



# Self-supervised Audio-Reactive Music Video Synthesis

Measuring and optimizing audiovisual correlation

Hans Brouwer



# Self-supervised Audio-Reactive Music Video Synthesis

Measuring and optimizing audiovisual correlation

by

Hans Brouwer

to obtain the degree of Master of Science  
at the Delft University of Technology,  
to be defended publicly on Thursday June 29, 2021 at 10:00 AM.

Student number: 4376625  
Project duration: November 17, 2021 – June 29, 2022  
Thesis committee: Dr. L. Y. Chen, TU Delft, supervisor  
Dr. C. C. S. Liem MMus, TU Delft, supervisor  
Dr. M. Weinmann, TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

# Summary

Synthesizing audio-reactive videos to accompany music is a challenging multi-domain task that requires both a visual synthesis skill set and an understanding of musical information extraction. In recent years a new flexible class of visual synthesis methods has gained popularity: generative adversarial networks. These deep neural networks can be trained to reproduce arbitrary images based on a dataset of about 10000 examples. After training, they can be harnessed to synthesize audio-reactive videos by constructing sequences of inputs based on musical information.

Current approaches suffer from a few problems which hamper the quality and usability of GAN-based audio-reactive video synthesis. Some approaches consider only a few possible musical inputs and ways of mapping these to the GAN's parameters. This leads to weak audio-reactivity which has a similar motion characteristic across all musical inputs. Other approaches do harness the full design space, but are difficult to configure correctly for effective results.

This thesis aims to address the trade-off between audio-reactive flexibility and ease of attaining effective results. We introduce multiple algorithms that explore the design space by using machine learning to generate sequences of inputs for the GAN.

To develop these machine learning algorithms, we first introduce a metric, the audiovisual correlation, that measures the audio-reactivity in a video. We use this metric to train models based only on a dataset of audio examples, avoiding the need of a large dataset of example audio-reactive videos. This self-supervised approach can even be extended to optimize a single audio-reactive video directly, removing the need to even train a model beforehand.

Our evaluation of the methods shows that our algorithms out-perform prior work in terms of their audio-reactivity. Our solutions explore a wider range of the audio-reactive space and do so without the need for manual feature extraction or configuration.

# Contents

<b>Summary</b>	<b>i</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>3</b>
2.1 Generative Adversarial Networks . . . . .	3
2.2 Audio-reactive Latent Interpolations . . . . .	3
2.2.1 Previous Work . . . . .	4
2.2.2 StyleGAN Specifics . . . . .	4
<b>3 Audiovisual Correlation</b>	<b>7</b>
3.1 Audio Features . . . . .	8
3.2 Video Features . . . . .	10
3.3 Matrix Correlations . . . . .	12
3.3.1 (Adjusted) RV Coefficient . . . . .	12
3.3.2 Canonical Correlation Analysis . . . . .	13
3.3.3 Orthogonal Procrustes Correlation . . . . .	14
3.3.4 Similarity of Matrices Index . . . . .	15
3.4 Empirical Analysis and Validation . . . . .	16
3.4.1 Experimental Setup . . . . .	16
3.4.2 Selecting A Correlation Metric . . . . .	17
3.4.3 Comparing Interpolation Groups . . . . .	19
3.4.4 Smoothness Bias . . . . .	20
<b>4 Video Synthesis</b>	<b>22</b>
4.1 Randomizer . . . . .	23
4.1.1 Latent Patches . . . . .	24
4.1.2 Noise Patches . . . . .	25
4.2 Supervised and Self-Supervised . . . . .	26
4.2.1 Model Architecture . . . . .	27
4.2.2 Training . . . . .	28
4.3 HiPPO . . . . .	30
4.3.1 HiPPO Parameterization . . . . .	31
<b>5 Experiments</b>	<b>33</b>
5.1 Frameworks . . . . .	33
5.2 Dataset . . . . .	33
5.3 Evaluation . . . . .	34
5.3.1 Audiovisual Correlation . . . . .	34
5.3.2 Fréchet SwAV Distance . . . . .	34
5.3.3 Inference Time . . . . .	35
5.4 Baselines . . . . .	35
5.5 Results . . . . .	35
5.5.1 Audiovisual Correlation . . . . .	35



---

5.5.2	Fréchet SwAV Distance . . . . .	36
5.5.3	Inference Time . . . . .	38
5.5.4	Supervised vs. Self-Supervised . . . . .	39
5.5.5	HiPPO . . . . .	41
<b>6</b>	<b>Conclusion</b>	<b>43</b>
6.1	Summary of the Contributions . . . . .	43
6.2	Future Work & Recommendations . . . . .	44
6.3	Human-in-the-loop Workflow . . . . .	45
6.4	Final Thoughts . . . . .	45
	<b>References</b>	<b>47</b>
<b>A</b>	<b>Model Details</b>	<b>53</b>
A.1	Learned Decoder . . . . .	53
A.2	Parameter-free Decoder . . . . .	53
A.3	Envelope Generator . . . . .	53
A.4	HiPPO . . . . .	54
<b>B</b>	<b>Negative / Inconclusive Results</b>	<b>55</b>

# 1

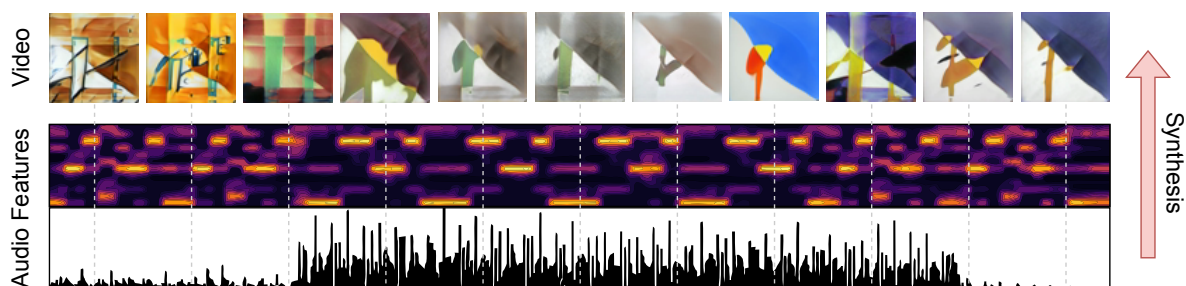
## Introduction

Audio-reactivity has fascinated humans since pre-historic times. First in physical form, starting with dance, and growing alongside us as our technology has improved. The invention of computers has enabled ever more complex audio-reactive visuals. The seminal open-source work of Ryan M. Geiss on the WinAmp plugin MilkDrop [1] popularized and illustrated the great promise of audio-reactive music videos.

The general principle of an audio-reactive music video is to drive elements of the video with elements in the music. There is a vast range of systems which have been controlled by music, including 3D renders, fractals, analog video synthesizers, robots, and many other custom-designed visuals. Whatever the visual system, the process is the same: find some set of parameters that control the image and vary them over time according to signals extracted from the audio.

