# Creating Latent Representations of Synthesizer Patches using Variational Autoencoders

Matthew Peachey
*Graphics and Experiential Media Lab*
*Dalhousie University*
Halifax, Canada
peacheym@dal.ca

Sageev Oore
*Faculty of Computer Science*
*Dalhousie University*
Halifax, Canada
sageev@dal.ca

Joseph Malloch
*Graphics and Experiential Media Lab*
*Dalhousie University*
Halifax, Canada
jmalloch@dal.ca

***Abstract*—Digital synthesizers typically feature a user-adjustable parameter space (i.e. the set of user-adjustable parameters) that is used to shape the sound (or timbre) of the instrument. A synthesizer patch is a snapshot of the state of the instrument's parameter space at a given time and is the representation most familiar to synthesizer users. Creating patches can often be repetitive, tedious, and complicated for synthesizers with large parameter spaces. This paper presents the creation and use of latent representations of synthesizer patches generated by training a Variational Autoencoder (VAE) on a library of existing patches. We demonstrate how to generate previously unseen patches by exploring this latent representation via interpolation through the latent space. Using the open-source synthesizer amSynth as a test bed, we evaluate reconstructed patches against a ground truth both, numerically and timbrally, as well as show how generating new patches from the latent space result in diverse yet musically pleasing timbres.**

## I. Introduction

Digital synthesizers are capable of creating sounds with a wide range of *timbres*—the term used to describe the various perceptual qualities of a sound other than pitch and loudness [22]. Musicians typically shape the timbre of a synthesizer by adjusting *parameters* of their instrument to create *patches*—snapshots of the state of the entire instrument. The creation of synthesizer patches is not always a straightforward process due to the potential complexity of an instrument's parameter space, both in terms of dimensionality as well as the interrelationships between parameters and their combined effects on timbre. Users may either not be familiar with the inner workings of a given synthesizer or simply not have the bandwidth to tediously test hundreds of combinations of parameter values in order to find their desired sound. Due to the combination of each of these factors, computer assisted synthesizer patch creation is an interesting area of research. Researchers have explored techniques for creating new synthesizer patches including live collaboration [15], genetic algorithms [19], and stochastic exploration [14]. Each of these techniques aim to reduce the amount of effort a musician has to spend adjusting their instrument to find a desirable sound. In the case of live collaboration, the amount of effort required to create a desirable patch is reduced simply due to the number of musicians working on the task at the same time. In the case of genetic algorithms and stochastic exploration, new patches are identified through algorithmic analysis of the parameter space or properties of the resulting sound, allowing the user to run these operations in the background without explicit control over each step of the system.

Our research presents a different approach to synthesizer patch creation. We explore the use of Variational Autoencoders, and how the latent spaces derived through the training of these unsupervised machine learning (ML) models can enable patch generation by providing musicians with an alternative representation of the range of patches and timbres their instrument can produce. A well-formed latent space is typically a much lower-dimensional representation of the original data, which when sampled from will produce "realistic" outputs (i.e. indistinguishable from real, or given, data).

In this work, we aim to create an extremely low-dimensional latent space for synthesizer patches, while still generating musically interesting timbres, to allow users to explore that space through interpolation while using a minimal number of degrees of freedom. Specifically, we explore a two-dimensional latent space, as this couples well with human spatial experience and many available input devices. In doing this, we allow users to interpolate in real time through a latent space, resulting in the generation of many synthesizer patches (i.e. one per "sampling" of the latent space) using a dimensionality that is both familiar and compact. It is important to note that our VAE-based method of synthesizer patch creation should not be viewed as a replacement for creating patches from scratch, as that technique has many desirable features (increasing familiarity with the inner workings of a given instrument, a high level of precision over timbral qualities, creative aesthetic, etc.) but rather as an alternative creative tool that musicians can work with if they wish.

This paper presents an overview of our work using Variational Autoencoders for generating latent representations of synthesizer patches. We provide a brief background motivating the use of generative ML / VAEs in a creative context, an overview of our ML pipeline, and finally a discussion around the results of this work.

