Real-time Audio Variational Autoencoders

Chunarkar Sumeeth Kumar
Indian Institute of Technology, Hyderabad
ai21btech11008@iith.ac.in

Gujjula Samarasimha Reddy
Indian Institute of Technology, Hyderabad
ai23mtech02001@iith.ac.in

Velma Dhatri Reddy
Indian Institute of Technology, Hyderabad
ai21btech11030@iith.ac.in

Nani Meka
Indian Institute of Technology, Hyderabad
ee23resch01001@iith.ac.in

Sumohana Channappayya
Indian Institute of Technology, Hyderabad
sumohana@ee.iith.ac.in

Abstract

Deep learning models like GANs, VAEs, and flow-based generative models have achieved remarkable success in estimating data distributions. However, applying them to raw waveform modeling, especially in music, is challenging due to computational intensity and reliance on low sampling rates. This project introduces RAVE (Realtime Audio Variational autoEncoder), a specialized variational autoencoder for fast, high-quality neural audio synthesis. RAVE employs a two-stage training process, commencing with multiband decomposition followed by Representation learning and adversarial fine-tuning. This methodology enables RAVE to decode audio, ultimately allowing for the generation of 48kHz audio signals with superior quality.

1. Introduction

Supervised learning is a form of machine learning utilized to train algorithms using labeled data. The resulting representations learned by the algorithms are not highly robust and generalized. This is primarily due to the manual and task-specific nature of the data labeling process. To address these challenges, we are transitioning towards employing unsupervised learning algorithms.

In unsupervised learning, we train the model using labels that naturally exist within the input data. There are several techniques to learn these representations. In this project, we will focus on one of the most common strategies for unsupervised learning, which involves reconstructing or generating the data from a corrupted view.

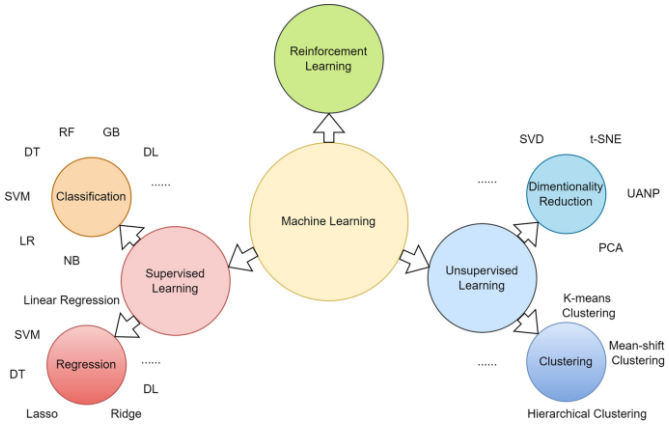


Figure 1. Classification of machine learning algorithms

2. Latent space

Suppose x belongs to $\mathbb{R}^n$ and is an observed data point generated with noise, either linearly or non-linearly, from some lower dimension z belonging to $\mathbb{R}^m$ $(m < n)$. In general, what we observe is highly dimensional and noisy, with a very high correlation between data points. Hence, we aim to estimate lower dimensional compact and compressed points. As these points are not directly accessible, they are termed latent or hidden variables, and the space spanned by latent vectors is the latent space.

In the linear transformation $X = AZ + v$, a classical unsupervised algorithm called **PCA** is employed to discover the underlying hidden representation z. However, it may fail if the hidden representation has a complex relationship with the raw data. In such cases, a deep autoencoder is utilized.

2.1. Autoencoders

Autoencoders are unsupervised neural networks comprising two main components. Firstly, there is an encoder that takes an input x and efficiently learns to compress and encode the data into a structure known as a latent vector. Subsequently, the decoder is trained to reconstruct data representations from the encoded data, aiming for the output $\hat{x}$ to closely resemble the original input data.

In essence, autoencoders effectively learn to recognize the relevant aspects of observed raw data and minimize the noise in the data that can be discarded. The objective function used is simply MSE $= \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$.

