent Multi-talker Speech Separation. https://doi.org/10.48550/arXiv.1607.00325

Zhao, J., Liu, X., Zhao, Jinzheng, Yuan, Y., Kong, Q., Plumbley, M.D., Wang, W., 2024. Universal Sound Separation with Self-Supervised Audio Masked Autoencoder. https://doi.org/10.48550/arXiv.2407.11745

Zhao, S., Ma, Y., Ni, C., Zhang, C., Wang, H., Nguyen, T.H., Zhou, K., Yip, J., Ng, D., Ma, B., 2024. MossFormer2: Combining Transformer and RNN-Free Recurrent Network for Enhanced Time-Domain Monaural Speech Separation. https://doi.org/10.48550/arXiv.2312.11825

Zhou, Z., Cai, H., Rong, S., Song, Y., Ren, K., Zhang, W., Yu, Y., Wang, J., 2018. Activation Maximization Generative Adversarial Nets. https://doi.org/10.48550/arXiv.1703.02000

Appendix

The codebase for this thesis, including model architectures, will be made available at:

https://github.com/ShaunBuckley/Unsupervised-Detection-and-Synthesis-of-Speech-and-Environmental-Sounds-Using-Generative-Networks

Supplementary materials for Chapter 3

We provide a trace using the torchinfo library of our base U-Net model in Figure 7.1 for a clear understanding of the architecture. A typical configuration for using this can be found in our Github repository.

Table 7.1: U-Net Architecture (no axial bottleneck)

Layer (type:depth-idx)	Output Shape	Param #
├─MelLikeEncoder: 1-1	[-1, 1, 128, 79]	--
│ └─Conv1d: 2-1	[-1, 128, 79]	131,200
├─UNet: 1-2	[-1, 2, 128, 80]	--
│ └─ModuleList: 2	[]	--
│ │ └─DownsampleBlock: 3-1	[-1, 64, 64, 40]	--
│ │ │ └─ConvStack: 4-1	[-1, 64, 128, 80]	--
│ │ │ │ └─Sequential: 5-1	[-1, 64, 128, 80]	--
│ │ │ │ │ └─Conv2d: 6-1	[-1, 64, 128, 80]	640
│ │ │ │ │ └─BatchNorm2d: 6-2	[-1, 64, 128, 80]	128
│ │ │ │ │ └─ReLU: 6-3	[-1, 64, 128, 80]	--
│ │ │ │ │ └─Conv2d: 6-4	[-1, 64, 128, 80]	36,928
│ │ │ │ │ └─BatchNorm2d: 6-5	[-1, 64, 128, 80]	128
│ │ │ │ │ └─ReLU: 6-6	[-1, 64, 128, 80]	--
│ │ │ └─MaxPool2d: 4-2	[-1, 64, 64, 40]	--
│ │ └─DownsampleBlock: 3-2	[-1, 128, 32, 20]	--
│ │ │ └─ConvStack: 4-3	[-1, 128, 64, 40]	--
│ │ │ │ └─Sequential: 5-2	[-1, 128, 64, 40]	--
│ │ │ │ │ └─Conv2d: 6-7	[-1, 128, 64, 40]	73,856
│ │ │ │ │ └─BatchNorm2d: 6-8	[-1, 128, 64, 40]	256
│ │ │ │ │ └─ReLU: 6-9	[-1, 128, 64, 40]	--