## II. Background

### A. Digital Synthesizers

Digital synthesizers are instruments that generate audio signals using digital signal processing. A user-adjustable "parameter space", which allows musicians to make adjustments to the sound of their instrument, is a key component of digital synthesizers. While early synthesizers exposed only simple parameters such as pitch and amplitude [3], both the number and complexity of parameters in modern synthesizers have increased. The specifics of a synthesizer's parameter space are dependent on both the synthesis technique as well as the synthesizer's specific implementation. The concept of a "patch" was introduced as a method of saving a snapshot of the state of a synthesizer for use at a later time. Many digital synthesizers allow users to save patches directly within the instrument, allowing them to be quickly recalled at a later date or be shared with other musicians.

For this research, we use amSynth [4], an open-source synthesizer that follows a typical subtractive synthesis paradigm [10] and includes a range of features including MIDI control and user-defined patch banks, making it a well rounded and highly usable instrument. Amsynth's parameter space includes 30 continuous parameters and 11 categorical parameters resulting in a total parameter space size of 41.


Fig. 1: A screenshot of amSynth's user interface. This synthesizer follows a typical "subtractive synthesis" paradigm and includes features such as MIDI control and reverb. This synthesizer has a reasonably sized parameter space with 41 user adjustable parameters.

### B. Generative Machine Learning

Unlike typical supervised ML architectures (i.e. models that use labelled data to learn and predict outcomes), unsupervised learning attempts to learn about any underlying patterns or structure found within the entirety of a given data set [11]. Recent work in deep learning has produced unsupervised ML architectures that not only learn about patterns found within data, but also learn probabilistic distributions of the given data set. These architectures are then able to use those learned distributions to generate new data that upon inspection are determined to be plausible members of the original data set [7]. These models are called "Generative Models", and have many interesting applications, including within the music and creativity spaces.

### C. Variational Autoencoders

Variational Autoencoders (VAEs) are a type of generative machine learning model that are used to learn lower dimensional representations of unlabelled data [9]. As shown in Figure 2, VAEs aim to reduce the original feature space of a dataset into a latent space, typically denoted as $z$, while minimizing the amount of information loss. This latent space is represented by a Gaussian probability distribution with mean ($\mu$) and variance ($\sigma$). VAEs work by training two symmetrical neural networks to encode the original data into a latent space with the goal of minimizing the reconstruction error of that data upon decoding from the same latent space [12]. Once the encoder and decoder have both been trained, a user is able to run inference on the decoder by sampling the latent space $z$ resulting in the generation of previously unseen datapoints.

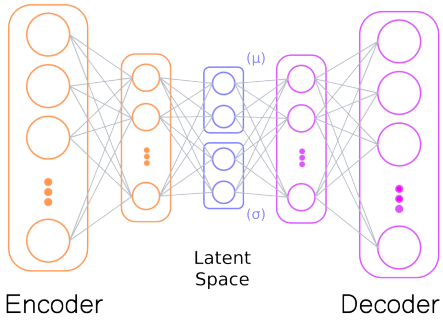



Fig. 2: Our VAE's architecture. The encoder compresses the original data into a latent space and the decoder reconstructs latent vectors back into the data's original format. The VAE architecture is trained by minimizing the difference between real and reconstructed data as examples are shown to the model.

### D. Machine Learning for Music

Machine learning, and more specifically generative machine learning, has seen recent progress in areas of the computer music and creativity space. Projects such as GanSynth [5] make use of Generative Adversarial Networks (an unsupervised ML architecture similar to VAEs but trained in a fundamentally different manner) [8] to generate entire sequences of audio in parallel from a single latent vector. This architecture has also been used as a method for morphing between realistic sounds [6]. Pati et al. used VAEs as a method for generating new measures of music [16] that fluidly connect to other pieces. Similarly, Roberts et al. introduced MusicVAE which they used to smoothly interpolate between two musical melodies [17]. Furthermore, Chen et al. demonstrates how decisions regarding the representation of musical data can have an effect on the success of training ML models [2]. VAEs have also already been effectively used for synthesizer patch creation

