Synthesis by Layering: Learning a Variational Space of Drum Sounds

Elliott Waissbluth
UC Berkeley
`ewaissbluth@berkeley.edu`

Jon Gillick
UC Berkeley
`jongillick@berkeley.edu`

Carmine Cella
UC Berkeley
`carmine.cella@berkeley.edu`

Abstract

In this work, we demonstrate a variational autoencoder designed to reconstruct drum samples using linear combinations from a small predefined library of existing samples. Inspired by the music production practice of layering two or more samples on top of each other to create rich and unique textures, we synthesize drum sounds by producing sparse sets of mixing coefficients to apply to the predefined library, which are then layered to create new audio samples. By training this model to approximate a range of professionally produced and recorded drum samples, we aim to learn a distribution over possible layering strategies given a fixed sample library, which we can subsequently sample from or otherwise manipulate. We find that varying a particular dimension of the latent vectors in the space learned by the model does not simply linearly scale the mixing weights; rather, it smoothly varies the perceptual nature of the sample by swapping different samples in and out of the sparse mixture. We present a user-interface prototype to engage intuitively with our system, discuss the performance of our modeling approach, and highlight potential applications in a studio production environment.

1 Introduction

Layering, the practice of stacking complementary sounds to create new timbres and textures, is one of the most powerful and widely used sound design techniques in music and audio production [Bazil, 2009, Pejrolo, 2012, Bazil, 2012]. While it is commonly thought of in association with orchestration, in which different groups of instruments play the same part, layering also plays an important role in designing short sounds like percussive hits or one-time sound effects. Regardless of the specific musical context, working with layers is an attractive choice because it enables creators to explore a broad range of design possibilities while at the same time maintaining a consistent underlying sonic palette.

In this work, we explore machine learning models for building layers, focusing on the setting of designing snare drum sounds. Drum sound design is an important yet challenging task for many creators, especially those in pop and electronic music, who need to balance competing pulls toward either conforming to stylistic expectations defined by genre boundaries, or establishing a set of distinctive and recognizable timbres that make up the signature sound of an individual artist. Previous research has engaged with drum sound design both as a retrieval problem (e.g. through beat-box style vocal queries [Mehrabi et al., 2018] or similarity-based search [Pampalk et al., 2004]) and as a synthesis problem (e.g. through generating new percussion sounds with Generative Adversarial Network (GAN) models [Nistal et al., 2020]). Neither the retrieval nor the generative approach, however, have incorporated layering into their analyses of drum sounds; this paper presents a first look into modeling drum sounds as layers.

While, in practice, the artistic process of designing composite sounds (sounds made by overlaying distinct components from different sources [Russ, 2012]) often involves several steps including processing layers individually (e.g. to keep the low frequencies from one layer and the high frequencies from another) and shaping envelopes of both the individual and layered sounds, we start with two defining steps: **(1)** choosing the set of sounds to put together, and **(2)** mixing the relative volumes of these layers.

In designing machine learning models for the layering task, we focus on a few underlying goals centered on creative applications. First, we would like to keep the number of "source" sounds (the database of sounds that we draw from when choosing layers) relatively small, so that artists can reasonably curate their own collection of samples; in our experiments, we use 200. Second, the synthesized sounds should be free of audible artifacts and, ideally, should come close to the subjective quality and realism of the professionally produced samples used for training. Third, we would like to be able to manipulate intermediate representations or latent variables in order to navigate the space of possible outputs. Finally, the outputs of the model should be sparse, so that only a few layers are used at a time. This sparsity makes it possible to quickly visualize and listen to the component layers in order to interactively turn them up and down, or on and off. This set of design considerations informs the modeling choices outlined in the rest of this paper, and our experiments aim to measure the degree to which we can make progress in these avenues.

In summary, this paper's contributions include the following:

- We introduce, motivate, and outline a set of design considerations for the task of modeling drum samples as layers of sparse linear combinations from a given sample library.
- We design, implement, and experiment with a variational autoencoder-based neural network for this task, and we examine its ability to approximate the sounds from professionally produced sample libraries.
- To explore possibilities for interactive creative applications, we investigate the behavior of our trained models through experiments in manipulating latent representations.