					└─Conv2d: 6-10	[-1, 128, 64, 40]	147,584
					└─BatchNorm2d: 6-11	[-1, 128, 64, 40]	256
					└─ReLU: 6-12	[-1, 128, 64, 40]	--
			└─MaxPool2d: 4-4			[-1, 128, 32, 20]	--
		└─DownsampleBlock: 3-3				[-1, 256, 16, 10]	--
			└─ConvStack: 4-5			[-1, 256, 32, 20]	--
				└─Sequential: 5-3		[-1, 256, 32, 20]	--
					└─Conv2d: 6-13	[-1, 256, 32, 20]	295,168
					└─BatchNorm2d: 6-14	[-1, 256, 32, 20]	512
					└─ReLU: 6-15	[-1, 256, 32, 20]	--
					└─Conv2d: 6-16	[-1, 256, 32, 20]	590,080
					└─BatchNorm2d: 6-17	[-1, 256, 32, 20]	512
					└─ReLU: 6-18	[-1, 256, 32, 20]	--
			└─MaxPool2d: 4-6			[-1, 256, 16, 10]	--
		└─DownsampleBlock: 3-4				[-1, 512, 8, 5]	--
			└─ConvStack: 4-7			[-1, 512, 16, 10]	--
				└─Sequential: 5-4		[-1, 512, 16, 10]	--
					└─Conv2d: 6-19	[-1, 512, 16, 10]	1,180,160
					└─BatchNorm2d: 6-20	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-21	[-1, 512, 16, 10]	--
					└─Conv2d: 6-22	[-1, 512, 16, 10]	2,359,808
					└─BatchNorm2d: 6-23	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-24	[-1, 512, 16, 10]	--
			└─MaxPool2d: 4-8			[-1, 512, 8, 5]	--
	└─ConvStack: 2-2					[-1, 512, 8, 5]	--
		└─Sequential: 3-5				[-1, 512, 8, 5]	--
			└─Conv2d: 4-9			[-1, 512, 8, 5]	2,359,808
			└─BatchNorm2d: 4-10			[-1, 512, 8, 5]	1,024
			└─ReLU: 4-11			[-1, 512, 8, 5]	--
			└─Conv2d: 4-12			[-1, 512, 8, 5]	2,359,808
			└─BatchNorm2d: 4-13			[-1, 512, 8, 5]	1,024
			└─ReLU: 4-14			[-1, 512, 8, 5]	--
	└─ModuleList: 2					[]	--
		└─UpsampleBlock: 3-6				[-1, 512, 16, 10]	--

			└─Upsample: 4-15	[-1, 512, 16, 10]	--
			└─ConvStack: 4-16	[-1, 512, 16, 10]	--
				└─Sequential: 5-5	[-1, 512, 16, 10]	--
					└─Conv2d: 6-25	[-1, 512, 16, 10]	4,719,104
					└─BatchNorm2d: 6-26	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-27	[-1, 512, 16, 10]	--
					└─Conv2d: 6-28	[-1, 512, 16, 10]	2,359,808
					└─BatchNorm2d: 6-29	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-30	[-1, 512, 16, 10]	--
		└─UpsampleBlock: 3-7				[-1, 256, 32, 20]	--
			└─Upsample: 4-17			[-1, 512, 32, 20]	--
			└─ConvStack: 4-18			[-1, 256, 32, 20]	--
				└─Sequential: 5-6		[-1, 256, 32, 20]	--
					└─Conv2d: 6-31	[-1, 256, 32, 20]	1,769,728
					└─BatchNorm2d: 6-32	[-1, 256, 32, 20]	512
					└─ReLU: 6-33	[-1, 256, 32, 20]	--
					└─Conv2d: 6-34	[-1, 256, 32, 20]	590,080
					└─BatchNorm2d: 6-35	[-1, 256, 32, 20]	512
					└─ReLU: 6-36	[-1, 256, 32, 20]	--
		└─UpsampleBlock: 3-8				[-1, 128, 64, 40]	--
			└─Upsample: 4-19			[-1, 256, 64, 40]	--
			└─ConvStack: 4-20			[-1, 128, 64, 40]	--
				└─Sequential: 5-7		[-1, 128, 64, 40]	--
					└─Conv2d: 6-37	[-1, 128, 64, 40]	442,496
					└─BatchNorm2d: 6-38	[-1, 128, 64, 40]	256
					└─ReLU: 6-39	[-1, 128, 64, 40]	--
					└─Conv2d: 6-40	[-1, 128, 64, 40]	147,584
					└─BatchNorm2d: 6-41	[-1, 128, 64, 40]	256
					└─ReLU: 6-42	[-1, 128, 64, 40]	--
		└─UpsampleBlock: 3-9				[-1, 64, 128, 80]	--
			└─Upsample: 4-21			[-1, 128, 128, 80]	--
			└─ConvStack: 4-22			[-1, 64, 128, 80]	--
				└─Sequential: 5-8		[-1, 64, 128, 80]	--
					└─Conv2d: 6-43	[-1, 64, 128, 80]	110,656

								└─BatchNorm2d: 6-44	[-1, 64, 128, 80]	128
								└─ReLU: 6-45	[-1, 64, 128, 80]	--
								└─Conv2d: 6-46	[-1, 64, 128, 80]	36,928
								└─BatchNorm2d: 6-47	[-1, 64, 128, 80]	128
								└─ReLU: 6-48	[-1, 64, 128, 80]	--
						└─Conv2d: 2-3			[-1, 2, 128, 80]	130
├─MelLikeDecoder: 1-3									[-1, 1, 40192]	--
						└─ConvTranspose1d: 2-4			[-1, 1, 40192]	131,073

We also provide a trace of our U-AxialNet model in Table 7.2, which can be activated in place of the baseline U-Net using the configuration file by changing the 'axial' parameter to True, an example of which can be found in our Github repository.