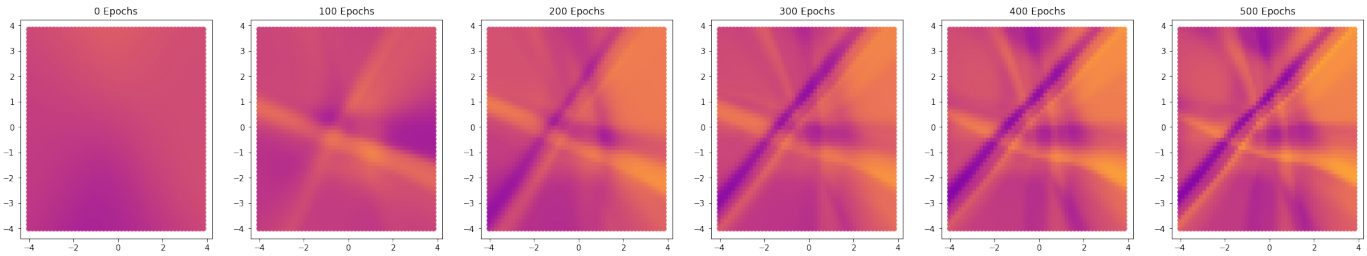



Fig. 3: Emergence of a latent structure for the "*Filter Cutoff Frequency*" parameter. We observe that a latent structure emerges quickly as the loss curve trends towards 0 and changes become more subtle as the loss curve plateaus. A perceptually uniform colour map is used to show the range of values throughout the latent space.

in recent literature. Le Vaillant et al. showed that VAE's can both infer parameters from audio clips as well as generate new patches when using a very large latent dimension [20]. Furthermore, Roche et al. showed that dedicating a subset of the latent space to learning perceptual ques reduced audio quality but increased the interpretability of the latent space [18]. Finally, deep learning models such as VAEs typically require a tremendous amount of training data, but recent work in the space has shown that "small data" approaches can also be effective in a creative context. Vigliensoni et al. introduced R-VAE which achieved good results when exploring latent spaces for musical rhythms by training a VAE on a small amount of data [21]. Each of these projects show the value in latent-space based methods for musical creativity & expression, and serve as inspiration for our work.

## III. Methods

The goal of this research is to discover methods for using generative ML, specifically VAEs, to represent synthesizer patches with a latent representation in order to generate new patches. The following methods section describes the data processing, ML architecture, and training processed used for our work.

### A. Data Pre-processing

One reason for working with amSynth for this research was the fact that it comes with nearly 3000 pre-made patches from its user base. This amount of pre-existing data was extremely beneficial for conducting ML experiments, as having to create that many patches on our own would be tedious, time consuming, and run the risk of being repetitive or introducing researcher-caused bias. To make use of the existing patches, we wrote a small python script that converted and saved the patches into a .csv file. Next, we populate a pandas dataframe using that .csv file, convert categorical variables (such as waveform or filter type selectors) into *one-hot-encodings* and finally normalize the data for use with neural networks. The result of our pre-processing pipeline is a pandas dataframe with 2586 rows, each of which containing 56 columns with values ranging between 0 and 1. This data is finally split into a validation set ($\sim 10\%$) and train set ($\sim 90\%$) for use with the remainder of our ML pipeline.

### B. Model Architecture

As shown in Figure 2, a VAE utilizes both an encoder and a decoder, each of which are implemented as neural networks. Our encoder utilizes an input layer with 56 neurons (which corresponds to an original data point) and feeds into a hidden layer with 32 neurons and finally into two output layers (one each for mean ($\mu$) and variance ($\sigma$)) with two neurons each which corresponds to the dimension of the latent space. Our decoder correspondingly follows a 2 neuron, 32 neuron, 56 neuron structure in the network, resulting in the same size data at either end of the VAE architecture. As discussed earlier, we use a two dimensional latent representation in order to allow users to easily navigate the space using familiar interfaces.