Audio examples are included in the Supplementary Material,[1] and our code will be made available upon publication in our GitHub repository. [2]

2 Related Work

2.1 Neural Audio Synthesis

A number of recent studies have explored methods for musical audio synthesis using neural networks, including synthesizing drum sounds with autoencoders [Aouameur et al., 2019] or GANs [Nistal et al., 2020]. Perhaps most similar to our approach is Neural Drum Machine [Aouameur et al., 2019], a method for creating drum samples with variational autoencoders that generate waveforms using a convolutional decoder architecture. In comparison with these kinds of methods for synthesizing raw sound waves, which cover a large space of output possibilities, our approach takes a narrower focus by enforcing a simple synthesis method based on layering. This choice emphasizes interpretability and interaction with model outputs (by allowing a user to easily visualize the mixing coefficients output by the model and change them), and it puts less burden on a decoder to generate sounds from scratch without artifacts. In exchange for these benefits, we sacrifice some of the flexibility offered by large neural decoders. Other recent work in neural audio processing has shown that replacing neural decoders with differentiable parametric synthesizers or effects [Engel et al., 2020, Martinez Ramirez et al., 2021] enables easier training with less data; our approach is trained end to end with Gradient Descent in the same fashion as these methods, but we make an even stricter assumption about the synthesis method in designing outputs with just a handful of adjustable parameters.

2.2 Assisted Orchestration

In restricting our model to predicting mixes from a predefined sample library, we draw from Assisted Orchestration [Cella et al., 2020], in which systems are designed to search for approximate matches

[1] `https://github.com/anonpapersubmit/SBL_AIMC2022`

[2] `https://github.com/elliottwaissbluth/DrumSynthesis`

a single target sound (such as a field recording of a real-life scene like chirping birds or whistling trains) by mixing and concatenating samples from a database (typically an orchestral sample library like SOL [Cella et al., 2020]). Our approach diverges from orchestration or concatenative synthesis [Schwarz et al., 2004], however, in that instead of searching for matches to a single target, we are instead interested in modeling a *distribution* of targets (defined by a training set of professionally designed drum samples) in order to explore the space of possibilities for how the sources in our library might be layered to sound like new professional samples.

3 Methods

3.1 Data

Our dataset is built from a broad collection of drum samples from professional sample libraries. We choose these samples because they represent real-world examples of the types of material used by producers in practice: some of these samples are live recordings of unprocessed drums, while others were created by professional sound designers using a range of synthesized, sampled, and processed sources. We construct a dataset of snare drum samples ($n = 2495$) from this collection to use for our prototype and experiments. We preprocess each data point by setting its sample rate $f_s = 22.05$ kHz, truncating to 1 second in length, and normalizing the amplitude.

From the snares dataset, we select 200 samples to represent the relatively small library of "source" samples available to the model, which are linearly combined to synthesize outputs. We choose these samples using the Greedy Frank-Wolfe Algorithm for Exemplar Selection [Cheng et al., 2020]. This is a computationally efficient algorithm which selects the n samples from a collection that are able to best represent the others as linear combinations. We find that this algorithm makes for easier model training than manual or random selection. We split the remaining 2295 samples into a 80-10-10 train-validation-test datasets to use as "targets" for training.

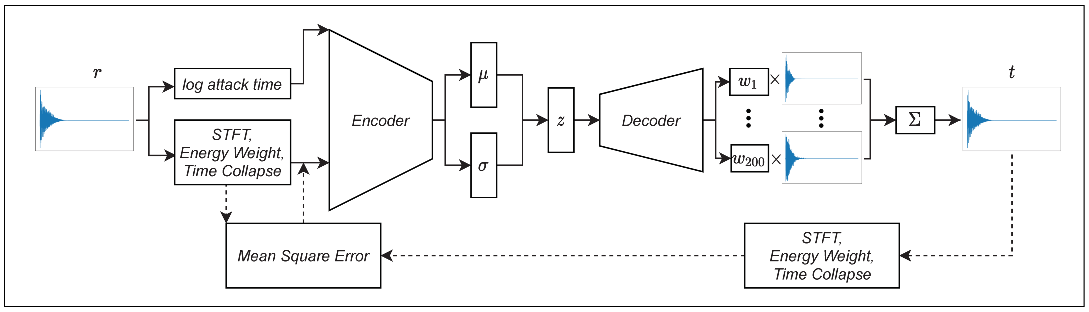
3.2 Features

We featurize drum samples using standard spectral features (2048-dimensional magnitude spectra) weighted by RMS energy. Following the procedure from [Gillick et al., 2019], we calculate this feature by first computing a magnitude spectrogram (at 44 frames per second with a hop size of 512 samples) and then averaging the spectral features across the time dimension, weighting by the RMS energy at each frame. This feature emphasizes the timbre near the onset of the sample while ignoring quieter tails that tend toward white noise. Because this approach to averaging over time discards information about the sound's envelope, we also calculate a feature for log-attack time, following [Peeters, 2004], concatenating this value to the feature representation. We expect that different feature representations here may lead to different modeling outcomes and user experiences; for now, we leave other featurizations to future work and instead focus on relatively simple features that we hope to be able to model reasonably well with a small dataset, while at the same time capturing some of the most distinctive timbral characteristics in the data.