2.2. Auto encoders as PCA

If there exists only a single hidden layer without any non-linear activation function, this model is somewhat akin to PCA. When applying PCA, let's assume we have a point x belonging to $\mathbb{R}^n$. We then apply a transformation $y = A^T \times x$ to obtain the point z belonging to $\mathbb{R}^m$ $(m < n)$, and subsequently transform z to $\hat{x}$ by applying a linear transformation $\hat{x} = A \times z$. This process is essentially similar to what occurs in PCA, and the objective function is given by

$$\text{minimize}\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

$$\text{minimize}\frac{1}{n}\sum_{i=1}^{n}(x_i - A^T A x_i)^2$$

with the exception that in PCA, we enforce an orthonormal constraint $A^T A = I$. Hence, in a sense, it is not entirely identical; it will not precisely capture the same basis as PCA, but it spans the same space.

2.3. Deep Autoencoders

Hence, by adding more layers with a non-linear activation function, it evolves into a deep autoencoder as depicted in Figure 2, serving to capture non-linearly generated data.

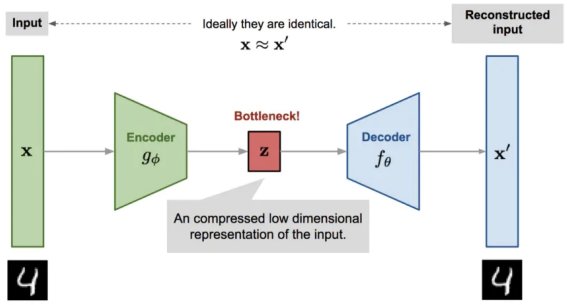


Figure 2. Deep autoencoder

The encoder network g_ϕ takes x as input and learns to obtain a compact point representation z. Conversely, the decoder network f_θ takes z as the latent representation and masters the art of reconstructing the input data. If we train on the entire dataset, the latent representation may resemble the one shown in Figure 3 (2D-dimensional).

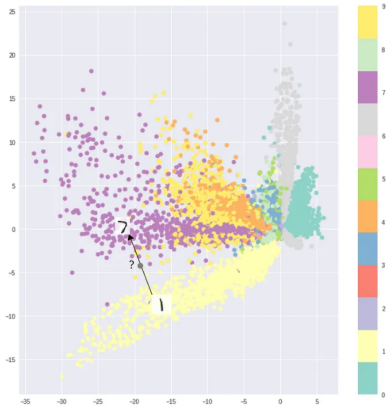


Figure 3. Autoencoder Latent space

Since there are no restrictions on the latent variable z, the values it can take are continuous. Consequently, the latent space appears disorderly. Why is this a concern? During training, the encoder acts as inference; it deduces a value for z that allows us to reconstruct the original input from the encoder representation. Conversely, the decoder acts as a generator; it learns to generate the meaningful whole output. Consequently, after training on the complete dataset, isolating encoder and decoder networks by sampling points from the latent space will yield a meaningful overall outcome. That is acceptable if it is learned for a particular latent point during training. However, what happens if a point sampled from the latent space, as illustrated in Figure:3, generates an output that is unrealistic and does not belong to the training dataset. In summary, we face two limitations: the latent space is disordered, and the volume of the latent space is not compact.

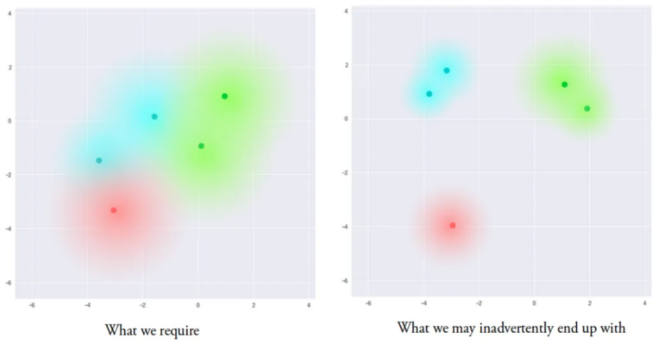


Figure 4

2.4. Variational Autoencoders

Ideally, the latent space should organize semantically similar data points close to each other and separate dissimilar points, encapsulating a compact volume. Let us delve into how Variational Autoencoders (VAEs) [2] address these constraints. Let x be the input data and z its latent representation. When perturbed by a small value, the decoder needs to generate data points closer to x. Instead of point estimation, as done in autoencoders, here we model a distribution for each latent point. This ensures that points neighboring the latent vector represent semantically similar data points. Upon enforcing this condition, the latent space takes the appearance shown in Figure 4. However, the volume of the latent space remains non-compact. To address this, we transition from modeling each latent vector as a distinct Gaussian to imposing an identical distribution on all latent points. This adjustment leads to the latent space depicted in Figure 5 (middle).

