VAE Drum Machine

Group 6 - Aparajita Saha ( ), Danny Murphy ( ), Kevin Bretz ( )

Leiden Institute of Advanced Computer Science, The Netherlands

1 Introduction

For our project, we wanted to gather experience in latent space exploration, specifically within the sound or music domain. While there were many projects that we found inspiring, such as neural vocal effects or building AI-based synthesizer modules, we decided to first try our hands on a more straightforward, yet equally exciting experiment: building a VAE-powered drum machine. This project is our attempt at creating a VAE that functions as a traditional drum machine, enabling the generation of completely novel sounds by interpolating through the latent space. This project takes inspiration from the NSynth [2] project.

2 Setup

2.1 Data Description

In this project we use the license free SmapleSwap Drum Dataset, which contains a wide range of various one-shot drum samples. These samples are preprocessed into EnCodec tokens, producing corresponding .pt files stored in the encodec_cache directory. We use these EnCodec files as both the input and output for our VAE to reduce computational complexity and accelerate training while maintaining effectiveness.

2.2 EnCodec [1]

Our EnCodec data is specifically encoded using 16 codebooks per token, generating 150 tokens per second. EnCodec codebooks are hierarchical in nature - they use a residual process, Residual Vector Quantization (RVQ), so the next codebook is determined by the error of the previous codebook. Although it is easy to misinterpret these codebooks as frequency bands, they don't contain specific spectral information. Instead, they can be better regarded as different resolutions that all need to be stacked on top of each other to produce the complete output. Each codebook contains information of a certain level of detail or fidelity.

2.3 Training Objectives

During training, the model must learn to minimize certain objectives.

1. **(Main Objective) Reconstruction Loss** This is calculated as the cross-entropy between the original data and its reconstruction. More specifically, each EnCodec codebook (out of 16 in total), which represents a certain degree of detail in the data, gets its own reconstruction loss. Effectively weighting these is key for efficient training.

2. **Kullback-Leibler Divergence (KL) Loss** This loss represents the smoothness of the distribution within the latent space. You want to avoid having values near 0 or extremely large.
3. **(Optional) Total Correlation (TC) Loss** This loss represents the "disentanglement" within the latent space. For the sake of stable learning, we excluded this objective from our experiments.
4. **(Auxiliary) Length Loss** This loss represents the length of the generated output. As transformers generally have outputs of fixed token length, yet we want to be able to generate an output of varying lengths, we decide to address this by employing a separate MLP head dedicated to predicting the output length. This also directly encodes the length into our latent space.

3 Experiments

3.1 Limited Data and Lightweight Model

We conducted two experiments to try and better determine the effect of our model complexity as well as our dataset distribution on learning and performance.

1. **Model trained on limited data** To try to make the task of mapping all sounds into a meaningful and smoothly distributed latent space easier for out model, we train on only a subsample of the original data, containing exclusively Snare, Hat, and Kick sounds.
2. **Smaller model trained** As our first models were intentionally huge (2.25GB), we wanted to see how reducing the total model complexity would impact performance.

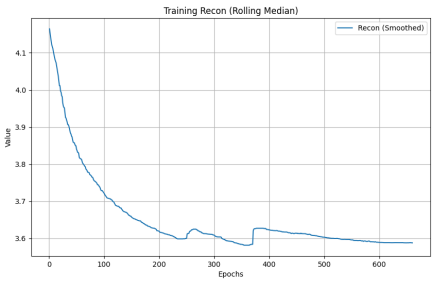
3.2 Reconstruction Loss Strategies

Multiple methods of weighting the individual reconstruction losses across all notebooks were tested.

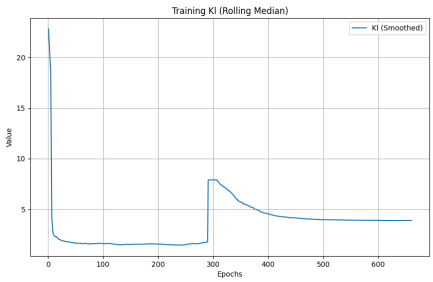
1. **Mean** The mean of all codebook reconstruction losses.
2. **Weighted Mean** The mean of all codebook reconstruction losses, with various weight decays applied. Specifically, a linear and exponential weight decay were applied.
3. **Sequential Mean Progression** First the reconstruction loss is only defined by the first codebook. As soon as this reaches a certain threshold loss, the next codebook's loss is added to the averaging. This is continued sequentially.
4. **Cascading Autoregression** We propose cascading the autoregression in multiple "dimensions".

4 Results

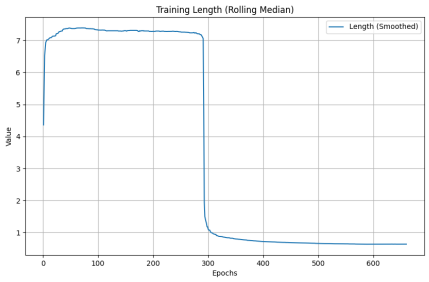
Our results show that the model is effectively able to optimize for each of its objectives, given appropriate hyperparameter adjustments. Judging the models outputs with our ears, we can tell that the outputs are not equal to their original sounds. Additionally, interpolation does not always lead to smooth results. However, as a general purpose drum sound generator, it functions as intended.



(a)



(b)



(c)

Fig. 1: Training Metrics. (a) **Reconstruction Loss** converges towards 3.6. Jumps in progression represent re-initializations of training with updated hyperparameters. (b) **KL Loss** stabilizes within the 1-100 range after re-initialization. (c) **Length Loss** drops below 1, indicating successful duration prediction.

5 Conclusion and future work

5.1 Limited Data and Lightweight Model

Due to even our smallest models being able to produce passable reproductions when using teacher forcing, we conclude that model complexity is not a limiting factor in performance or output fidelity, and a even less powerful model could possibly be used for this task. On the other hand, our findings on the impact of the data distribution are somewhat inconclusive. No improvement in performance can be found when we filter out all outliers and greatly reduce the training dataset variety. We acknowledge that this doesn't align with the expectations, and conclude that the datasets full distribution may be equally as suited as the reduced one.

5.2 Reconstruction Loss Strategies

Using either exponential weight decay or a sequential mean progression lead to the most acoustically recognizable progress. This suggests that focusing on optimizing the reconstruction loss for

lower level codebooks leads to improvements in training efficiency and possibly final performance. Finally, we propose future work to investigate the impact of a cascading autoregressive process, where each codebook prediction gets fed into the MLP to produce the next codebook prediction. While greatly increasing computational demand and also slowing down training by means of removing parallel processing capacity, this could in theory due to the heirarchical nature of the EnCodec codebooks lead to great improvements.

References

1. Défossez, A., Copet, J., Synnaeve, G., Adi, Y.: High fidelity neural audio compression (2022), https://arxiv.org/abs/2210.13438
2. Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., Norouzi, M.: Neural audio synthesis of musical notes with wavenet autoencoders (2017), https://arxiv.org/abs/1704.01279