### C. Model Training

A critical part of creating effective ML models for research is the training loop and selection of hyper-parameters [13]. We trained our VAE for 500 epochs with a learning rate of 0.001, a batch size of 16, and the Adam optimizer. We use a multi-component loss function that combines Mean Squared Error (for the continuous features), Cross Entropy Loss (for each of the categorical / one hot encoded features) and Kullback-Leibler divergence (for the probabilistic distribution as is typical in a VAE architecture). Figure 4 shows both the training and validation loss curves plotted over the duration of the training loop. We note that while both training and validation loss steadily decrease, the validation loss does not converge with training loss, indicating that the compression of the data set into a small latent space results in a lossy reconstruction of that data. While both the training and validation loss curves trend downwards, a result that appears to indicate successful training, it is essential to evaluate actual generated samples and listen to the resulting sounds in order to validate the VAE's true effectiveness for synthesizer patch generation.

Furthermore, Figure 3 shows a single synthesizer parameter's latent representation being learned, in this case we show the values of the *Filter Cutoff Frequency* parameter as sampled from the latent space. We see that before the model is trained (i.e. after 0 epochs) the latent space has no substantial structure, but is rather in an initial random state. As the VAE is trained, we observe a latent structure emerge. This emergence of a latent representation can be observed across all of the

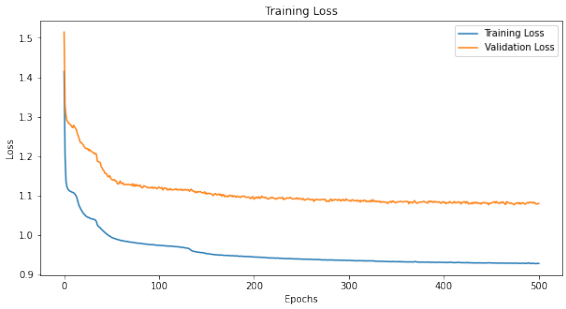



Fig. 4: The training and validation loss curves after training our VAE for 500 epochs. The downward trending curves indicates that the loss quickly decreases early on and then continues to decrease slowly over time.

other parameters of the synthesizer, including both categorical as well as continuous parameters.

## IV. Results & Discussion

Upon completion of the training process for our VAE, we evaluate the resulting model to determine its effectiveness at generating synthesizer patches. While analysing quantitative metrics associated with the model—such as a loss curve—provides insights into the model, it does not tell the full story. It is much more important to compare generated patches to their "ground truth" counterparts (i.e. samples from the dataset) as well as evaluating the quality of the patches that are generated when passing new latent vectors to the decoder of the VAE. Both of these evaluatory processes are discussed in the following sections.

### *A. Timbral Comparison between Real and Reconstructed Patches*

The AudioCommons project provides a toolkit[1] for measuring timbral qualities of a sound file which we use to analyze and evaluate our generated synthesizer patches. This tool provides models for measuring *hardness, depth, brightness, roughness, warmth, sharpness, booming, and reverberation* [1], each falling within a normalized range of 0 to 100. Our first evaluation for the effectiveness of a VAE for generating synthesizer patches is to compare the timbral qualities of patches from the validation set against the reconstruction (via the VAE) of those same patches. For each patch, we record a four second audio clip of the synthesizer playing a C4 note at a constant loudness. We choose not to analyze reverberation, as this quality is measured as a binary classification rather than a numerical value and thus does not provide enough relevant information to be included.

As visualized through a parallel coordinates chart in Figure 5, we see that the reconstructed patches do not reach the same local extrema as the real patches with respect to each of the timbral qualities. Instead, the VAE appears to generate reconstructions of the original data that squashes the timbral qualities into a smaller range of values. We recognize that this compression is due to the lossy nature of our very low dimensional latent space, but are willing to make this trade off in order to provide users with representations of synthesizer patches that are in familiar and easily understood dimensionalities. Furthermore, although we are using timbral analysis of the resulting sounds to determine the effectiveness of the VAE for synth patch recreation, the VAE is only being trained on parameter values and not on any timbral attributes of the sound of patches. We anticipate that this further contributes to the imperfect representations observed in timbre space.