3.3 Model

We use a variational autoencoder (VAE) [Kingma and Welling, 2014], which we train to reconstruct its d-dimensional input features from a compressed d_s-dimensional latent vector representation z, with $d_s < d$. We use multi-layer perceptrons (MLP) with a single hidden layer for both our encoder and decoder. We choose this architecture for its simplicity and its fast training speed and inference time. The encoder has a hidden layer of size $d_h^e = 1500$; it encodes the input features of the drum samples into a latent variable z with dimension 32. The decoder is smaller (because of the constrained output space), with a hidden layer of size $d_h^d = 144$, which feeds into an output with dimension $d_{out} = 200$. The values of this output layer are the weights to be applied to the 200 source samples, which are then multiplied and summed to produce the final audio output. We constrain the output weights to sum to 1 through normalization. Figure 1 visualizes the model architecture.

To calculate a reconstruction loss, we featurize the output in the same manner as the input, computing the energy weighted, time summed STFT, before computing the Mean Squared Error between the input and output vectors. To ensure that this loss is differentiable end-to-end, we use differentiable implementations of the STFT in Pytorch [Paszke et al., 2019]. To this, we add the other loss component of VAE training by computing the KL-divergence between the latent vector z and a Standard Normal distribution prior. Following the same implementation used by [Roberts et al., 2018], we use one hyperparameter β to weight between these two loss components and a second hyperparameter called "free bits" [Higgins et al., 2017], which allows a budget for z to diverge from the prior distribution before it incurs a loss. We use hyperparameter values of $\beta = 0.02$ and free bits $= 1000$.



r
log attack time
STFT, Energy Weight, Time Collapse
Encoder
μ
σ
z
Decoder
w1
w200
Σ
t
Mean Square Error
STFT, Energy Weight, Time Collapse

Figure 1: Model architecture. Solid lines describe inference, dotted lines describe training.

3.4 Training

We train our model in two phases: **(1)** a discriminative pretraining phase where the model learns to identify the source samples, and **(2)** a generative training phase where the model learns to reconstruct training data using the source samples. We find that pretraining on the source samples themselves improves the training loss during the generative training phase. We pretrain by running the 200 source samples through the model as training data using a binary cross-entropy classification loss, and then we train the generative model for 150 epochs using 1836 novel training samples.

4 Experiments

In this section, we explore several properties of our trained model with the aim of gaining insight into its potential applicability in creative production environments.

4.1 Interpolation

First, we are interested in exploring relationships between the learned space of embeddings z and perceptual qualities of generated samples. One of the main reasons for using generative models like VAE in creative settings is that they can provide a mechanism for controlling high-level properties of model outputs by manipulating latent representations. For our purposes, we seek an embedding space that captures smooth perceptual transitions along dimensions of timbre, texture, and frequency as we traverse a path between two points a and b via their corresponding embeddings z_a and z_b.

We approximate these perceptual qualities of generated drum samples using 13 Mel-Frequency Cepstrum Coefficients (MFCC) [Mermelstein, 1976] and energy weighted, averaged spectral centroids. We measure movement in these variables by computing absolute distance between averaged spectral centroids and Euclidean distance between MFCCs as we interpolate between z_a and z_b. These distance metrics provide an approximation for perceptual variation in generated samples. For comparison, we also compute distances in weight space (the space of possible mixing coefficients output by the model) via the Euclidean distance between the 200-dimensional output weight vectors as we interpolate between the embeddings of samples a and b.

We select 200 random pairs from the test set to interpolate between, computing pairwise distance metrics as well as the same metrics averaged across all 200 pairs. We find that as expected, on average, the output weights change linearly as we interpolate between two points. The two timbral features of the generated samples (MFCC and spectral centroid) also vary similarly. These trends are visualized in Figure 2.