Table 7.2: U-AxialNet Architecture

Layer (type:depth-idx)	Output Shape	Param #
├─MelLikeEncoder: 1-1	[-1, 1, 128, 79]	--
└─Conv1d: 2-1	[-1, 128, 79]	131,200
├─UNet: 1-2	[-1, 2, 128, 80]	--
└─ModuleList: 2	[]	--
└─DownsampleBlock: 3-1	[-1, 64, 64, 40]	--
└─ConvStack: 4-1	[-1, 64, 128, 80]	--
└─Sequential: 5-1	[-1, 64, 128, 80]	--
└─Conv2d: 6-1	[-1, 64, 128, 80]	640
└─BatchNorm2d: 6-2	[-1, 64, 128, 80]	128
└─ReLU: 6-3	[-1, 64, 128, 80]	--
└─Conv2d: 6-4	[-1, 64, 128, 80]	36,928
└─BatchNorm2d: 6-5	[-1, 64, 128, 80]	128
└─ReLU: 6-6	[-1, 64, 128, 80]	--
└─MaxPool2d: 4-2	[-1, 64, 64, 40]	--
└─DownsampleBlock: 3-2	[-1, 128, 32, 20]	--
└─ConvStack: 4-3	[-1, 128, 64, 40]	--
└─Sequential: 5-2	[-1, 128, 64, 40]	--

					└─Conv2d: 6-7	[-1, 128, 64, 40]	73,856
					└─BatchNorm2d: 6-8	[-1, 128, 64, 40]	256
					└─ReLU: 6-9	[-1, 128, 64, 40]	--
					└─Conv2d: 6-10	[-1, 128, 64, 40]	147,584
					└─BatchNorm2d: 6-11	[-1, 128, 64, 40]	256
					└─ReLU: 6-12	[-1, 128, 64, 40]	--
			└─MaxPool2d: 4-4			[-1, 128, 32, 20]	--
		└─DownsampleBlock: 3-3				[-1, 256, 16, 10]	--
			└─ConvStack: 4-5			[-1, 256, 32, 20]	--
				└─Sequential: 5-3		[-1, 256, 32, 20]	--
					└─Conv2d: 6-13	[-1, 256, 32, 20]	295,168
					└─BatchNorm2d: 6-14	[-1, 256, 32, 20]	512
					└─ReLU: 6-15	[-1, 256, 32, 20]	--
					└─Conv2d: 6-16	[-1, 256, 32, 20]	590,080
					└─BatchNorm2d: 6-17	[-1, 256, 32, 20]	512
					└─ReLU: 6-18	[-1, 256, 32, 20]	--
			└─MaxPool2d: 4-6			[-1, 256, 16, 10]	--
		└─DownsampleBlock: 3-4				[-1, 512, 8, 5]	--
			└─ConvStack: 4-7			[-1, 512, 16, 10]	--
				└─Sequential: 5-4		[-1, 512, 16, 10]	--
					└─Conv2d: 6-19	[-1, 512, 16, 10]	1,180,160
					└─BatchNorm2d: 6-20	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-21	[-1, 512, 16, 10]	--
					└─Conv2d: 6-22	[-1, 512, 16, 10]	2,359,808
					└─BatchNorm2d: 6-23	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-24	[-1, 512, 16, 10]	--
			└─MaxPool2d: 4-8			[-1, 512, 8, 5]	--
	└─Sequential: 2-2					[-1, 512, 8, 5]	--
		└─AxialAttentionBlock: 3-5				[-1, 512, 8, 5]	--
			└─AxialAttention: 4-9			[-1, 512, 8, 5]	--
				└─MultiheadAttention: 5-5		[-1, 8, 512]	--
				└─MultiheadAttention: 5-6		[-1, 5, 512]	--
			└─BatchNorm2d: 4-10			[-1, 512, 8, 5]	1,024
			└─ReLU: 4-11			[-1, 512, 8, 5]	--