Figure 5

2.5. Mathematical Analysis of VAE's

The objective of the auto encoders is to model the data distribution to generate a new sample that likely follows original data distribution .The posterior distribution of the latent variable requires an increasing computational complexity with the increasing dimension of the latent variable, so instead of modelling the multi modal posterior distribution we use an proxy to estimate. To avoid the computational complexity choose an family of functions belonging to the exponential, likely to be Gaussian family which are computationally tractable.

$$Proxy - posterior - distribution : q(z/x)$$

$$Intractable - posterior - distribution : p(z/x)$$

$$\min[D(q(z/x)//p(z/x))] = \sum q(z/x) \log_2 q(z/x)/p(z/x)$$

$$\min[D(q(z/x)//p(z/x))] = \sum q(z/x) \log_2 p(x,z)/q(z/x) + \log_2 p(x)$$

$$\log_2 p(x) = \min[D(q(z/x)//p(z/x))] + \sum q(z/x) \log_2 p(x,z)/q(z/x)$$

left hand side is a constant, where as in right hand side one term is minimized, so to balance the equation other term must be maximized. After solving the equation for evidence or marginal lower bound, the objective is turned as follows

$$L = \mathbb{E}_{q(z/x)}[\log_2 p(x/z)] - D(q(z/x)//p(z))$$

3. WAVENET: A GENERATIVE MODEL FOR RAW AUDIO

WaveNet [3] is a deep generative model for audio data, developed by DeepMind in 2016. It's known for its ability to generate highly realistic and natural-sounding speech and music. WaveNet is a type of neural network called a convolutional neural network (CNN) that's specifically designed for modeling audio waveforms. What sets it apart from other audio generation models is its autoregressive approach, where it generates one audio sample at a time, conditioned on the previous samples. This technique allows it to capture complex dependencies in the data, resulting in superior audio quality.

WaveNet's key innovation lies in its use of dilated convolutions, which enable the model to have a wide receptive field without the computational burden of traditional convolutions. This architecture allows it to generate audio at a high level of detail and realism. WaveNet was initially used for text-to-speech (TTS) applications, revolutionizing the quality of synthesized voices. Over time, it has found applications in various fields, including music generation, audio synthesis, and even the improvement of low-quality audio recordings.

Despite its impressive capabilities, WaveNet is computationally intensive and often requires significant resources to generate audio in real-time. Researchers have developed more efficient variants, such as WaveGlow and Parallel WaveGAN, to make real-time audio generation more feasible. These adaptations maintain the high quality of WaveNet-generated audio while significantly reducing the computational requirements, making the technology more accessible for various applications, from virtual assistants to creative audio production. WaveNet's contributions have had a profound impact on the field of deep learning for audio generation and have paved the way for numerous applications that benefit from realistic audio synthesis.

3.1. A variational autoencoder for fast and high-quality neural audio synthesis

In the realm of audio synthesis, the integration of deep learning methods has brought forth innovative applications

such as unconditional audio generation and timbre transfer between instruments. However, these approaches have been hampered by their substantial computational demands, primarily stemming from the extensive temporal dimensionality of audio data. To alleviate these computational constraints, many prior endeavors have resorted to low sampling rates, typically within the range of 16kHz to 20kHz. Although these rates prove adequate for speech applications, they often result in diminished audio quality, particularly in musical contexts. Moreover, the inherent autoregressive sampling nature of these models introduces prohibitive time delays, limiting their real-time applicability.