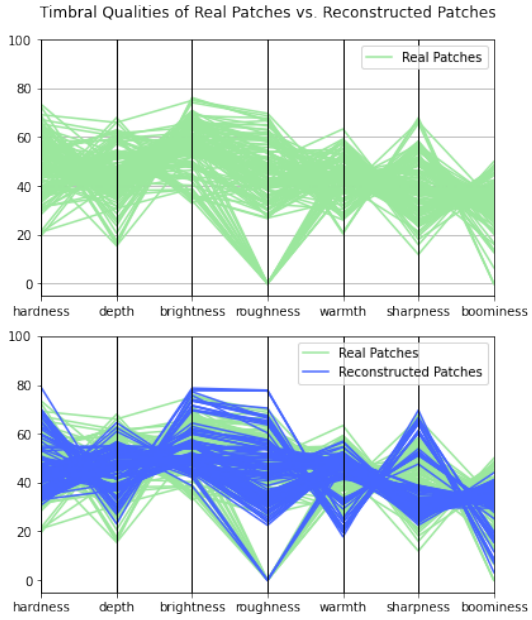


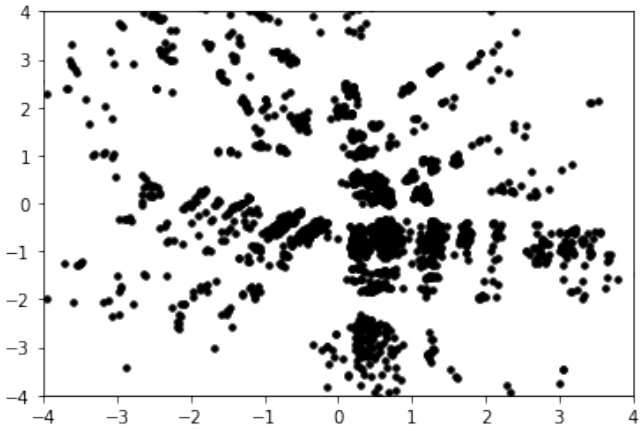

Fig. 5: Parallel coordinates visualization of the real data vs. reconstructed data in terms of timbral qualities. We observe that the reconstructed data does not capture the entire range of values for each timbral quality of the patches from the real data.

To accompany the parallel coordinates visualization, we also compute the error for each timbral quality's reconstruction using the root mean squared error (RMSE) metric. RMSE's output will be on the same scale as the target variable, which in this case will be between 0-100 as measured by the AudioCommons timbre analysis toolkit. We show the error distribution for each timbral quality in Figure 7, where we see that the *boominess* quality is reconstructed with the least error (mean of 8.34) whereas the *roughness* quality performs significantly worse than the others (mean of 17.68). Table I provides a breakdown of each timbral quality's error measurement.

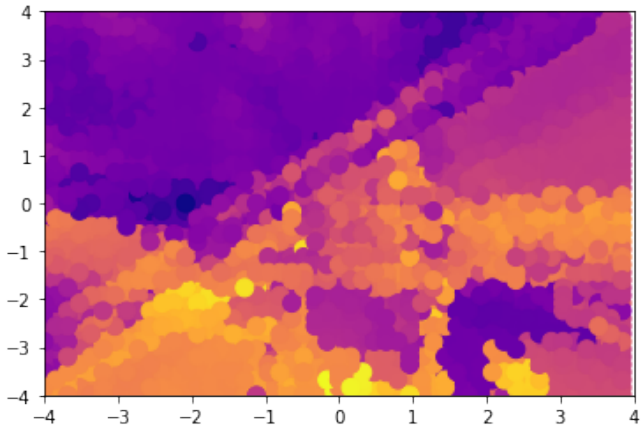
### *B. Generating New Patches with VAE*

Generating new patches with our VAE is done by simply sampling from the latent space. PyTorch, the ML framework used for this research, allows programmers to treat components of their ML model as functions that process user input. By

[1] https://github.com/AudioCommons/timbral_models



(a) Latent Coordinates Representation for the corpus of synthesizer patches used to train our VAE. Each point is the latent vector that the Encoder of the VAE outputs for a given patch.