			└─Conv2d: 4-12	[-1, 512, 8, 5]	262,656
		└─AxialAttentionBlock: 3-6		[-1, 512, 8, 5]	--
			└─AxialAttention: 4-13	[-1, 512, 8, 5]	--
				└─MultiheadAttention: 5-7	[-1, 8, 512]	--
				└─MultiheadAttention: 5-8	[-1, 5, 512]	--
			└─BatchNorm2d: 4-14	[-1, 512, 8, 5]	1,024
			└─ReLU: 4-15	[-1, 512, 8, 5]	--
			└─Conv2d: 4-16	[-1, 512, 8, 5]	262,656
	└─ModuleList: 2			[]	--
		└─UpsampleBlock: 3-6		[-1, 512, 16, 10]	--
			└─Upsample: 4-15	[-1, 512, 16, 10]	--
			└─ConvStack: 4-16	[-1, 512, 16, 10]	--
				└─Sequential: 5-5	[-1, 512, 16, 10]	--
					└─Conv2d: 6-25	[-1, 512, 16, 10]	4,719,104
					└─BatchNorm2d: 6-26	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-27	[-1, 512, 16, 10]	--
					└─Conv2d: 6-28	[-1, 512, 16, 10]	2,359,808
					└─BatchNorm2d: 6-29	[-1, 512, 16, 10]	1,024
					└─ReLU: 6-30	[-1, 512, 16, 10]	--
		└─UpsampleBlock: 3-7		[-1, 256, 32, 20]	--
			└─Upsample: 4-17	[-1, 512, 32, 20]	--
			└─ConvStack: 4-18	[-1, 256, 32, 20]	--
				└─Sequential: 5-6	[-1, 256, 32, 20]	--
					└─Conv2d: 6-31	[-1, 256, 32, 20]	1,769,728
					└─BatchNorm2d: 6-32	[-1, 256, 32, 20]	512
					└─ReLU: 6-33	[-1, 256, 32, 20]	--
					└─Conv2d: 6-34	[-1, 256, 32, 20]	590,080
					└─BatchNorm2d: 6-35	[-1, 256, 32, 20]	512
					└─ReLU: 6-36	[-1, 256, 32, 20]	--
		└─UpsampleBlock: 3-8		[-1, 128, 64, 40]	--
			└─Upsample: 4-19	[-1, 256, 64, 40]	--
			└─ConvStack: 4-20	[-1, 128, 64, 40]	--
				└─Sequential: 5-7	[-1, 128, 64, 40]	--
					└─Conv2d: 6-37	[-1, 128, 64, 40]	442,496

									└─BatchNorm2d: 6-38	[-1, 128, 64, 40]	256
									└─ReLU: 6-39	[-1, 128, 64, 40]	--
									└─Conv2d: 6-40	[-1, 128, 64, 40]	147,584
									└─BatchNorm2d: 6-41	[-1, 128, 64, 40]	256
									└─ReLU: 6-42	[-1, 128, 64, 40]	--
								└─UpsampleBlock: 3-9		[-1, 64, 128, 80]	--
									└─Upsample: 4-21	[-1, 128, 128, 80]	--
									└─ConvStack: 4-22	[-1, 64, 128, 80]	--
									└─Sequential: 5-8	[-1, 64, 128, 80]	--
									└─Conv2d: 6-43	[-1, 64, 128, 80]	110,656
									└─BatchNorm2d: 6-44	[-1, 64, 128, 80]	128
									└─ReLU: 6-45	[-1, 64, 128, 80]	--
									└─Conv2d: 6-46	[-1, 64, 128, 80]	36,928
									└─BatchNorm2d: 6-47	[-1, 64, 128, 80]	128
									└─ReLU: 6-48	[-1, 64, 128, 80]	--
								└─Conv2d: 2-3		[-1, 2, 128, 80]	130
							├─MelLikeDecoder: 1-3			[-1, 1, 40192]	--
								└─ConvTranspose1d: 2-4		[-1, 1, 40192]	131,073

Supplementary materials for Chapter 4

As with Chapter 3, we provide a trace of our MelSpecVAE model in Table 7.3. An example configuration for running it can be found in our Github repository.