To address these challenges, one prevalent strategy has been to precompute audio descriptors for conditioning models. While this approach has yielded state-of-the-art performance, it imposes restrictions on the types of signals that can be generated. Variational Autoencoders (VAEs) emerged as a promising alternative, offering a trainable analysis framework without imposing feature constraints. Nevertheless, traditional VAEs encounter issues related to poor reconstruction, primarily due to uninformative latent dimensions, necessitating a decision on the dimensionality of the latent space. In this research endeavor, we explore a

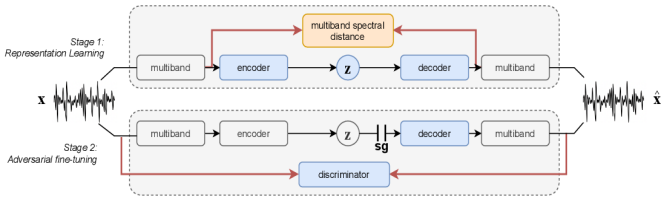


Figure 6. Overall architecture of the proposed approach

specialized VAE known as RAVE [1](Variational Autoencoder for Fast and High-Quality Neural Audio Synthesis). RAVE introduces a unique two-stage training procedure, initially operating as a standard VAE and subsequently fine-tuning with an adversarial generation objective. Furthermore, our approach involves the decomposition of audio into multiple frequency-based bands, enabling the synthesis of audio with sampling rates of up to 48kHz without imposing significant computational overhead. RAVE effectively addresses dimensionality issues in learned representations by employing singular value decomposition to distinguish informative from uninformative latent dimensions. This innovative model holds significant promise in revolutionizing the landscape of neural audio synthesis by offering fast and high-quality audio generation capabilities.

4. Model architecture

4.1. Encoder

The RAVE encoder employs a convolutional neural network featuring leaky ReLU activation and batch normalization (refer to figure 4). In all of our experiments, we consistently employ N = 4, with hidden sizes set at [64, 128, 256, 512] and corresponding strides of [4, 4, 4, 2]. The complete latent space consists of 128 dimensions.

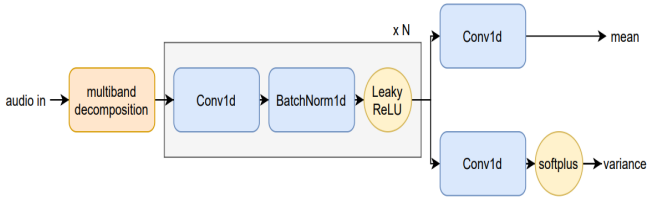


Figure 7. The structural design of the encoder within the RAVE model

4.2. Decoder

An insight into the suggested decoder's functioning. The latent representation undergoes an upsampling process, utilizing alternating upsampling layers and a residual stack. Subsequently, it is processed by three distinct sub-networks, each responsible for generating waveform, loudness envelope, and filtered noise signals.

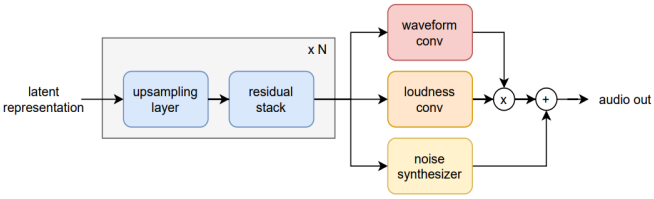


Figure 8

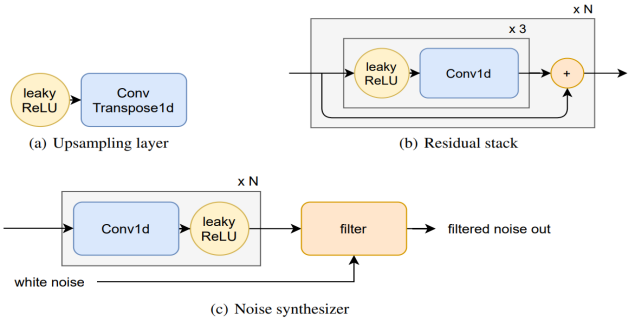


Figure 9. Detailed architecture of the decoder blocks used in the RAVE model

4.2.1 Balancing compactness and fidelity

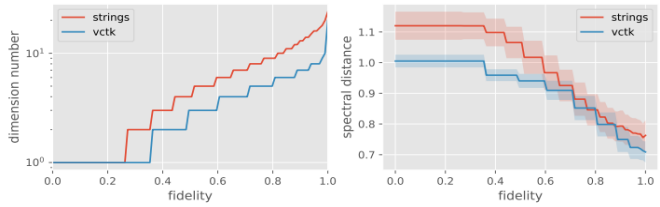
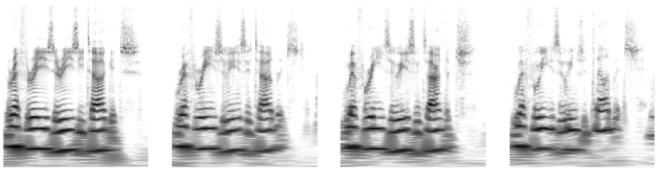