Designing such audio-reactive visual systems requires domain-specific knowledge of visual synthesis programs (e.g. Blender [2], TouchDesigner [3], or Processing [4]) as well as an understanding of computational music analysis. Once a given audiovisual synthesizer is created, in theory it can be applied to any number of new music inputs. However, designing a system that is visually variable enough to remain interesting over time as well as reacting well to many kinds of audio is challenging. Creating such an extensive visual system and customizing how signals from the audio control different parameters of it is an enormous undertaking.

Recently a specific class of visual synthesis methods have shown great potential for flexible, expressive video generation: generative adversarial networks (GAN). These deep neural networks are trained to mimic a large dataset of images. Once trained, they can generate essentially infinite amounts of new images that resemble the dataset. These images can be strung together to form videos, which, when conditioning on music, seem to react and dance along with the song. This provides a promising basis for an audiovisual synthesis system as the network can be trained to generate any style for which a large



**Figure 1.1:** Example of an audio-reactive video. The frames are synthesized based on some audio features extracted from the music.



enough dataset of images can be gathered. This can essentially eliminate the need to develop the visual system manually, yet still allow for a vast range of visual styles.

While there are some approaches which generate audio-reactive videos with GANs, so far they are either not particularly flexible in terms of their audio-reactivity (i.e. always move in a fairly similar way) or require deep understanding of both GANs and music information retrieval to use effectively. Many approaches rely on only a few methods of extracting information from the music and only part of the GAN's full generative capacity. This leads to videos which only react to a narrow range of possible, perceptually-relevant aspects in the music. On the other hand, there are so many musical features that can be used and so many ways to map these to control the GAN that it can be overwhelming to thoroughly explore the entire space of possibilities. This highlights the need to find a middle path where these GAN-based audiovisual synthesizers are easy to use yet still sufficiently expressive. The range of possible outputs should be as varied as possible in terms of audio features that might drive the video's reaction, but the decision of which audio features and how they are mapped to the GAN should be made automatically.

The primary goal of this thesis is to propose **methods that automate the translation of audio features to audio-reactive GAN inputs**—making it easier to make expressive audiovisuals. We propose multiple algorithms that can use a GAN to generate audio-reactive videos and show that they outperform prior work in terms of their audio-reactivity. These algorithms use a wider range of audio inputs and GAN parameters which allow them to react to more possible elements of the music without the need for configuring the specifics manually.

## Research Questions

- Can we design a metric that can capture some notion of the audio-reactive quality of a video?
  - Can the metric distinguish between videos with perceptually different audio-reactive characteristics?
  - Does the metric provide useful supervision for machine learning approaches?
- Can we learn to translate a wide range of audio features to effective audio-reactive GAN input sequences?
  - Is this possible without a large dataset of audiovisual examples? Do we even need to train at all?
  - Can we design methods that have better audio-reactivity than prior work?
  - Do the methods outperform randomly mapping a wider range of audio features to GAN inputs?

# 2

## Background

Before covering our algorithms for audiovisual correlation and audio-reactive video synthesis, there is some important prerequisite knowledge. In this chapter we explain the foundations of the GAN-based audiovisual synthesis process. We specifically focus on the StyleGAN [5] architecture, which has multiple properties that make it well-suited for expressive audio-reactive video synthesis.

### 2.1. Generative Adversarial Networks

Generative adversarial networks [6] learn to synthesize realistic, high-definition images with only a random vector as input (which is generally drawn from a simple noise distribution like the standard Gaussian). Training entails two competing networks and a large dataset of images. The first model, the generator, learns to map random vectors to images and the second model, the discriminator, tries to classify images as real (from the training dataset) or fake (synthesized by the generator). The generator tries to optimize for the discriminator misclassifying its creations as real, while the discriminator tries to optimize for classifying correctly. This dynamic leads both networks to improve together until the generator can create images close to the true data distribution.

An important dynamic of training GANs is that generally the generator learns a continuous image manifold [7]–[9] such that smoothly interpolating the input vector will also smoothly interpolate the output image. Therefore, continuous videos can be generated by constructing a smooth interpolation between input vectors. Furthermore, a given random interpolation can be rendered in different styles by swapping out the generator for another trained on a different dataset. This means that strategies for generating input sequences developed for one style can be applied to a vast range of visual styles without needing to be customized to each one; as would be required if the visual system were crafted manually using, for example, 3D renders or software like Processing [4].

### 2.2. Audio-reactive Latent Interpolations

The key to making a latent interpolation audio-reactive is to extract information from the audio signal and use it to modify the inputs to the generator network over time. Possible audio features to use range from the volume of the music, to the notes that an instrument is playing, to the rhythms present in a section, to the timbre of the reverb, or anything in between. The design space is essentially infinite, so first we discuss the approaches that have been used so far and analyze their strengths and weaknesses. Then we look at how StyleGAN’s flexible control over image synthesis can increase the space of audio-reactive possibilities even further.



### 2.2.1. Previous Work

Robert Luxemburg [10] uses a moving average of an audio waveform directly to either adjust movement along a pre-determined smooth interpolation or blend between different fixed latent vectors. While this does create visual movement in the output that is correlated to the audio, it is often jittery and does not follow perceptually important aspects of the music. The audio waveform may be the representation that we actually hear, but it is not necessarily effective as a source of GAN input modulation as all the musical information is overlaid at once.

A more useful representation of the audio is its spectrogram, the short-time Fourier transform of the signal. This representation separates out the different frequencies present at any time in the signal, which gives information about the volume of different pitches in the audio. Matt Siegelman [11], [12] and Mikael Alafritz [13] propose systems that make use of this spectral information. Siegelman uses BigGAN [14] as the visual prior, which has both a latent vector for sampling output images and a class vector determining which class from ImageNet [15] to synthesize. The latent vector sequence is generated based on the mean value at each timestamp in the spectrogram as well as the local difference in frequencies (similar to onset detection). The class conditioning is generated based on the chromagram [16] (a remapping of frequencies in the spectrogram to bins corresponding to the notes in Western musical scales). Alafritz builds on this work by using StyleGAN [5] instead of BigGAN and by separating harmonic and rhythmic content before calculating the spectrograms.