Table 7.3: MelSpecVAE Architecture

Layer (type:depth-idx)	Output Shape	Param #
├─DoubleConv: 1-1	[-1, 32, 128, 64]	--
│ └─Sequential: 2-1	[-1, 32, 128, 64]	--
│ │ └─Conv2d: 3-1	[-1, 32, 128, 64]	320
│ │ └─BatchNorm2d: 3-2	[-1, 32, 128, 64]	64
│ │ └─ReLU: 3-3	[-1, 32, 128, 64]	--
│ │ └─Conv2d: 3-4	[-1, 32, 128, 64]	9,248
│ │ └─BatchNorm2d: 3-5	[-1, 32, 128, 64]	64
│ │ └─ReLU: 3-6	[-1, 32, 128, 64]	--
├─Down: 1-2	[-1, 64, 64, 32]	--
│ └─Sequential: 2-2	[-1, 64, 64, 32]	--
│ │ └─MaxPool2d: 3-7	[-1, 32, 64, 32]	--
│ │ └─DoubleConv: 3-8	[-1, 64, 64, 32]	--
│ │ │ └─Sequential: 4-1	[-1, 64, 64, 32]	--
│ │ │ │ └─Conv2d: 5-1	[-1, 64, 64, 32]	18,496
│ │ │ │ └─BatchNorm2d: 5-2	[-1, 64, 64, 32]	128
│ │ │ │ └─ReLU: 5-3	[-1, 64, 64, 32]	--
│ │ │ │ └─Conv2d: 5-4	[-1, 64, 64, 32]	36,928
│ │ │ │ └─BatchNorm2d: 5-5	[-1, 64, 64, 32]	128
│ │ │ │ └─ReLU: 5-6	[-1, 64, 64, 32]	--
├─Down: 1-3	[-1, 128, 32, 16]	--
│ └─Sequential: 2-3	[-1, 128, 32, 16]	--
│ │ └─MaxPool2d: 3-9	[-1, 64, 32, 16]	--
│ │ └─DoubleConv: 3-10	[-1, 128, 32, 16]	--
│ │ │ └─Sequential: 4-2	[-1, 128, 32, 16]	--
│ │ │ │ └─Conv2d: 5-7	[-1, 128, 32, 16]	73,856
│ │ │ │ └─BatchNorm2d: 5-8	[-1, 128, 32, 16]	256

| | | | | └─ReLU: 5-9 [-1, 128, 32, 16] --
| | | | | └─Conv2d: 5-10 [-1, 128, 32, 16] 147,584
| | | | | └─BatchNorm2d: 5-11 [-1, 128, 32, 16] 256
| | | | | └─ReLU: 5-12 [-1, 128, 32, 16] --
├─Down: 1-4 [-1, 256, 16, 8] --
| └─Sequential: 2-4 [-1, 256, 16, 8] --
| | └─MaxPool2d: 3-11 [-1, 128, 16, 8] --
| | └─DoubleConv: 3-12 [-1, 256, 16, 8] --
| | | └─Sequential: 4-3 [-1, 256, 16, 8] --
| | | | └─Conv2d: 5-13 [-1, 256, 16, 8] 295,168
| | | | └─BatchNorm2d: 5-14 [-1, 256, 16, 8] 512
| | | | └─ReLU: 5-15 [-1, 256, 16, 8] --
| | | | └─Conv2d: 5-16 [-1, 256, 16, 8] 590,080
| | | | └─BatchNorm2d: 5-17 [-1, 256, 16, 8] 512
| | | | └─ReLU: 5-18 [-1, 256, 16, 8] --
├─Down: 1-5 [-1, 256, 8, 4] --
| └─Sequential: 2-5 [-1, 256, 8, 4] --
| | └─MaxPool2d: 3-13 [-1, 256, 8, 4] --
| | └─DoubleConv: 3-14 [-1, 256, 8, 4] --
| | | └─Sequential: 4-4 [-1, 256, 8, 4] --
| | | | └─Conv2d: 5-19 [-1, 256, 8, 4] 590,080
| | | | └─BatchNorm2d: 5-20 [-1, 256, 8, 4] 512
| | | | └─ReLU: 5-21 [-1, 256, 8, 4] --
| | | | └─Conv2d: 5-22 [-1, 256, 8, 4] 590,080
| | | | └─BatchNorm2d: 5-23 [-1, 256, 8, 4] 512
| | | | └─ReLU: 5-24 [-1, 256, 8, 4] --
├─Flatten: 1-6 [-1, 8192] --
├─Linear: 1-7 [-1, 128] 1,048,704
├─Linear: 1-8 [-1, 128] 1,048,704
├─Linear: 1-9 [-1, 10] 1,290
├─Linear: 1-10 [-1, 8192] 1,056,768
├─Up: 1-11 [-1, 256, 16, 8] --
| └─Upsample: 2-6 [-1, 256, 16, 8] --
| └─DoubleConv: 2-7 [-1, 256, 16, 8] --