Notably, however, interpolation does not produce linear paths between every *individual* pair of samples; the weight distance may vary significantly between sample a and b before settling at either sample. This suggests that the model has not converged to a trivial solution equivalent to turning all 200 mixing coefficients up or down at the same time; rather, the model swaps different source samples in and out of the layered mixture, indicating that the embedding space appears to have captured some abstract features of the data that correlate with timbre. This means that, for example, when decoding from an embedding z_c that is halfway between z_a and z_b, we might get an output drum sample with timbral characteristics somewhere between a and b, but which highlights a drum layer that is not used at all when we decode z_a or z_b; we argue that this property may be useful for finding new layering possibilities. Figure 3 displays plots of interpolations between 3 pairs of samples.

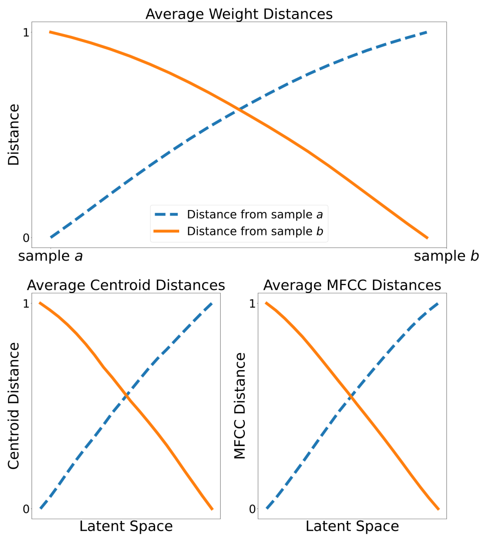


Figure 2: The weight distances, centroid distances, and MFCC distances vary linearly through interpolation on average. In each chart, the average of distances are plotted over the latent space between samples a_i and b_j, $i, j \in [1, 200], i \neq j$.

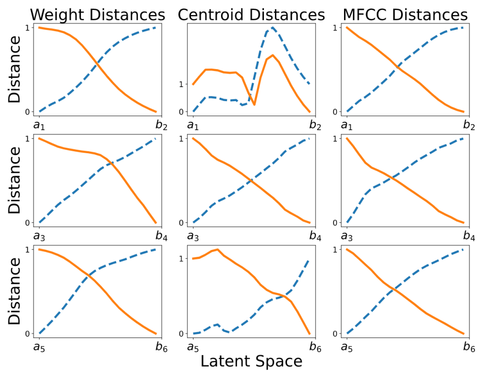


Figure 3: Weight, centroid, and MFCC distances for select sample interpolations. Notice how individual samples do not necessarily follow a linear path through perceptual space.

4.2 Varying Latent Representations Using PCA

We run Principal Component Analysis (PCA) on the latent vectors encoded by the model for the entire snares dataset to extract 11 eigenvectors corresponding to the directions of maximal variance. By manipulating latent representations of layered samples along these dimensions, we hope to uncover relationships between the learned latent space and perceptual dimensions. For example, one might find that varying the latent variables along some principal component corresponds to a perceptual shift in frequency, another principal component might correspond to sharpness, etc. Our interface prototype shown in Figure 6 demonstrates how users might interact with the model in this way.

As we vary a latent vector along the principal components, we find that timbre, frequency, and texture are modulated. In practice, the principal components are unpredictable in regards to the exact perceptual quality they will correspond to across different samples. One input sample might increase in frequency along the first principal component while another decreases in frequency along the same dimension.

Regardless of the perceptual mapping, we find that varying latent vectors along principal components produces a gradient of output samples that vary smoothly in perceptual quality. We show this by encoding a source sample to extract its latent representation z, before varying the encoded latent representation along the principal components as the output layers and their weights are displayed in real time. In Figure 4, we demonstrate the effect of varying the latent space along the first principal component for a particular sample.

Figure 4 suggests that the model varies perceptual qualities by layering certain samples in and out of a combination. We see several weights which persist throughout the variation, giving the sample its core tone. As auxiliary samples are layered in, the timbre and texture of the drum shift without deviating entirely from the initial reconstruction of the input. The last frame of the variation (corresponding to

$z = z_0 + 20u_0$) suddenly reduces the weights of many persistent source samples. The manipulations to the principal components were chosen to explore the latent space to the boundaries of the region in which it produces sparse outputs; this last frame shows a reconstruction where z has passed that boundary.