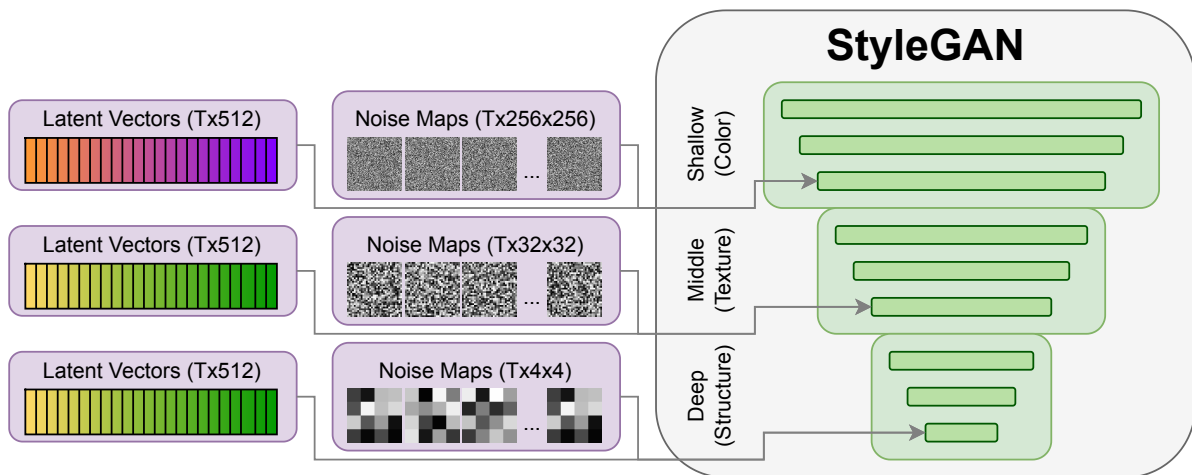
Employing spectral information to steer GANs significantly improves the audio-reactivity of these methods, however, they are hampered by the application of these signals to their visual synthesizers. Both Siegelman's and Alafritz's methods only make use of one single sequence of latent and class vectors each (when possible; some StyleGAN models do not even have class conditioning). This forces all audio information to affect the output video in the same way. In practice, when too many audio feature reactions are merged into the same latent sequence, the output is chaotic, and it becomes hard to link visual movement to the corresponding element in the music that caused it.

Towards increasing this "audio-reactive bandwidth" (the amount of visual changes that can be recognized as relating to an aural change), my previous work "Audio-reactive Latent Interpolations with StyleGAN" [17] investigated multiple ways of modulating the video output that could more clearly be distinguished from each other. These techniques allow for more expressive audio-reactive videos by layering multiple different visual reactions at once, similar to how multiple musical elements are usually layered within a song. These techniques rely on some specific features of the StyleGAN architecture.

### 2.2.2. StyleGAN Specifics

Karras et al. have published multiple iterations of the StyleGAN architecture [5], [18], [19]. For the purposes of audio-reactive video synthesis, StyleGAN 2 offers the most flexibility and so will be the focus of this section (the '2' will be left out for brevity). In theory, however, any architecture can be used as long as there are multiple different mechanisms that affect the output in visually distinct ways.

There are two major aspects of the StyleGAN architecture that allow for increasing the audio-reactive bandwidth: the hierarchical decomposition of the generator and the spatial noise maps. These two features offer ways of spreading musical information to affect only certain aspects of the generated images, rather than the entire frame all at once.



**Figure 2.1:** Simplified schematic representation of the StyleGAN architecture. Only three latent and noise sequences are shown, but each layer (dark green block) can accept a different pair of sequences. Each layer in the hierarchy contributes different aspects in the image, varying from large-scale structural components in the deepest layers to fine details in the shallowest ones.

**Hierarchical Generation** The first key aspect that makes StyleGAN a good visual prior for audio-reactive synthesis is its hierarchical generation style. The StyleGAN generator network consists of a sequence of convolutional blocks where each block upsamples the input by a factor of two and then applies two latent-modulated convolutions. The output of each layer is fed to the next layer in the hierarchy, but is also converted to RGB space by another modulated convolution. The RGB outputs of all layers are summed together to become the final output image. This strategy helps separate responsibility for different scales of components in the image to different layers in the network. The deepest layers influence large scale structures. In these layers the image is small and so each 'pixel' represents a large area in the final output. The middle layers control mid-level structures or textures, and the shallowest layers contribute fine details and colors.

To control the generation at each layer in the network, the convolutional blocks are modulated by a latent vector. This means that before applying the learned convolution operation to the input, the channel weights are first scaled by a learned, affine transformation of the latent vector. For a fixed input, varying the latent vector will result in a different "style" of output (hence StyleGAN). This allows for a technique which Karras et al. call "style mixing", where multiple differently colored images can be generated with the same structure (constant deep latents, varying shallow latents) or vice versa.

This is valuable in the context of audio-reactive latent interpolations as separate latent sequences can be used to change the structure versus the color of the video—more latent vector sequences, means more audio-reactive bandwidth.

**Noise Maps** The second important part of the StyleGAN architecture are the noise maps. After each convolution, random, normally-distributed noise is added to each pixel of the intermediate features. This helps to avoid the network wasting capacity modeling stochastic variation in the image. For example, given an image of trees, there are many arrangements of the leaves that are equally valid. Rather than have one of the latent dimensions represent an infinite amount of different configurations, the noise maps directly inject this randomness into the synthesis process. This hopefully frees the latent vector to model



semantic aspects like the visual difference between needles, leaves, or blossoms. This is important as it separates an audio-reactive interpolation into a semantic and stochastic stream of information. These two can be modulated by different audio features and still be visually distinct.

This provides another tool to influence the output video which also enjoys the same hierarchical scale properties as latent vectors (due to noise being inserted after each layer). Further, as the noise has the same spatial dimensions as the intermediate features, the effects can be localized to certain parts of the image as well. This allows for different audio features to affect different parts of the image yet still be recognizable, even within the same noise scale.

Brouwer [17] covers a few more possible advanced audio-reactive modulation targets (Network Bending [20] and Model Rewriting [21]), but these are not investigated further in this thesis. However, the algorithms presented later can support these extra targets as well. Figure 2.1 shows an overview of how each of these control points come together in the full model.

# 3

## Audiovisual Correlation

At the time of writing, research into synthesizing audio-reactive videos with StyleGAN has largely been led by qualitative considerations. Before we can teach a machine to automatically generate audio-reactive videos, we need to quantify what makes for effective audio-reactivity. In this chapter we present a metric for audiovisual correlation based on the informal "audio-reactive bandwidth" mentioned previously. The motivating assumption is that the perceptual concept of audio-reactivity stems from patterns in the audio being matched with patterns in the video. In other words, the more separate pairs of audiovisual changes that can be individually identified, the stronger the audio-reactivity of the video feels. For the clarity of the audio reactions, it is important that separate audiovisual pairs modulate separate parts of StyleGAN's semantic, stochastic, or spatial hierarchy—overloading one of these visual control channels muddies the signals carried over them.

There has been much research into audiovisual correlations for many problems such as video segmentation, audiovisual alignment, lip-reading, or information retrieval. However, the task of synthesizing audio-reactive videos to accompany music is unique in its subjective, creative nature. There is no clearly-defined, universally-accepted success metric like other tasks. Therefore, we argue that it makes sense to design a correlation metric which is flexible and can be adapted to different concepts of audio-reactivity.

To this end, we use a feature extraction approach where the correlation metric itself is generic to the exact set of audio and video features that is used. This way, the correlation can be adapted to different contexts by selecting different audio and video feature pairings that are important for the desired audio-reactivity.

In the following sections we introduce and motivate the set of audio and video features we use. Then, we discuss some candidate correlation metrics that can measure the correspondence between the audio and video features. Finally, we perform experiments to evaluate the different correlation metrics and choose one to use in the video synthesis algorithms.