Figure 10. Estimated latent space dimensionality according to the fidelity parameter and its corresponding influence on the reconstruction quality



(a) Original sample (b) $f = 0.99$ (c) $f = 0.90$ (d) $f = 0.80$

Figure 11. Reconstruction of an input sample with several fidelity parameters f

4.3. Limitations of VAE's

Since the latent space is discontinuous and the volume is not compact to generate the new samples, to avoid these issues we imposed Gaussian distribution over the latent representation with a zero mean, unit variance. From the right most Figure[5], the volume is compact but generation quality is decreasing. Effectively for different inputs we are geting closer latent representations, meaning latent representation loosing the condition on the input, which will result in poor reconstruction, since the posterior distribution is falling inside the prior distribution, we called it as $posterior\ collapse$.
Prior distribution imposed on the latent variable is a Gaussian distribution for all latent representations corresponding to the input data points, which will also contradicts, because gaussian is unimodal distribution, in general data follows a multimodal ditribution but tying with same prior to all the distributions may not capture the data dynamics. To overcome these two issuses there are various direction research is going on like $Disentangled\ variational\ autoencoders$, $beta-\ variational\ auto\ encoders$

5. Discrete Latent Space

Auto encoders latent space is continuous, as shown in right side plot in figure[5], the basic idea behind latent representation as discussed in above section to get compact numerical notation of input data. In general the real world data is discrete in nature, to represent the real word discreate data, finite set of numerical representation are enought. For example text is inherently discrete in nature, it can be represented by finite discrete set of text tokens, speech is also represented by discrete set of phonemes, and images are often represented by language. Utilising the model capacity on modelling the important features which span many dimensions by opposing to focusing on noise and imperceptible local properties, these things are to motivated for moving to discrete latent representations.

5.1. Vector Quantized-Variational Autoencoders

The frame work is similar to the variational auto encoders i.e encoder, decoder but in addition to these , code is also learned.The algorithm for VQ-VAE [4] consists of two phases, in first phase encoder,decoder and code parameters updated , in second phase only prior is learned

5.1.1 Training Phase I

In forward pass the input x is given to the encoder, which outputs continuous latent representation $z_e(x)$, Now calculate the euclidean distance for the latent point and code book vectors, code book vectors are randomly initialize which Will be updated in second phase of training, after finding the nearest code book vector store code book vector index in a dense matrix. Now give the quantized latent representations to the decoder it Will reconstruction the output $\hat{x}$ as close to input
In backward pass the derivative on MSE loss= $||x - \hat{x}||^2$ is propagated back to the decoder without any issue but coming to the encoder there exists a discontinuous process to propagate gradient. To overcome this issues, simply copying the gradients directly to encoder, since encoder output and decoder input having the same shape, it may not receive true gradients but it receives approximated gradients so that encoder will update it parameters to minimize the reconstruction error.Since no gradients received to update code book vectors, so it is updated independently by using one simple quantization algorithms, we try to minimize the intra class variance which is similar to k-means algorithm $||z_e - e||_2^2$ Finally, since the volume of the embedding space is dimensionless, it can grow arbitrarily if the embeddings e_i do not train as fast as the encoder parameters. To make sure the encoder commits to an embedding and its output does not grow, we add a commitment loss, the third term in equation 3. Thus, the total training objective becomes: Loss= $||x - \hat{x}||_2^2$+$||sg[z_e] - e||_2^2$+$||z_e - sg[e]||_2^2$

5.1.2 Training phase II

Now fix the encoder, decoder and code-book parameters and the prior over the quantized latent representation. For

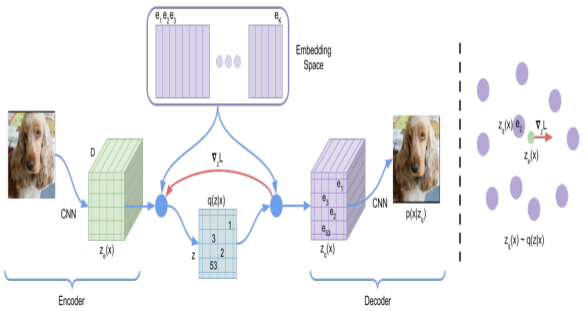


Figure 12. Illustration of model architecture

images Pixel CNN is used to learn the prior in autoregressive manner, similarly for audio case use wavenet model. The main difference here is normal pixel CNN is work in pixel domain but here code book indices are used instead of pixels. Once training of prior is completed now sample the from this autoregressive models to get vey high quality image and vedios.