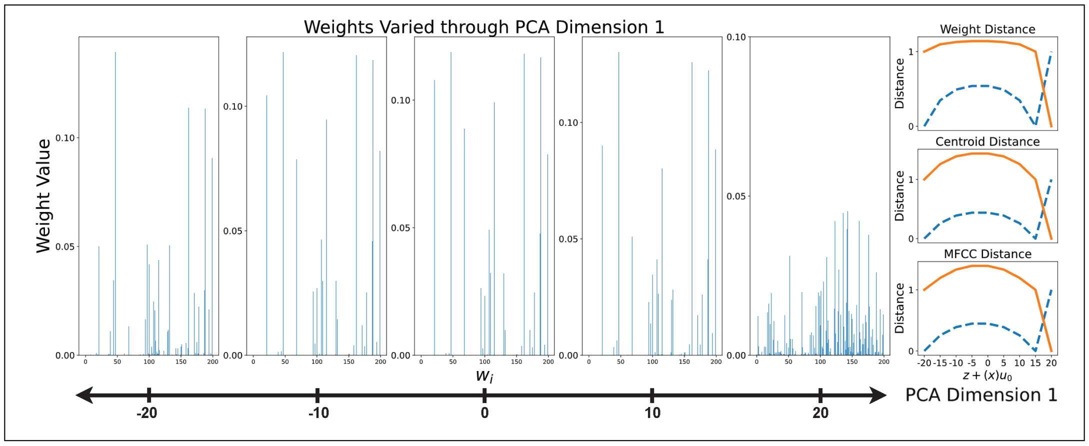


Figure 4: Beginning with a sample from the test set, we extract the latent representation z and vary it through PCA dimension 1, u_0. The output weights change as the latent representation is varied along this dimension. Perceptually, this variation translates to moving from a low-frequency, short attack drum sample to a high frequency, long attack drum sample. This distance plots are calculated along the latent space in$[z - 20u_0, z + 20u_0]$.

4.3 Cardinality of Output Space

From a usability perspective, it is important that the output space be sparse - a user who wants to synthesize a layered drum sample will likely find sparse outputs to be more understandable and easier to change by hand than an amalgamation of 200 equally mixed samples. For a creator, it is simpler to tune a selection of a few samples than to parse through many. Even though we do not explicitly enforce sparsity among the output weights during training, we find that most solutions do turn out to be sparse. The results are presented in Figure 5. Recall, we constrain the weights of the output space to sum to 1.

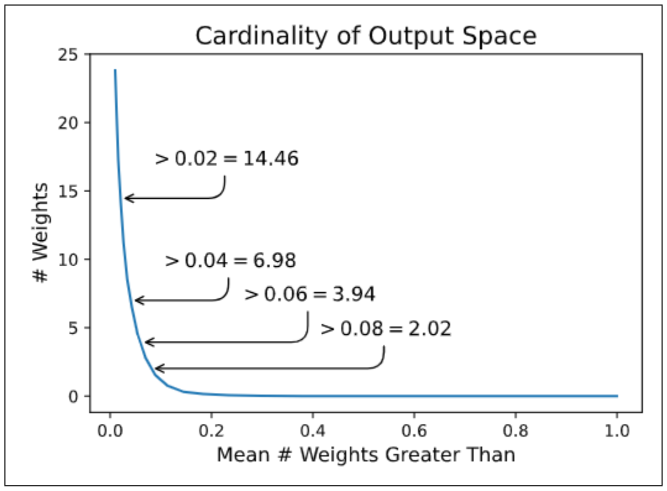


Figure 5: In this plot, we see the model generally produces sparse output. Select points are highlighted to demonstrate sparsity greater than a particular weight value.

Figure 5 shows that usually no more than about 15 weights meaningfully contribute to model outputs; in the examples shown in 4, clear peaks usually appear showing an emphasis on a handful of samples used for layering. At the same time, in most reconstructions, the maximum output weight does not exceed 0.15, which indicates that the model is indeed choosing to create layers, rather than collapsing to the trivial solution of choosing a coefficient of 1 for a single, most similar, sample. We argue that an output space with this cardinality profile can produce interesting behavior from the perspective of a user. The model selects few enough samples to be comprehensible and enough samples to produce smooth variation between latent space variables.