## 3.1. Audio Features

The audio features are selected to capture a range of different possible aspects in the music. We seek to capture four main classes of musical information: rhythmic, note-based, timbral, and sectional.

**Rhythmic** The rhythmic features capture transient aspects of the music. Things like drums or the spacing between notes. There are a few features which deal with this, most importantly the onset envelope. The onsets measure the spectral flux in the frequency spectrum over time. They give a high value when there are sudden changes such as a drum hit. Other features that capture rhythmic information include the RMS and the predominant local pulse (PLP). The RMS measures the overall energy in the frequency spectrum (i.e. the volume). The PLP is an envelope that tries to follow the strongest repeating pulse in music based on the STFT of the onset envelope. This captures the local tempo of the song.

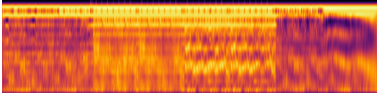
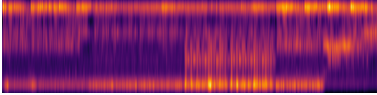
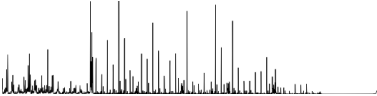
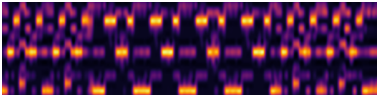
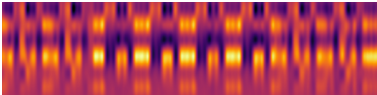
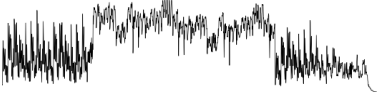

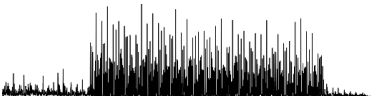


**Note-based** The note-based features are the chromagram and the tonnetz. The chromagram measures the energy in the frequencies corresponding to each of the 12 different notes in the Western musical scale. This is a feature which takes different values whenever different notes are played. The tonnetz is similar but maps these frequencies into features corresponding to the perfect fifth, minor third, and major third of the scale. In music theory, these are important ratios between notes and the root of the scale which often hold emotional connotations. The tonnetz can capture changes in these emotionally-linked aspects of the music.

**Timbral** The timbre of a sound its tonal quality. For example, even when playing the same note a violin and saxophone have a different sound—they have a different timbre. This is an especially nebulous aspect of the music which may or may not need to play a role in the video's movement. The features which capture timbre are thus also most general and closest to the raw spectral representation. These features are the Mel-frequency cepstral coefficients (MFCC), spectral contrast, and spectral flatness. The MFCCs are a modified version of the simple spectrogram, re-weighted according to perceptually relevance for the human auditory system. The spectral contrast and flatness characterize the dynamic range of different frequency bands and the noisiness of the overall spectrum.

**Sectional** The final features seek to capture longer-range information in the music. The Laplacian segmentation classifies sections of the song into different groups. This can recognize, for example, an ABABCD structure in a song, where certain phrases are repeated multiple times. The drop weight is a long-term moving average of the RMS value which is re-scaled to emphasize soft and loud sections. These sectional features can allow for a video to have different qualities in different sections of the song.

While the audio features used is not an exhaustive list of possible important features, they do capture a range of important information that intuitively seem like reasonable candidates for a video to react to. A full explanation and visualization of each audio feature can be found in Table 3.1.

**Table 3.1:** Descriptions of the audio features considered. Visualizations show the time on the x-axis. Feature values are represented by the color when features are multidimensional or otherwise plotted on the y-axis.

Name	Size	Description	Visualization
Mel Frequency Cepstral Coefficients	20	A spectral representation of audio in the Mel frequency scale (which models human hearing). Represents timbre of the audio signal.	
Spectral Contrast	7	Difference between the highest energy quantile and lowest energy quantile in each band of the frequency spectrum. Represents dynamic range of each frequency band.	
Spectral Flatness	1	Tonality coefficient. Represents how close to white noise the signal is. Lower values are closer to a pure sine, high values closer to pure white noise.	
Chromagram	12	Energy in bands of the frequency spectrum corresponding to the 12 notes of the western scale. Represents which notes are being played.	
Tonnetz	6	Tonal centroids. Remapping of the chromagram into two-dimensional features corresponding to the perfect fifth, minor third, and major third. Captures musical mode information, which can be seen as a weak proxy for emotion; the major mode tends to be perceived as happy while the minor mode sounds sad.	
RMS	1	Total power in the audio signal over time. Represents the volume of the audio signal.	
Drop Weight	1	Long-term moving average of the RMS, soft-clipped to be approximately zero when below average power and approximately one when above. Loosely segments audio signal into loud and quiet sections.	
Onsets	1	Spectral flux in Mel frequencies of the rhythmic component of the audio. Represents timing of transients in the audio signal (e.g. drums).	
Predominant Local Pulse	1	Pulse synchronized to the predominant frequency in the onset envelope's spectrum. Represents the local musical tempo.	
Laplacian Segmentation	K	Recurrence-based segmentation of the audio signal into K sections. Captures repetition and long-timescale patterns in the audio signal.	

## 3.2. Video Features

The video features are selected based on an understanding of the type of video interpolations that StyleGAN can generate. Audio-reactive latent interpolations allow for color control through use of the latent sequence and structural control at different scales in the image through the noise maps. This motivates us to choose features which can recognize the changes in colors over time as well as features that can quantify changes in the video across different spatial scales.

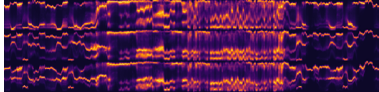
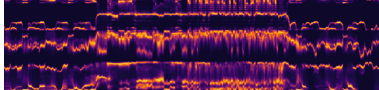
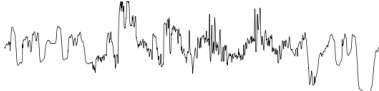
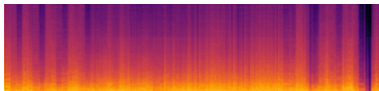

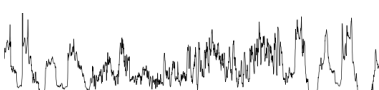

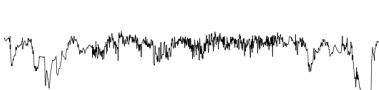

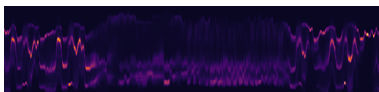


To quantify the color over time, we can analyze the distribution of pixel values directly. Each frame can be decomposed into a histogram of its pixel values to capture changes in color over time. On top of the RGB values, we also transform the video to HSV space (hue, saturation, value) and calculate a similar histogram over time feature.