		└─Sequential: 3-15	[-1, 256, 16, 8]	--
			└─Conv2d: 4-5	[-1, 256, 16, 8]	590,080
			└─BatchNorm2d: 4-6	[-1, 256, 16, 8]	512
			└─ReLU: 4-7	[-1, 256, 16, 8]	--
			└─Conv2d: 4-8	[-1, 256, 16, 8]	590,080
			└─BatchNorm2d: 4-9	[-1, 256, 16, 8]	512
			└─ReLU: 4-10	[-1, 256, 16, 8]	--
├─Up: 1-12				[-1, 128, 32, 16]	--
		└─Upsample: 2-8		[-1, 256, 32, 16]	--
		└─DoubleConv: 2-9		[-1, 128, 32, 16]	--
		└─Sequential: 3-16		[-1, 128, 32, 16]	--
			└─Conv2d: 4-11	[-1, 128, 32, 16]	295,040
			└─BatchNorm2d: 4-12	[-1, 128, 32, 16]	256
			└─ReLU: 4-13	[-1, 128, 32, 16]	--
			└─Conv2d: 4-14	[-1, 128, 32, 16]	147,584
			└─BatchNorm2d: 4-15	[-1, 128, 32, 16]	256
			└─ReLU: 4-16	[-1, 128, 32, 16]	--
├─Up: 1-13				[-1, 64, 64, 32]	--
		└─Upsample: 2-10		[-1, 128, 64, 32]	--
		└─DoubleConv: 2-11		[-1, 64, 64, 32]	--
		└─Sequential: 3-17		[-1, 64, 64, 32]	--
			└─Conv2d: 4-17	[-1, 64, 64, 32]	73,792
			└─BatchNorm2d: 4-18	[-1, 64, 64, 32]	128
			└─ReLU: 4-19	[-1, 64, 64, 32]	--
			└─Conv2d: 4-20	[-1, 64, 64, 32]	36,928
			└─BatchNorm2d: 4-21	[-1, 64, 64, 32]	128
			└─ReLU: 4-22	[-1, 64, 64, 32]	--
├─Up: 1-14				[-1, 32, 128, 64]	--
		└─Upsample: 2-12		[-1, 64, 128, 64]	--
		└─DoubleConv: 2-13		[-1, 32, 128, 64]	--
		└─Sequential: 3-18		[-1, 32, 128, 64]	--
			└─Conv2d: 4-23	[-1, 32, 128, 64]	18,464
			└─BatchNorm2d: 4-24	[-1, 32, 128, 64]	64
			└─ReLU: 4-25	[-1, 32, 128, 64]	--

			└─Conv2d: 4-26	[-1, 32, 128, 64]	9,248
			└─BatchNorm2d: 4-27	[-1, 32, 128, 64]	64
			└─ReLU: 4-28	[-1, 32, 128, 64]	--
	├─OutConv: 1-15			[-1, 1, 128, 64]	--
			└─Conv2d: 2-14	[-1, 1, 128, 64]	33

Supplementary materials for Chapter 5

An overview of the DCGAN and StyleGAN based generative audio model architectures that we tested the ACRP infinite GANs algorithm on can be found below.

SpecGAN (DCGAN)

SpecGAN, which is itself a spectrogram-based modification of WaveGAN, was one of the first audio based GANs to effectively synthesise intelligible raw audio from latent vectors and are often used as baselines to compare more modern architectures against. They formulated a spectrogram-representation allowing for approximate inversion after synthesis with a network based on an image based 2D convolutional DCGAN architecture. The invertible format is important for converting this image representation back to perceivable audio without lossy estimation or training a network specifically for this inversion.

Both the generator and discriminator networks are fully convolutional architectures of equal depth. The generator consists of a fully connected layer to project the conditioned latent noise to 3-dimensional space of size $[1024, 4, 4]$. This then passes through 4 transposed 2D convolutional layers that gradually upsample the feature maps by halving the channel dimension and doubling both the frequency and temporal dimensions, such that the projected noise vector is transformed from $[1024, 4, 4] \rightarrow [64, 64, 64]$. Following every convolutional layer is a batch norm layer, finishing with a ReLU activation before being passed into the next upsampling convolution. The output block consists of the final transposed convolutional layer and a tanh activation, resulting in a batch of synthesised mel-spectrograms of size $[1, 128, 128]$. The discriminator is an inversion of this architecture, which takes either fake or real mel-spectrograms as input and successively downsamples the frequency and temporal components whilst increasing the number of channels. Rather than transposed convolutions, the discriminator uses Conv2D layers for this upsampling, transforming the input from $[1, 128, 128] \rightarrow [1024, 4, 4]$. No batch norm is applied in the first upsampling operation, but for all following layers batch normalisation is used. The output of each upsampling block is passed into LeakyReLU activation layers. Following the upsampling process, the output is then flattened and used as input for a final linear layer, which during initialisation computes the probability the input is real or fake. During the CRP stage, the initial estimation for the number of modes in the

data is taken as the output dimension in the linear layer, calculating the probability a sample is real with respect to each cluster, taking the highest activation out of all modes.