(b) Timbral Representation of our VAE's Latent Space (*depth* is shown here). The latent space is uniformly sampled across the X and Y axis and timbral analysis is conducted on an audio recording of each latent vector's resulting synthesizer patch.

Fig. 6: The latent space of our VAE trained on synthesizer patches. Each of these representations of the latent space are provided as an interactive GUI that allows users to explore the space and generate new synthesizer patches in real time.

| Timbral Quality | Mean | Standard Deviation |
|---|---|---|
| Hardness | 11.74 | 8.17 |
| Depth | 10.96 | 7.73 |
| Brightness | 10.33 | 6.65 |
| Roughness | **17.68** | 14.57 |
| Warmth | 9.69 | 7.13 |
| Sharpness | 12.64 | 9.18 |
| Boominess | **8.34** | 7.52 |

TABLE I: The mean and standard deviation of the root mean squared error between real & reconstructed patches as measured for each timbral quality. Minimum and maximum mean errors are noted with bold font.

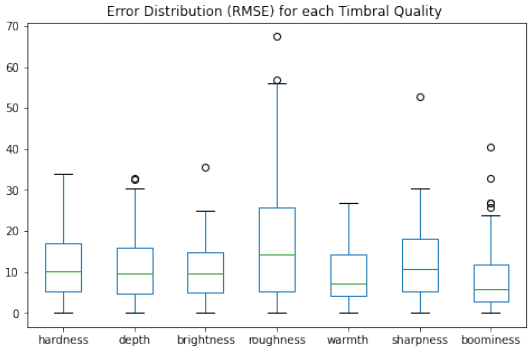



Fig. 7: Box-plot visualization of error distributions between real and reconstructed patches for each timbral quality. The *boominess* quality is reconstructed the best whereas *roughness* performs the worst.

passing a tensor with two components (i.e. a latent vector) to the decoder, the model will return a tensor of length 56 which matches the original format of the processed data set. In order to convert that tensor into the format expected by amSynth's preset loader, we select the maximum value from each one-hot encoded segment for categorical parameters and re-scale each normalized value back to its expected range for continuous parameters. Finally, the re-scaled data is associated with its parameter name and the result is saved as an amSynth readable file.

Interpolation-based exploration of the latent space allowed us to evaluate the VAE by listening to many of the samples it generated. By interpolating through the latent space in small increments using a purposed built graphical user interface, we are able to identify many interesting patches with a range of different timbres. We observe synthesizer patches that produce pleasing timbres (such as gritty bass and warm leads) without any human modification to the generated patch. We recognize that patches & timbres may often be improved by even small tweaks from a human user, which is certainly an interesting and creative workflow that we encourage. For example, a musician may navigate through a latent space in order to get close to their desired timbral qualities and then opt to use the more traditional knobs & sliders representation to make precise changes to the timbre of their instrument. However, the ability for our VAE to learn to generate patches with these classic timbral qualities without any human adjustments points towards a promising use for a VAE-based workflow for synthesizer patch generation.

We intend to further validate these findings through an upcoming user study that will be focused on both sound-matching and sound-discovery tasks. This study will aim to better understand if interpolating through a latent space is more effective (in terms of time, effort, etc.) than using a traditional synthesizer interface for creating patches as well as allow synthesizer users to assess the quality of the patches generated by our VAE. The results of this study will provide

important insights into the effectiveness of utilizing VAE's with low-dimensional latent spaces for generating synthesizer patches.

### C. Representations of Synthesizer Patches

Overall, the patches and associated timbres generated by exploring the latent space of our trained VAE were observed to be musically interesting and structured. By learning an underlying structure of a patch dataset we are able to provide musicians with two new representations of synthesizer patches, each of which provides a different way for musicians to interact with the instrument.