To capture the structural changes is a little less straightforward. One notion of structure is the amount of variation in the image. A simple way of calculating this is by taking the variance of each frame. Another way of looking at this structure is through the lens of the Fourier transform. This decomposes the image into frequencies and their magnitudes. This allows us to understand the presence of different scales of structure by looking at the amplitudes of groups of frequencies in the images. In fact, this allows us to create a spectrogram of frequencies in the video over time similar to spectrograms of audio. An analog of the onset envelope can also be calculated for this video spectrogram.

Another important aspect of videos is captured by the optical flow (i.e. how pixels in the video move over time). This can similarly be converted into a spectrum of different bands each corresponding to the magnitude of movement in different directions called the Directogram Davis and Agrawala [22]. Just as with the other spectrograms, the spectral flux can be taken to get an onset envelope, this one tailored to motion in the video.

Altogether these features capture several important aspects of video: the color, structure, and movement. Just as with the audio features, this is not an exhaustive list of possible features, but a decent baseline set. If there are other visual effects are important for the audiovisual correlation, other features can be added to the set. Table 3.2 contains a list of all the video features used.

**Table 3.2:** Descriptions of the video features used. Visualizations show the time on the x-axis. Feature values are represented by the color when features are multidimensional or otherwise plotted on the y-axis.

Name	Size	Description	Visualization
RGB Histogram	96	32-bin histogram of each color channel of each video frame. Captures change in color distribution.	
HSV Histogram	96	32-bin histogram of hue, saturation, and value of each video frame. Captures change in color distribution.	
Visual Variance	1	Variance of each frame's pixel values. Captures overall noise scale in the frames of the video.	
Video Spectrogram	32	Fourier transform of each frame, transformed to polar coordinates, and averaged over theta. Captures a spectrum of the strength of spatial frequencies in the images, ranging from low-frequency, long range components to high-frequency, fine details.	
Low Frequency RMS	1	The root-mean-square (volume) of the lower third of the video spectrogram. Captures the strength of large-scale structural components in the image.	
Mid Frequency RMS	1	The root-mean-square (volume) of the middle third of the video spectrogram. Captures the strength of medium-scale textural components in the image.	
High Frequency RMS	1	The root-mean-square (volume) of the upper third of the video spectrogram. Captures the strength of small-scale details in the image.	
Adaptive Frequency RMS	1	The root-mean-square (volume) of the bands in the video spectrogram with most variance over time. Captures the strength of components in image that vary most throughout the video.	
Absolute Difference	1	The absolute value of the frame-by-frame difference. Captures sudden changes in the video.	
Directogram	16	A spectrogram based on the optical flow in the video. Each band represents the average strength of movement in 16 directions spaced equally in 360 degrees.	
Video Spectral Onsets	1	Spectral flux onset envelope based on the Video Spectrogram. Captures rhythmic elements in the "noisiness" of the video.	
Video Flow Onsets	1	Spectral flux onset envelope based on the Directogram. Captures rhythmic elements of movement in the video.	

### 3.3. Matrix Correlations

Now that we have a set of features that can capture a broad range of components of audio and video, we turn to correlation metrics that can capture the similarity between these features. In the case of simple, univariate features like the audio or video onset envelopes, the correlation can be calculated directly. However, for multivariate features like the chromagram or visual spectrogram, the best method is less clear.

One of the simplest ways of addressing the aforementioned problems is to instead consider the autocorrelation of each feature. The autocorrelation takes the outer product of the feature with itself, which results in a square matrix with the length of the sequence along each side. These autocorrelation matrices can then be compared as long as the sequence length for all audio and video features is the same. The intuition is that two features which have a high correlation will be self-similar in the same places and so have highly correlated autocorrelations.

With this idea as a guide, we survey a number of candidate matrix correlation metrics. In Section 3.4 we compare their performance and select one based on a preliminary study. While the approaches do not all directly use the autocorrelation of features, they all allow for comparisons between multivariate time series that do not necessarily share the same feature space. Crucially, the following metrics do assume that each feature corresponds to the same underlying observations seen through different lenses (i.e. that the compared sequences of audio and video features are aligned and of the same length).

#### 3.3.1. (Adjusted) RV Coefficient

Of the audiovisual correlation metrics we discuss, the RV coefficient [23] is closest to the intuition that two features which have a high correlation will be self-similar in the same places and so have highly correlated autocorrelations. The RV coefficient is the multivariate generalization of the Pearson correlation. The main difficulty in the multivariate case is that distances in one feature space may not correspond to distances in the other. Therefore, we instead consider the distance between the autocorrelations of each matrix  $S_X = XX^T$ . To ensure that the global scale is the same regardless of  $X$  we normalize by a factor of  $\sqrt{\text{tr}(S_X^2)}$  ( $\text{tr}(\cdot)$  being the matrix trace, the sum of diagonal values). This ensures the L2-norm of the normalized autocorrelation,  $\hat{S}_X$ , is always 1. Given two matrices  $A$  and  $B$ , we can find the distance between their transformed representations

$$\begin{aligned}
 d(A, B) &= \left\| \frac{S_A}{\sqrt{\text{tr}(S_A^2)}} - \frac{S_B}{\sqrt{\text{tr}(S_B^2)}} \right\| \\
 d(A, B) &= \|\hat{S}_A - \hat{S}_B\| \\
 d(A, B) &= \sqrt{\|\hat{S}_A\|^2 + \|\hat{S}_B\|^2 - 2\hat{S}_A^T \hat{S}_B} \\
 d(A, B) &= \sqrt{1^2 + 1^2 - 2\hat{S}_A^T \hat{S}_B} \\
 d(A, B) &= \sqrt{2 - 2 \frac{S_A^T S_B}{\sqrt{\text{tr}(S_A^2) \text{tr}(S_B^2)}}} \\
 d(A, B) &= \sqrt{2 - 2 \frac{\text{tr}(S_A^T S_B)}{\sqrt{\text{tr}(S_A^2) \text{tr}(S_B^2)}}}
 \end{aligned}$$



Neglecting the square root and the scaling factor, we see that  $d(A, B) \approx 1 - \rho(A, B)$ . This correlation-like function  $\rho(A, B)$  is the RV coefficient.

$$\rho_{rv} = \frac{\text{tr}(A^T A B B^T)}{\sqrt{\text{tr}(A A^T)^2 \text{tr}(B B^T)^2}}$$

The metric varies between 0 and 1 with values closer to 1 representing matrices with high correlation and values closer to 0 representing matrices with low correlation.

One issue with the RV coefficient, as investigated by Smilde, Kiers, Bijlsma, *et al.* [24], is that the RV coefficient does not actually go to zero when purely random numbers are used. This is undesirable as it implies correlation between matrices which have none. The cause becomes clear by analyzing the form of the RV coefficient. The diagonal values of the  $S_X$  matrices will always be positive. This means that the numerator of the correlation will, on average, be greater than zero for random values due to the small correlation between the positive diagonals of  $S_A$  and  $S_B$ . As a solution, Smilde, Kiers, Bijlsma, *et al.* [24] propose the adjusted RV coefficient, in which the RV coefficient is defined in terms of  $\tilde{S}_X = X X^T - \text{diag}(X X^T)$  instead.