ASGAN (StyleGAN)

Audio StyleGAN (ASGAN) is a deep learning model based heavily on the original StyleGAN literature, but modified for unconditional audio synthesis instead of images, similar to SpecGAN vs the original DCGAN. StyleGAN is a more modern and far more powerful GAN in comparison to DCGAN, utilizing a much deeper architecture, and entirely new additions such as a mapping network for the latent vector used in synthesis, and "style blocks" used to control specific visual features of the generated sample (e.g. colour, shape etc.) at each layer of the network. They also inject random noise into each style block for stochastic variation in synthesised samples. Essentially this only affects details that don't drastically change the overall appearance of the image (coarse features) but gives far greater control over the finer features. ASGAN employs all of these additions to create a powerful model capable of not only producing high quality audio samples, but using styles gives direct control over the coarse features (i.e. the word being spoken) and the finer features (i.e. the tone and speaker saying the word).

As with SpecGAN, first a latent noise z sampled from a normal distribution is combined with a conditional embedding depending on the GANs being sampled. However, the ASGAN generator consists of mapping network W, consisting of linear and LeakyReLU activation layers to map z into a latent linearly disentangled vector w of the same dimensionality (i.e. z of size 512 becomes w of size 512). Once the latent vector has been mapped to w, it is passed into a layer new to the original StyleGAN named a Fourier feature (FF) layer to convert the latent vector into a sequence of cosine features of fixed length, which helps the network to better learn to generate high-frequency components in the data (Tancik et al., 2020) that store fine-grained detail in the audio. This produces a sequence of vectors to be passed into a sequence of style blocks consisting of a modulated convolution layer, an upsampling step, a LeakyReLU activation layer surrounded by low-pass filter (LPF) layers, and finally a downsampling step. The generator contains a total of 5 style blocks at a dimensionality of 1024, 4 style blocks of size 512, 3 of size 256, 2 of size 128, then into a final modulated convolutional layer. In the final style block of each stack, the output is upsampled by a factor of 4, leading to the target size of $[128, 128, 1]$ for synthesised mel-spectrogram data; the process is also very similar for training on discrete HuBERT (Hsu et al., 2021) audio features extracted from a pretrained HuBERT model. Each block is also supplemented with the style vector taken from latent mapping w for fine-tuned control and variation of speech features, as in the original StyleGAN paper (Karras et al., 2019).

ASGAN's discriminator consists of 4 ConvD blocks and a network head to predict whether the input is real or not. The ConvD blocks contain several 1D convolutional layers using LeakyReLU

activations, a downsampling layer and a final skip connection with the addition of an anti-aliasing LPF layer. To keep the discriminator balanced with the generator, the number of layers and blocks are selected such that the discriminator contains approximately the same number of parameters. In the original paper, to convert these feature maps into logits the output of the ConvD blocks is passed into the network head beginning with a minibatch standard deviation layer, then into a 1D convolutional layer whose output is flattened and finally passed through 2 LeakyReLU activation and linear layers.

Comparative Analysis

The key benefit of SpecGAN is the much faster training time as a result of its simpler architecture. This is a massive benefit due to the high dimensionality of audio data leading to much longer training sessions compared with most comparable image tasks which uses datasets with samples of much smaller size. ASGAN's main advantage is the quality and diversity of the samples it can synthesize due to its much more complex architecture. For our purposes not all of the benefits offered by StyleGAN based models over DCGAN based models are necessary, such as stochastic variation and control over specific features using styles, but strong generative results in the initialisation stage of training are important for the later CRP stage to work effectively.

This means ASGAN takes much longer to train and SpecGAN produces lower quality synthesised audio, leading to a trade-off between performance and speed. This could be mitigated to an extent by either adding more complexity or depth to the SpecGAN architecture or reducing the size of the data through downsampling or other augmentation of the audio data, but as a test of the framework these two GANs with drastically different strengths and limitations show how different networks influence the performance of the CRP process. By making only small necessary modifications to these models, we can test how any generic GAN or similar generative model could benefit from training with infinite conditions for automatically capturing and separating modes within the dataset for diverse generation.

SpecGAN and Encoder Model Traces

We provide a trace of our implementation of the SpecGAN model's generator and discriminator in Table 7.4 and Table 7.5, and a trace of our encoder module in Table 7.6. A configuration file for training it with the ACRP process is provided in our Github repository.

