Assignment 1 Report

Kiya Aminfar, Sean Steinle

Table of Contents

We split our project into three notebooks to make things as simple to run as possible. Likewise, this report is split into three corresponding sections. The first two notebooks (`notebooks/simple_vae.ipynb`, `notebooks/improving_vae.ipynb`) create a regular VAE and the final notebook (`notebooks/extending_vae.ipynb`) creates a few extensions of the VAE, including a hierarchical VAE and a vector-quantized VAE.

Note: This report is submitted in both Markdown and .pdf. If you review in Markdown, you will be able to play predictions from our models in-line!

The Simple VAE Model

A Key Distinction: Raw Audio vs. Spectrograms

We started off by creating a simple VAE with only a few layers for the encoder and decoder. We also explored the audio modality since neither of us had much experience. From working with the data, we found that there is a key distinction to be made in whether audio is represented as raw audio in a .wav file or whether the audio has been transformed into a spectrogram. You can see this difference qualitatively in our `notebooks/simple_vae.ipynb` notebook where we play the raw audio, a version of the raw audio which was translated into a spectrogram and back, and our predicted spectrogram translated into audio. This distinction is key, because while our first samples were very poor (essentially white noise), this gap in quality between raw audio and transformed audio will act as a ceiling on our model's performance since we are generating spectrogram samples.

Performance and Avenues for Improvement

As mentioned above, our performance was fairly poor on our first run. This is likely due to a few factors: we only trained for 20 epochs, we only used 1000 samples to train, and our model was overly simplistic. Luckily, these limitations also act as avenues to improve our model!

In case you don't want to run the notebook, here is a sample from our first run.

Editorial Note: Looking back, this occurred because we were not doing sufficient audio processing! We were not padding and truncating our audio files and also we did not 'unnormalize' our generated spectrograms before converting them to audio. This resulted in an absolutely wretched 4 second clip for each of our predictions.

An Improved VAE Model

Revisiting the Spectrogram - Audio Dichotomy

The challenges listed above led us to develop a new model in `notebooks/improved_vae.ipynb`, inspired by a Medium blogpost. This model is more modular but more importantly we learned how to handle audio data more adeptly. This is evident in our best sample from this model. I've linked the audio from the ground-truth spectrogram and our generation–can you guess which is which?

Extending VAE

Introduction

The objective of this work was to develop and evaluate generative models for audio synthesis, specifically focusing on Variational Autoencoders (VAE) and their variants, including Hierarchical VAE (HVAE) and Vector Quantized VAE (VQVAE). The goal was to efficiently encode temporal structures into a low-dimensional latent space while preserving the acoustic properties necessary for realistic audio generation. The primary challenge was to ensure that the models captured meaningful latent representations without experiencing posterior collapse or generating incoherent outputs. Additionally, processing long sequences posed computational challenges that required careful architectural and data processing decisions to balance expressiveness, efficiency, and training feasibility.

Modeling Approach

Initially, spectrogram representations were explored alongside raw waveforms to assess their effectiveness in capturing audio features. The spectrogram-based approach involved extracting Mel spectrograms, normalizing them, and mapping them to a latent space for reconstruction. However, the conversion process from audio to spectrogram and back introduced artifacts and loss of information, which affected the quality of the reconstructed signals. To mitigate this, we experimented with different spectrogram resolutions, particularly increasing the number of Mel filters to 256, which yielded better results compared to 128 but was still limited by computational constraints.

For the raw signal approach, the audio waveforms were normalized to prevent numerical instability and segmented into overlapping chunks to manage memory constraints. The encoder extracted temporal features using strided 1D convolutional layers, reducing dimensionality while preserving key signal characteristics. A latent sampling mechanism was implemented with KL divergence regularization to ensure a well-structured latent space. The decoder mirrored the encoder, applying transposed convolutions and learned upsampling strategies to reconstruct the waveform without excessive cropping, which could remove valuable information. The loss function was a combination of mean squared error for reconstruction quality and a KL divergence penalty with -VAE constraints to balance latent space regularization and reconstruction fidelity.