This adjusted RV coefficient (ARV) does have slightly different properties though. It varies between -1 and 1 instead of 0 and 1. This brings its interpretation even closer to the Pearson correlation, allowing for measuring anti-correlation. In the audio-reactive context, it is not important whether something is correlated or anti-correlated, so we take the absolute value of the ARV as the correlation metric.

$$\rho_{arv} = \left| \frac{\text{tr}(\tilde{S}_A \tilde{S}_B)}{\sqrt{\text{tr} \tilde{S}_A^2 \text{tr} \tilde{S}_B^2}} \right| \quad (3.1)$$

### 3.3.2. Canonical Correlation Analysis

Canonical correlation analysis (CCA) is a statistical tool that analyzes two sets of multi-variate observations. It is conceptually similar to least squares regression, which finds a linear combination of a set of observations that maximizes the correlation with a given set of outcomes. CCA, on the other hand, finds linear combinations of both sets of observations that maximize the correlations between the transformed representations.

More formally, given two matrices  $A \in \mathbb{R}^{n \times a}$  and  $B \in \mathbb{R}^{n \times b}$ , the task is to find vectors  $u$  and  $v$  which maximize the correlation:

$$\rho = \frac{u^T A \cdot v^T B}{\|u^T A\| \|v^T B\|}$$

This can be rewritten in terms of the covariances and cross-covariance of the matrices:

$$\rho = \frac{u^T \Sigma_{A,B} v}{\sqrt{u^T \Sigma_A u} \sqrt{v^T \Sigma_B v}}$$

With a clever substitution for  $u$  and  $v$ , the expression can be solved with a singular value decomposition. Let  $u = \Sigma_A^{-1/2} \hat{u}$  and  $v = \Sigma_B^{-1/2} \hat{v}$ , then the above equation becomes:

$$\rho = \frac{\hat{u}^T \Sigma_A^{-1/2} \Sigma_{A,B} \Sigma_B^{-1/2} \hat{v}}{\sqrt{\hat{u}^T \hat{u}} \sqrt{\hat{v}^T \hat{v}}}$$

Which is equivalent to the first singular value of

$$\Sigma_A^{-1/2} \Sigma_{A,B} \Sigma_B^{-1/2} = \hat{U} \Lambda \hat{V}$$

The full singular value decomposition yields  $n$  different  $(u, \rho, v)$  triplets each representing the maximal correlation under a different orthogonal linear combination of  $A$  and  $B$ . This is a rich representation of the similarity between  $A$  and  $B$  under many projections, however, it is not immediately obvious how to combine the series of correlation values to a single score.

Morcos, Raghu, and Bengio [25] investigate series of CCA triplets in the context of analyzing deep network representations. They find that many of the triplets are not consistent over the course of training (especially early on) and so correspond to spurious, noisy correlations rather than meaningful ones. Therefore, they propose to re-weight the  $n$  correlations by the proportion of the input matrices they account for,  $\alpha$ :

$$\alpha_i = \sum_{j=0}^a |(\hat{U}_i^T A_i) A_j|$$

Re-weighting the singular values from above with the  $\alpha$  values gives the final projection-weighted canonical correlation coefficient:

$$\rho_{pwcca} = \frac{\sum_{i=0}^n \alpha_i \Lambda_i}{\sum_{i=0}^n \alpha_i} \quad (3.2)$$

The re-weighting scheme ensures that correlations between linear projections which retain the most information of the original signals are given the most importance. This results in a matrix correlation score which adapts the transformations of both matrices such that they are maximally correlated yet still least altered by the transformation.

### 3.3.3. Orthogonal Procrustes Correlation

The orthogonal Procrustes problem is a classic approximation task in linear algebra. The goal is to find an orthogonal matrix,  $\Omega$ , that maps a matrix  $A$  as closely as possible to  $B$ . This has applications in many fields where data points need to be remapped to match through translation, rotation, and scaling.

The correlation between the two matrices after the transformation has been applied provides a better understanding of the true correlation between the matrices.

Schönemann [26] describe a solution for the orthogonal Procrustes problem, showing that finding

$$\arg \min_{\Omega} \|\Omega A - B\|_F^2$$

is equivalent to finding a matrix  $R$ , such that

$$\min_R \|R - BA^T\|_F$$

This can be solved using the singular value decomposition

$$R = UV^T \quad \text{with} \quad BA^T = U\Lambda V^T$$

Plugging  $R$  into the distance, we can find an expression that directly gives the distance

based on  $A$  and  $B$ :

$$\begin{aligned}
d_{op} &= \|RA - B\|_F^2 \\
d_{op} &= \|RA\|_F^2 + \|B\|_F^2 - 2\langle RA, B \rangle_F \\
d_{op} &= \|A\|_F^2 + \|B\|_F^2 - 2\langle UV^T A, B \rangle_F \\
d_{op} &= \|A\|_F^2 + \|B\|_F^2 - 2\text{tr}(U^T A^T B V) \\
d_{op} &= \|A\|_F^2 + \|B\|_F^2 - 2 \sum_i \Lambda_i \quad (\text{because } BA^T = U\Lambda V^T) \\
d_{op} &= \|A\|_F^2 + \|B\|_F^2 - 2\|A^T B\|_*
\end{aligned}$$

where  $\|\cdot\|_*$  is the nuclear norm and  $\langle \cdot \rangle_F$  the Frobenius inner product. Given that  $A$  and  $B$  are normalized, this distance is between 0 and 2 [27]. Therefore, we can define the orthogonal Procrustes correlation  $\rho_{op}$  as:

$$\rho_{op} = 1 - (\|A\|_F^2 + \|B\|_F^2 - 2\|A^T B\|_*)/2 \quad (3.3)$$

Similar to  $\rho_{pwcca}$ ,  $\rho_{op}$  can be seen as a two-step process where first the matrices are transformed to be as similar as possible (in a least-squares sense) and then a correlation is taken between the transformed representations.

### 3.3.4. Similarity of Matrices Index

The Similarity of Matrices Index (SMI) [28] is a framework which tries to generalize the two-step process seen in the  $\rho_{pwcca}$  and  $\rho_{op}$  metrics. SMI first extracts stable subspaces (e.g. through PCA or CCA) to find some orthonormal bases to compare under. Then, these canonical representations are compared using orthogonal projection or Procrustes rotation.