Table 7.4: SpecGAN Generator Architecture

Layer (type:depth-idx)	Output Shape	Param #
Generator	[64, 1, 128, 128]	--
├─LatentEmbeddingAdd: 1-1	[64, 128]	--
│ └─Embedding: 2-1	[64, 128]	1,920
├─Linear: 1-2	[64, 16384]	2,113,536
├─ConvTranspose2d: 1-3	[64, 512, 8, 8]	8,388,608
├─BatchNorm2d: 1-4	[64, 512, 8, 8]	1,024
├─ConvTranspose2d: 1-5	[64, 256, 16, 16]	2,097,152
├─BatchNorm2d: 1-6	[64, 256, 16, 16]	512
├─ConvTranspose2d: 1-7	[64, 128, 32, 32]	524,288
├─BatchNorm2d: 1-8	[64, 128, 32, 32]	256
├─ConvTranspose2d: 1-9	[64, 64, 64, 64]	131,072
├─BatchNorm2d: 1-10	[64, 64, 64, 64]	128
├─Sequential: 1-11	[64, 1, 128, 128]	--
│ └─ConvTranspose2d: 2-2	[64, 1, 128, 128]	1,024
│ └─Tanh: 2-3	[64, 1, 128, 128]	--

Table 7.5: SpecGAN Discriminator Architecture

Layer (type:depth-idx)	Output Shape	Param #
Discriminator	[64]	16,385
├─Sequential: 1-1	[64, 64, 64, 64]	--
│ └─Conv2d: 2-1	[64, 64, 64, 64]	1,024
│ └─LeakyReLU: 2-2	[64, 64, 64, 64]	--
├─Sequential: 1-2	[64, 128, 32, 32]	--
│ └─Conv2d: 2-3	[64, 128, 32, 32]	131,072
│ └─BatchNorm2d: 2-4	[64, 128, 32, 32]	256
│ └─LeakyReLU: 2-5	[64, 128, 32, 32]	--
├─Sequential: 1-3	[64, 256, 16, 16]	--
│ └─Conv2d: 2-6	[64, 256, 16, 16]	524,288
│ └─BatchNorm2d: 2-7	[64, 256, 16, 16]	512
│ └─LeakyReLU: 2-8	[64, 256, 16, 16]	--
├─Sequential: 1-4	[64, 512, 8, 8]	--
│ └─Conv2d: 2-9	[64, 512, 8, 8]	2,097,152
│ └─BatchNorm2d: 2-10	[64, 512, 8, 8]	1,024
│ └─LeakyReLU: 2-11	[64, 512, 8, 8]	--
├─Sequential: 1-5	[64, 1024, 4, 4]	--
│ └─Conv2d: 2-12	[64, 1024, 4, 4]	8,388,608
│ └─BatchNorm2d: 2-13	[64, 1024, 4, 4]	2,048
│ └─LeakyReLU: 2-14	[64, 1024, 4, 4]	--
├─LinearConditionalMaskLogits: 1-6	[64]	--
│ └─Linear: 2-15	[64, 15]	245,775

Table 7.6: Encoder Architecture

Layer (type:depth-idx)	Output Shape	Param #
Encoder	[64, 15]	--
├─Sequential: 1-1	[64, 64, 64, 64]	--
│ └─Conv2d: 2-1	[64, 64, 64, 64]	1,088
│ └─LeakyReLU: 2-2	[64, 64, 64, 64]	--
├─Sequential: 1-2	[64, 128, 32, 32]	--
│ └─Conv2d: 2-3	[64, 128, 32, 32]	131,200
│ └─LeakyReLU: 2-4	[64, 128, 32, 32]	--
├─Sequential: 1-3	[64, 256, 16, 16]	--
│ └─Conv2d: 2-5	[64, 256, 16, 16]	524,544
│ └─LeakyReLU: 2-6	[64, 256, 16, 16]	--
├─Sequential: 1-4	[64, 512, 8, 8]	--
│ └─Conv2d: 2-7	[64, 512, 8, 8]	2,097,664
│ └─LeakyReLU: 2-8	[64, 512, 8, 8]	--
├─Sequential: 1-5	[64, 1024, 4, 4]	--
│ └─Conv2d: 2-9	[64, 1024, 4, 4]	8,389,632
│ └─LeakyReLU: 2-10	[64, 1024, 4, 4]	--
├─Linear: 1-6	[64, 15]	245,775