4.4 Reconstruction

Our primary focus in this paper is on designing a model that holds potential as a useful creative tool for exploration during a sound design process; for this reason, we do not emphasize the quantitative performance of the model's ability to reconstruct samples. Other methods, such as the Matching Pursuit algorithm [Gribonval and Bacry, 2003], may be more suitable for optimally approximating a given target from a linear combination of sources. Instead, we explore the model's reconstruction capabilities subjectively using samples from the validation set as reconstruction. We find that the model is capable of approximating the core timbre and texture of most of the snare drums in the validation set. The model's raw output is often noisier than the input sample if we listen directly to the reconstruction without strictly enforcing sparsity constraints, because a large number of source samples mixed in at low levels combine into an undercurrent of noise. This can be mitigated, however, by removing all source samples with weights lower than a certain threshold (e.g. 0.04) before we listen to samples. We include raw reconstruction samples in our supplementary materials.[1]

5 Interface and Applications

5.1 Interactive Prototype

Our user-interface prototype leverages the perceptual qualities found by varying the latent representations z along their principal components. We encode these dimensions as sliders which the user can control to generate samples in real time. First, the user can upload a sample to work from as a starting point. Encoding the sample with the model, the user is left with a reconstruction of that sample using a combination of the 200 source samples. By adjusting the encoding along any of the 11 principal components, including multiple principal components simultaneously, it is possible to explore a wide range of layering combinations that alter the timbre and texture of the original sample. The user may also change the weights for a particular layer manually by selecting them from the visual output.

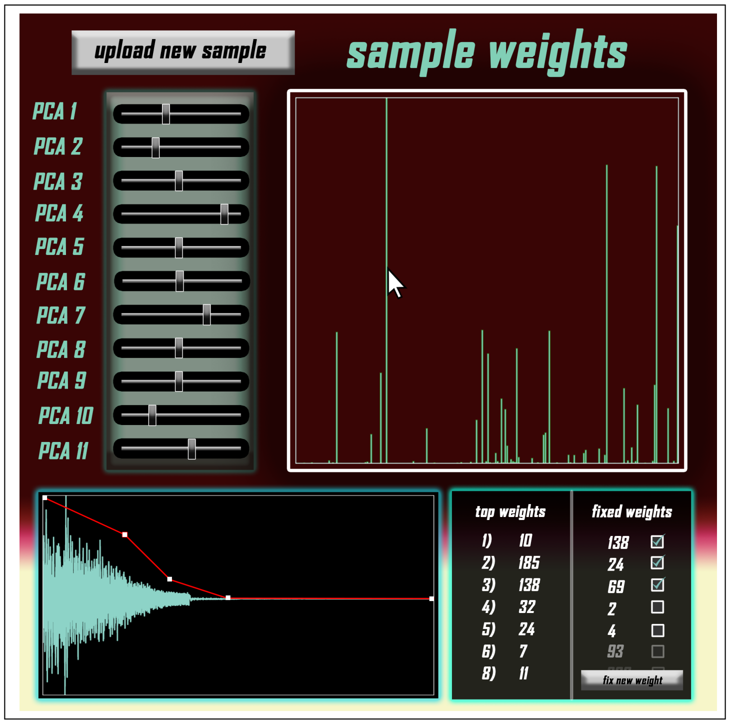
As the user moves through the model's latent space, the output weights are displayed. The user sees a wave of output weights that fade in and out as the timbre and texture of the generated sample varies. From this, we hope to provide users an intuitive sense of the inner workings of the model. We present a mock-up of our prototype in Figure 6.

5.2 Applications in a Studio Production Environment

Drum sound design is a difficult task for many creators, who are faced with the challenge of balancing stylistic expectations within their genres, while at the same time needing to establish unique and recognizable timbres that define their signature sounds as an artist. Creators may find it overwhelming to browse through libraries containing thousands of drum samples, with no obvious place to start looking for samples to layer together. Our model offers an approach to simplifying this process by automatically generating layered drum samples using only a specific set of samples as component layers. We envision that creators would curate their own sets of source samples to train this model. We believe that by retraining this model using their own samples, our approach can provide a powerful tool that serves as a basis for exploring timbres and textures using layered samples.

5.3 Future Work

Music creators are interested in layering other types of drum samples beyond snare drums, as well as other instruments more broadly. Future work might generalize the layering approach proposed in this paper to a broader suite of samples and instruments to provide music producers with a full layering toolkit.



top weights	
1)	10
2)	185
3)	138
4)	32
5)	24
6)	7
8)	11

fixed weights	
138	☑
24	☑
69	☑
2	☐
4	☐
93	☐

Figure 6: The interactive system includes PCA sliders, weight fixing, and envelope control. A user will see the weights vary as the PCA sliders are manipulated in real time.