For the purposes of designing our audiovisual correlation metric, we consider extracting subspaces with PCA and comparing representations with orthogonal projection as the other cases are very similar to the metrics proposed above. In this case, the procedure for calculating the SMI begins by taking the singular value decomposition of both matrices  $A$  and  $B$ :

$$U_A \Lambda_A V_A = A$$

$$U_B \Lambda_B V_B = B$$

The rank  $r$  of the matrices is estimated by counting the singular values larger than a tolerance value and the corresponding left singular vectors taken as the reduced-rank basis:

$$r = \min(r_A, r_B)$$

$$\hat{U}_B = U_B^{(0:r)}$$

$$\hat{U}_A = U_A^{(0:r)}$$

The SMI  $\rho_{smi}$  is then calculated by taking the average of the squared singular values  $S$  of the cross-correlation of the reduced-rank bases:

$$S = \hat{U}_A^T \hat{U}_B = U_S \Lambda_S V_S$$

$$\rho_{smi} = \sum_i^r (\Lambda_S^{(i)})^2 / r \quad (3.4)$$

## 3.4. Empirical Analysis and Validation

To decide which of the matrix correlations is best for our audiovisual correlation, we performed a preliminary study on synthetic groups of audio-reactive interpolations. Here we validate whether our feature-based correlation approach can measure the difference between videos with different levels of audio-reactivity.

First, we explain the setup of the experiment and the groups of audio-reactive videos that we compare. Then we compare the different metric values to determine which one has the most desirable properties and then evaluate its ability to distinguish between our video groups.

### 3.4.1. Experimental Setup

These groups are designed to have high correlations in some circumstances and low correlations in others. We select the matrix correlation which most closely matches these expectations.

We propose two ways of calculating correlations. First, the concatenated correlation, which concatenates all audio features together, and all video features together before taking a single correlation between them. This represents the overall audiovisual correlation of the video. Secondly, we consider individual pairwise correlations between audio and video features. This gives a more fine-grained understanding of which audio features are more strongly represented in the video. Ideally the concatenated and pairwise correlation values should be comparable with each other for easier analysis.

The synthetic groups of audio-reactive interpolations are generated by manually converting audio features to latent interpolations using the methods described in Section 2.2. Six different interpolation groups are made with 500, 16-second video snippets in each. The audio for each snippet is chosen randomly from a corpus of about 30 minutes of music from the test set (further details in Section 5.2). Two groups are based on random interpolations and four groups are audio-reactive.

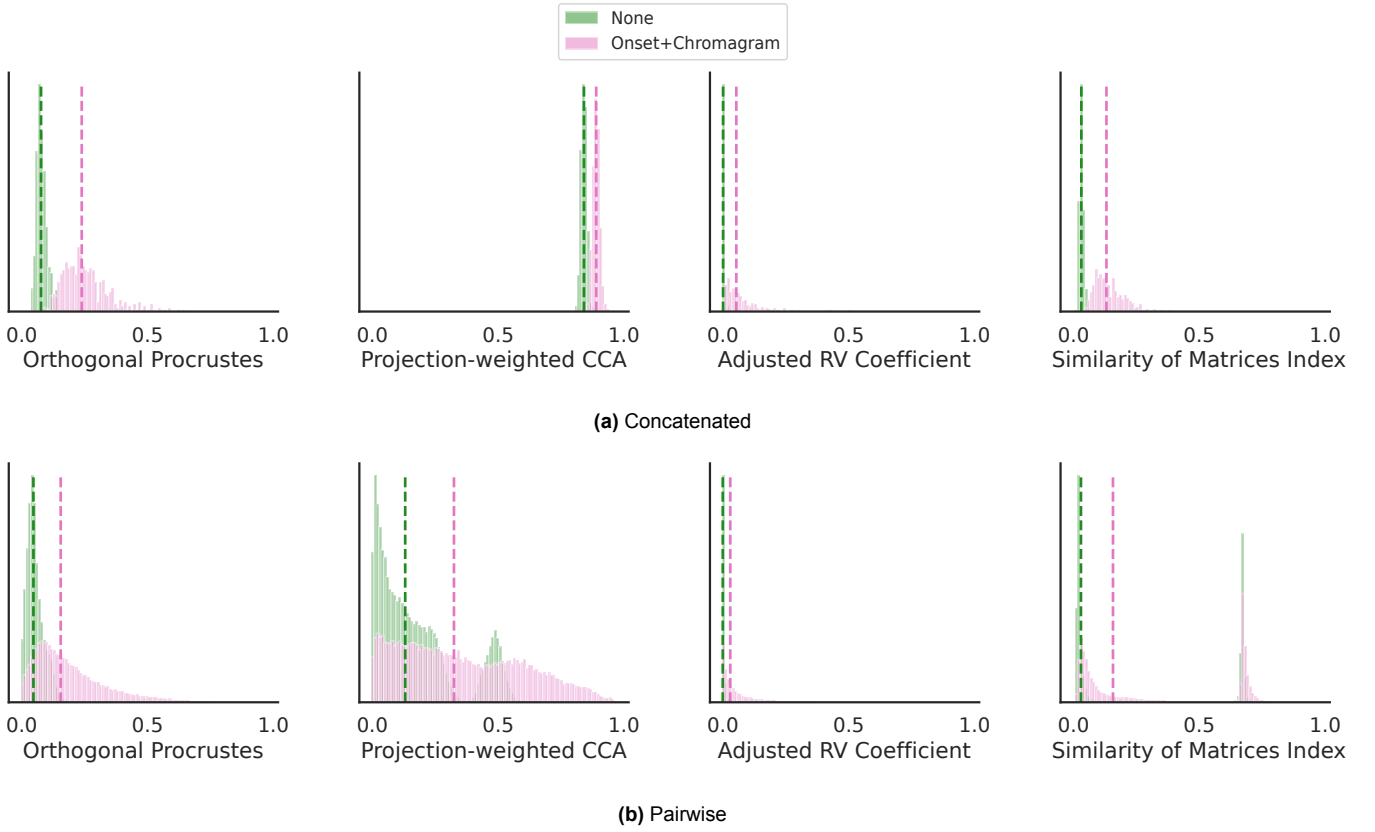
#### Interpolation Groups

- **None:** control group that did not interpolate at all. Each frame is generated with a random Z-space latent vector and random noise maps. This group should have the lowest correlation with all audio features as it is purely random.
- **Random:** second control group with random interpolations. The latent vectors and noise maps are smoothed over time with a Gaussian filter. These random interpolations should also have low correlation as they are not actually reacting to the audio.
- **Onsets:** onset-weighted interpolations. The latent vectors bounce between two random vectors based on the value of the onset strength. Noise maps for this group are initialized as random smoothed noise interpolations and then multiplied by the onset strength mapped to vary between 0.5 and 1.5. The effect of this modulation is that the standard deviation of the noise is higher when an onset occurs and lower when there is no onset. This group should have high correlation between the onset strength and video features (and therefore should also score better than **None** or **Random** on concatenated correlation).
- **Chroma:** chromagram-weighted interpolations. The latent vectors are the weighted average of 12 random vectors, each corresponding to one of the bins in the chromagram. The noise maps are likewise a weighted average of 12 randomly drawn noise matrices. This group should have high correlation between the chromagram and video features.

- **Onset+Chroma:** Each latent and noise sequence the average of the Onset and Chroma group procedures. This group should have high correlation between the video features and both the onsets and chromagram.
- **Manual:** hand-tuned audio-reactive interpolations. This group is randomly-sampled snippets of the videos in the test set (detailed further in Section 5.2). These videos are custom-made to fit well with the audio through manual iterative refinement (i.e. audio feature parameters and their mappings to the GAN are guessed and refined slowly by hand). This group should have high correlation across the board, but especially with the RMS, drop strength, onsets, and chromagram (as these are generally the primary driving features for these interpolations).