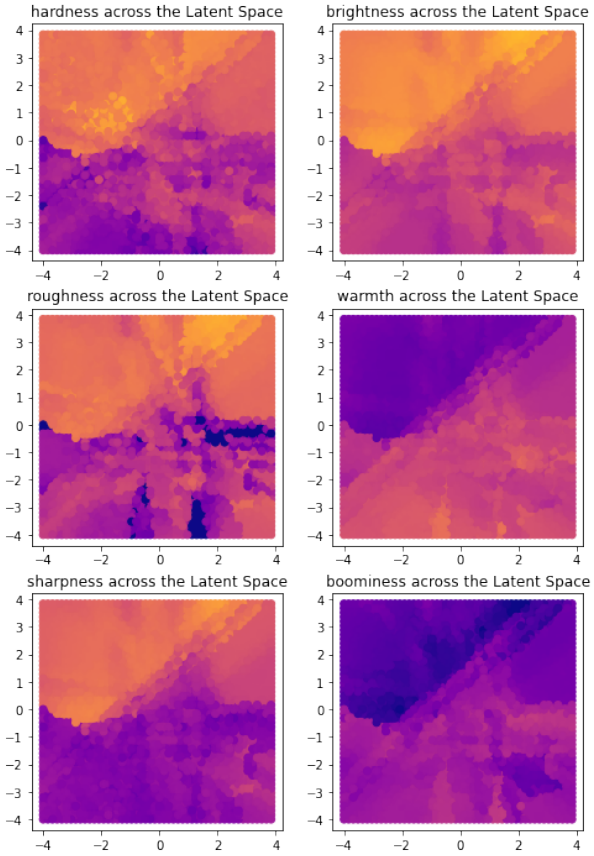



Fig. 8: Timbral representations of the Latent Space for each of the remaining descriptors. Each representation is generated by uniformly sampling from the latent space and coloring the space based on the resulting timbral values.

Firstly, the actual parameter space of the synthesizer (e.g. Figure 1) is already a valid representation of synthesizer patches that many experienced users are familiar with. The first representation that our research provides is shown in Figure 6a and demonstrates what we call "Latent Coordinates", where each scatter plot point represents an existing patch from the training dataset while the remainder of the space can also be sampled from in order to generate new patches. This representation is derived by encoding every existing patch using the trained VAE and plotting the resulting two dimensional latent vectors. Finally, Figure 6b provides users with a Timbral representation of the latent space (*depth* is shown), allowing users to generate patches that closely match their desired timbral properties for a sound. This representation is derived by uniformly sampling across both the X and Y components of the latent space, generating patches for each coordinate, recording a short audio clip of the associated sound played through the synthesizer, and running timbral analysis once again using the AudioCommons toolkit. A perceptually uniform colormap is used in order to show that darker sections of the representation are associated with lower measured values of the timbral attributes and brighter sections are conversely associated with higher measured timbral values. Figure 8 depicts the same representation for each of the other timbral qualities. Each of these representations of synthesizer patches can be used as deemed appropriate by musicians when they are searching for or discovering new sounds.

## V. Conclusion & Future Work

This paper introduces our method for generating synthesizer patches using a Variational Autoencoder. By training a VAE on an existing library of patches created by amSynth users, we compress the data into a latent space with two dimensions. We observe that the VAE is indeed able to learn a latent space that effectively reconstructs sample patches, albeit in a lossy way. We also found that exploring the latent space through interpolation facilitated the generation of new synthesizer patches with musically interesting timbres. Audio samples of the patches generated via the VAE as well as the source code for this project (including ML models, interpolation demos, and visualizations) are available at https://github.com/peacheym/LatentRepresentations. We encourage readers to listen to the sounds associated with the generated synthesizer patches.