6 Conclusion

In this paper we presented an approach for sound design that focuses on the concept of layering drum samples, a common practice in music production. The core idea behind our approach is to use a variational autoencoder as a means to discover a distribution over possible layering strategies using a small pre-defined library of source samples, which can ultimately be used to produce new sounds while still fitting into an artist's sonic palette. We explored new studio-production possibilities offered by the affordances of this neural network model. The user-centric method we developed has several advantages from a musical standpoint:

- It produces a set of parameters that is easily interpretable by the user.
- The synthesized sounds are generated from a limited number of samples (sparse), thus being conducive to easy manual manipulation for fine-tuning by hand.
- The system requires a relatively small number of samples to be trained, allowing the creation of ad-hoc datasets.
- The variational latent space we learn has a quasi-linear behaviour and smooth variation in perceptual space, a valuable feature for exploration by users during music production.

We also proposed a preliminary user-interface to engage with a model, which relies on a PCA-based dimensionality reduction of the learned latent space. While the proposed system is relatively simple, it is able to create production-grade drum samples and shows potential for real applications.

We believe the idea of using generative networks such as VAE for discovering the parameters of an interpretable model for layering samples (as opposed to generating sound samples directly) has a lot of potential for music applications and we hope that this work will foster some reflection in this direction.

References

[1] Cyran Aouameur, Philippe Esling, and Gaëtan Hadjeres. Neural drum machine: An interactive system for real-time synthesis of drum sounds. *arXiv preprint arXiv:1907.02637*, 2019.

[2] Eddie Bazil. *Art of Drum Layering*. PC Publishing, 2009.

[3] Eddie Bazil. Layers of complexity: The sos drum-layering masterclass. *Sound on Sound*, Nov 2012.

[4] Carmine Emanuele Cella, Luke Dzwonczyk, Alejandro Saldarriaga-Fuertes, Hongfu Liu, and Helene-Camille Crayencour. A study on neural models for target-based computer-assisted musical orchestration. In *2020 Joint Conference on AI Music Creativity (CSMC+ MuMe)*, 2020.

[5] Carmine Emanuele Cella, Daniele Ghisi, Vincent Lostanlen, Fabien Lévy, Joshua Fineberg, and Yan Maresz. Orchideasol: a dataset of extended instrumental techniques for computer-aided orchestration. *arXiv preprint arXiv:2007.00763*, 2020.

[6] Gary Cheng, Armin Askari, Kannan Ramchandran, and Laurent El Ghaoui. Greedy frank-wolfe algorithm for exemplar selection. 2020. URL `http://arxiv.org/abs/1811.02702`.

[7] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

[8] Jon Gillick, Carmine-Emanuele Cella, and David Bamman. Estimating unobserved audio features for target-based orchestration. In *ISMIR*, pages 192–199, 2019.

[9] Rémi Gribonval and Emmanuel Bacry. Harmonic decomposition of audio signals with matching pursuit. *IEEE Transactions on signal processing*, 51(1):101–111, 2003.

[10] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning Basic Visual Concepts With a Constrained Variational Framework. page 13, 2017.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[12] Marco A. Martinez Ramirez, Oliver Wang, Paris Smaragdis, and Nicholas J. Bryan. Differentiable signal processing with black-box audio effects. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021.

[13] Adib Mehrabi, Keunwoo Choi, Simon Dixon, and Mark Sandler. Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 356–360. IEEE, 2018.

[14] Paul Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, pages 374–388, 1976.

[15] Javier Nistal, Stephan Lattner, and Gaël Richard. Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks. *arXiv preprint arXiv:2008.12073*, 2020.

[16] Elias Pampalk, Peter Hlavac, and Perfecto Herrera. Hierarchical organization and visualization of drum sample libraries. In *Proc Intl Conf Digital Audio Effects*, pages 3–8, 2004.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[18] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2004.

[19] Andrea Pejrolo. *Creative sequencing techniques for music production*. CRC Press, 2012.

[20] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *International conference on machine learning*, pages 4364–4373. PMLR, 2018.

[21] Martin Russ. *Sound synthesis and sampling*. Routledge, 2012.

[22] Diemo Schwarz et al. Data-driven concatenative sound synthesis. 2004.