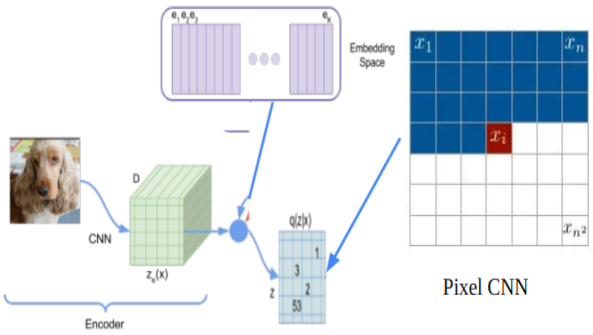


Figure 13. learning prior with Pixel CNN

6. Results

6.1. Images

Images contain a lot of redundant information as most of the pixels are correlated and noisy, therefore learning models at the pixel level could be wasteful

6.2. Audio

7. Conclusion

Variational Autoencoders (VAEs) effctive in generative modeling, by impossing probabilistic latent variables for new sample generation. However, achieving trade-off between reconstruction accuracy and latent space smoothness remains a challenge. Vector Quantized Variational Autoencoders (VQ-VAEs) extend VAEs by introducing discrete

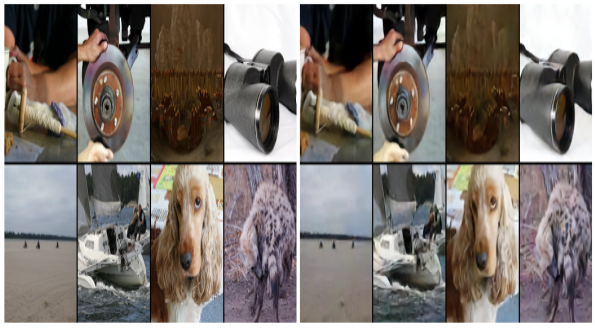


Figure 14. ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with K=512

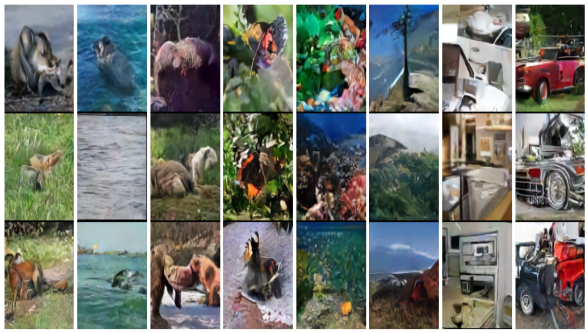


Figure 15. Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images.From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup

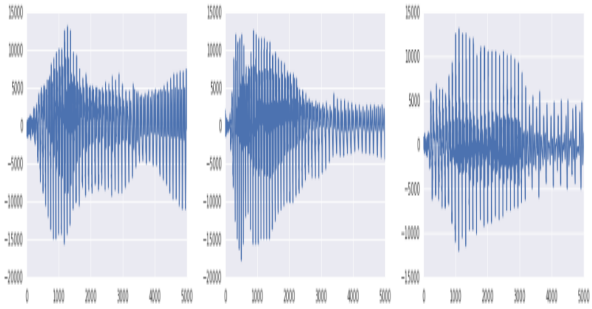


Figure 16. Left: original waveform, middle: reconstructed with same speaker-id, right: reconstructed with different speaker-id. The contents of the three waveforms are the same

latent representations. With a codebooks, VQ-VAEs effectively capture both local and global features, by over coming the issues like posterior collapse. In summary, VQ-VAEs providing structured latent spaces and addressing challenges in discrete representation learning.

References

[1] Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis. *arXiv preprint arXiv:2111.05011*, 2021. 4

[2] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[3] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 3

[4] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 5