Future work for this project includes exploring and comparing the use of different ML architectures and parameters (latent dimensions, loss functions, disentangled VAEs, etc.) in order to maximize effectiveness of our ML model. Furthermore, user-centric interfaces for generating synthesizer patches using our latent representations will continue to be developed in order to provide an alternative interaction technique for musicians and other synthesizer users working creatively. These interfaces will also be evaluated with a user study as described in Section IV.B in order to further validate this research.

## References

[1] Russell Mason Andy Pearce Tim Brookes. "Hierarchical ontology of timbral semantic descriptors," in: *Audio-Commons - Deliverable D5.1* (2016), pp. 1–34.

[2] Ke Chen et al. "The effect of explicit structure encoding of deep neural networks for symbolic music generation". In: *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE. 2019, pp. 77–84.

[3] Giovanni De Poli. “A tutorial on digital sound synthesis techniques”. In: *Computer Music Journal* 7.4 (1983), pp. 8–26.

[4] Nick Dowell. *amSynth*. https://github.com/amsynth/amsynth. 2022.

[5] Jesse Engel et al. “GANSynth: Adversarial Neural Audio Synthesis”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=H1xQVn09FX.

[6] Jesse Engel et al. “Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1068–1077. URL: https://proceedings.mlr.press/v70/engel17a.html.

[7] David Foster. *Generative deep learning*. ”O’Reilly Media, Inc.”, 2022.

[8] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[9] GM Harshvardhan et al. “A comprehensive survey and analysis of generative models in machine learning”. In: *Computer Science Review* 38 (2020), p. 100285.

[10] Antti Huovilainen and Vesa Välimäki. “New Approaches to Digital Subtractive Synthesis”. en. In: *Proc. Int. Computer Music Conf. (ICMC’05)* (2005), p. 5.

[11] Taeho Jo. *Machine Learning Foundations*. Springer, 2021.

[12] Diederik P Kingma, Max Welling, et al. “An introduction to variational autoencoders”. In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[13] Michael A. Lones. *How to avoid machine learning pitfalls: a guide for academic researchers*. 2021. DOI: 10.48550/ARXIV.2108.02497. URL: https://arxiv.org/abs/2108.02497.

[14] Sean Luke. “Stochastic synthesizer patch exploration in edisyn”. In: *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer. 2019, pp. 188–200.

[15] Brian D Mayton et al. “Patchwork: Multi-User Network Control of a Massive Modular Synthesizer.” In: *NIME*. 2012.

[16] Ashis Pati, Alexander Lerch, and Gaëtan Hadjeres. “Learning to Traverse Latent Spaces for Musical Score Inpainting”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference* (Delft, The Netherlands). Delft, The Netherlands: ISMIR, Nov. 2019, pp. 343–351. DOI: 10.5281/zenodo.3527814. URL: https://doi.org/10.5281/zenodo.3527814.

[17] Adam Roberts et al. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4364–4373. URL: https://proceedings.mlr.press/v80/roberts18a.html.

[18] Fanny Roche et al. “Make That Sound More Metallic: Towards a Perceptually Relevant Control of the Timbre of Synthesizer Sounds Using a Variational Autoencoder”. In: *Transactions of the International Society for Music Information Retrieval* (May 2021). DOI: 10.5334/tismir.76.

[19] Kıvanç Tatar, Matthieu Macret, and Philippe Pasquier. “Automatic synthesizer preset generation with presetgen”. In: *Journal of New Music Research* 45.2 (2016), pp. 124–144.

[20] Gwendal Le Vaillant, Thierry Dutoit, and Sebastien Dekeyser. “Improving Synthesizer Programming From Variational Autoencoders Latent Space”. In: *2021 24th International Conference on Digital Audio Effects (DAFx)*. 2021, pp. 276–283. DOI: 10.23919/DAFx51585.2021.9768218.

[21] Gabriel Vigliensoni, Louis McCallum, and Rebecca Fiebrink. “Creating latent spaces for modern music genre rhythms using minimal training data”. In: *International Conference on Computational Creativity (ICCC’20)* (2020).

[22] David L Wessel. “Timbre space as a musical control structure”. In: *Computer music journal* (1979), pp. 45–52.