Challenges and Solutions in VAE and HVAE

The spectrogram-based VAE initially trained smoothly, but the reconstruction quality was suboptimal due to the inherent noise in converting between the spectrogram and waveform domains. Increasing the latent dimensionality and adjusting the learning rate provided some improvements, but further refinement of the model size was necessary to capture more complex structures. The HVAE was introduced to extract hierarchical latent features with the hypothesis that a two-level latent space would improve the representation power of the model. However, results were inconsistent, and in some cases, reconstruction performance degraded. This was likely due to difficulties in selecting appropriate hyperparameters to balance information propagation between hierarchical latent spaces. Ensuring that the higher-level latents meaningfully influenced lower-level representations required refining the hierarchical prior, adjusting the KL divergence weighting, and explicitly defining relationships between the two latent levels in the decoder.

For the raw waveform-based VAE, the reconstruction quality was initially poor, leading to modifications in the network architecture. Using global average pooling instead of flattening in the encoder significantly reduced the number of parameters while maintaining critical temporal information. The model also exhibited high variance in latent encodings, making training unstable. Batch normalization was introduced in the latent space to address this issue, leading to improved training stability.

Listen to the best sample from the HVAE here, and the worst sample from the HVAE here.

Challenges and Solutions in VQVAE

The VQVAE posed additional challenges, particularly in codebook utilization and commitment loss tuning. A key issue was mode collapse, where only a limited number of codebook embeddings were used, reducing the model's capacity for diverse generation. This was mitigated by modifying the codebook update mechanism and adding a loss term to encourage a more even distribution of embedding usage. Balancing the commitment loss was also critical because setting it too high caused embeddings to be overly constrained, leading to poor generalization, while setting it too low resulted in unstable training dynamics. To address this, the parameter in the loss function was adjusted dynamically based on the variance of the latent encodings, improving convergence.

Architecturally, discrepancies in output dimensions between the encoder and decoder necessitated restructuring the upsampling path to ensure smooth dimensionality restoration. This adjustment reduced the need for explicit cropping, which could discard essential features. Despite multiple adjustments to network parameters and settings, the VQ loss remained highly unstable, even when reconstruction loss improved. This instability likely stemmed from the discrete nature of vector quantization, where embeddings struggle to converge

due to insufficient gradient flow. Further tuning of the training dynamics and alternative methods for updating the codebook could be explored to address this issue.

Handling Long Sequences and Data Processing

A significant challenge across all models was processing long sequences efficiently. Given memory constraints, full-length waveforms could not be directly used, necessitating chunking strategies to preserve meaningful temporal dependencies. Instead of simple truncation, overlapping segments were extracted to maintain contextual continuity. A sequence length of 22050 samples (corresponding to one second of audio at a 22.05 kHz sample rate) was selected to balance signal fidelity and computational feasibility. Additionally, waveform normalization was applied before training.

Evaluation and Observations

The models were evaluated using both quantitative and qualitative metrics. MSE and cosine similarity provided numerical assessments of reconstruction quality, while subjective listening tests revealed that generations were often dominated by noise. Sampling directly from the latent prior produced incoherent outputs, prompting a shift toward interpolating real encodings to explore the latent space more effectively. The VAE exhibited limited fidelity in preserving fine audio details, while the HVAE provided marginal improvements but suffered from mode averaging, which blurred intricate features. The VQ-VAE showed potential in preserving discrete structures but required additional refinements to ensure better embedding utilization.

Conclusion

This work explored generative audio synthesis using VAE, HVAE, and VQVAE models, each presenting unique challenges in capturing and reconstructing audio features. While the spectrogram-based models offered structured feature extraction, they introduced conversion noise that limited reconstruction quality. The raw waveform models were more challenging to train but provided a more direct representation of the audio signal. The hierarchical and vector quantization approaches introduced complexity in latent space structuring, requiring careful balancing of information flow and loss weighting. Future improvements could involve alternative priors, adversarial training, or attention mechanisms to enhance generation quality and stability.

Appendix

Appendix A: Saving Models

Something I've accidentally spent a lot of time on is trying to save out our VAE model. This would be convenient so that we do not need to retrain our

model from scratch each time we create it. However, saving Keras objects is non-trivial, and must be done carefully. Something I've quickly learned is that it's probably best to keep your model very modular as opposed to monolithic. For example, it would be better to almost treat the encoder and decoder as separate models which can be passed in to assemble the VAE object, rather than trying to serialize the entire VAE object.

Here are some threads I've been looking at for reference: 1. Blog on how to build a VAE in Keras 2. Reddit thread on saving Keras VAE's