### 3.4.2. Selecting A Correlation Metric

Before diving into the full comparison across groups, we first select a matrix correlation based on the **None** and **Onset+Chroma** groups. These two groups should show a stark difference and so can be used to gauge overall efficacy of the different metrics. Figure 3.1 shows the histograms of the concatenated correlations (3.1a) and all pairwise correlations (3.1b) for every interpolation in the two groups. The median correlation is denoted with a vertical dotted line.



**Figure 3.1:** Histograms of audiovisual correlation values for different matrix correlations. (a) shows the correlation between the concatenated audio and video features while (b) shows the correlations between all pairs of audio and video features. The vertical dotted line represents the median of each distribution.

First off, across the board, the median of the correlation distribution is higher for the **Onset+Chroma** group than for the **None** group. This is encouraging as it shows that audiovi-



sual correlation can be measured based on this set of features and matrix correlations—at least in very obvious cases.

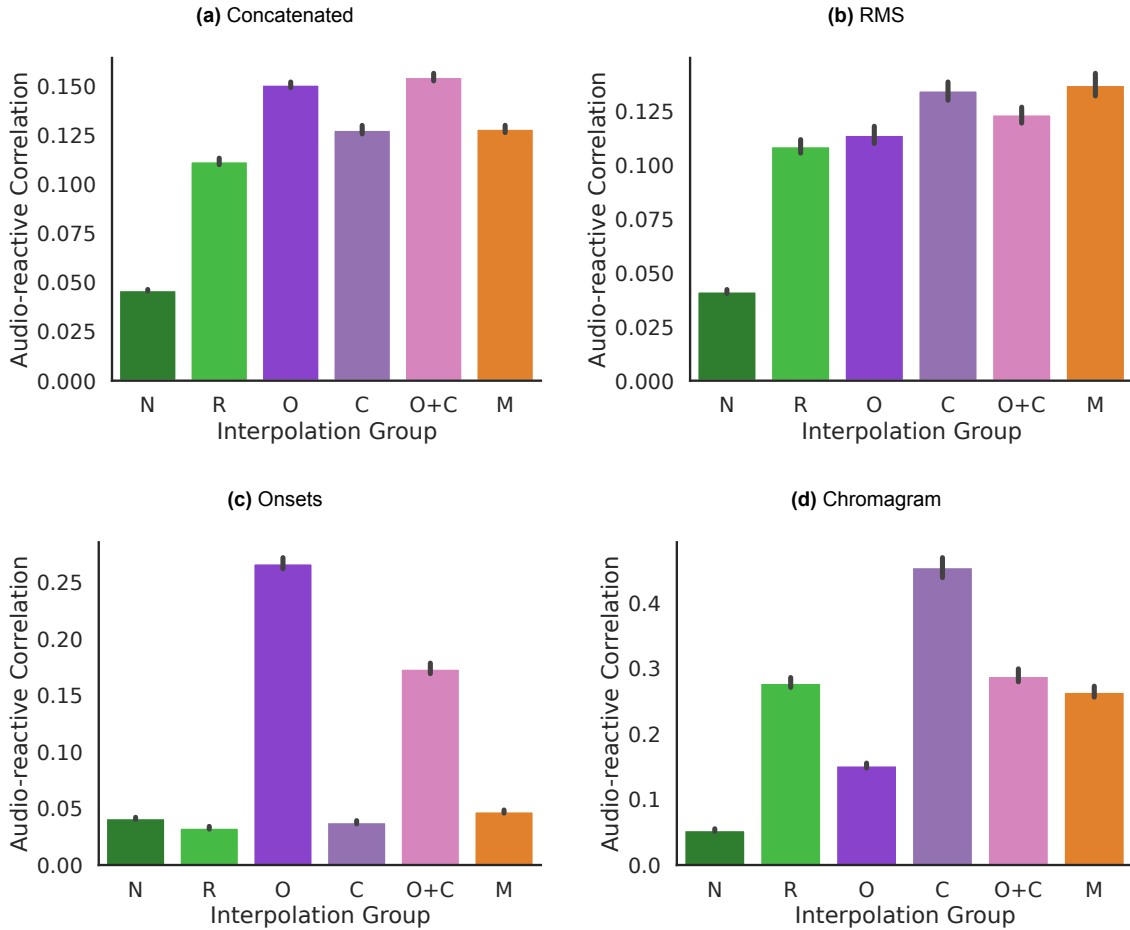
Also apparent is that not all matrix correlations have the same characteristics. For one, PWCCA attributes high correlation to both the `None` and `Onset+Chroma` group with concatenated features. This is undesirable as the `None` group should have essentially no correlation, and we expect higher variability in scores in both cases.

Another interesting result are the bimodal distributions in the pairwise PWCCA and SMI histograms. As PWCCA already has shown undesirable properties, we focus on SMI. Closer inspection of the pairwise correlations shows that the upper mode consists of correlations between two univariate features and the lower mode consists of correlations with at least one multivariate feature. The SMI between two univariate features degenerates to the cosine similarity between these two. There are a number of sparse univariate features which have a disproportionately high cosine similarity.

This leaves only the OP and ARV correlations. It seems that the relative difference in medians is roughly the same and neither one has a bimodal distribution. The OP metric spans a larger range of values between 0 and 1 which makes it preferable if only for ease of inspection of the results. The apparent range of the ARV scores can be increased using an upper bound on the attainable RV coefficient [29], however, we leave this possibility for future work as it does not have any obvious benefits over OP correlation.

### 3.4.3. Comparing Interpolation Groups

Next we compare the orthogonal Procrustes audiovisual correlation between the full set of interpolation groups. Figure 3.2 shows the median audiovisual correlation of each group with different audio feature targets.



**Figure 3.2:** Comparison of the median audiovisual correlation on the different sets of interpolations: None (N), Random (R), Onsets (O), Chroma (C), Onset+Chroma (O+C), and Manual (M). The figures are: (a) concatenated features, (b) RMS, (c) onsets, and (d) chromagrams.

Figure 3.2a shows the concatenated feature audiovisual correlation, our most general metric. The `None` group scores correctly as expected, yet, while the audio-reactive groups are the highest scorers, the `Random` group shows a higher correlation than expected. To get a more comprehensive understanding of these results, we group the pairwise correlations by audio features in Figures 3.2b, 3.2c, and 3.2d.

The correlations between the video features and RMS shown in Figure 3.2b are quite similar to the concatenated results. Following expectations, the `Manual` group has the highest score. Notable is that the other audio-reactive groups outperform the `Random` group (although only slightly) despite not using the RMS feature to generate their interpolations.

The onset strength correlations in Figure 3.2c show strong scores for both the `Onsets` and `Onset+Chroma` group but not for the `Manual` group, despite onsets being one of the primary driving features for this group.

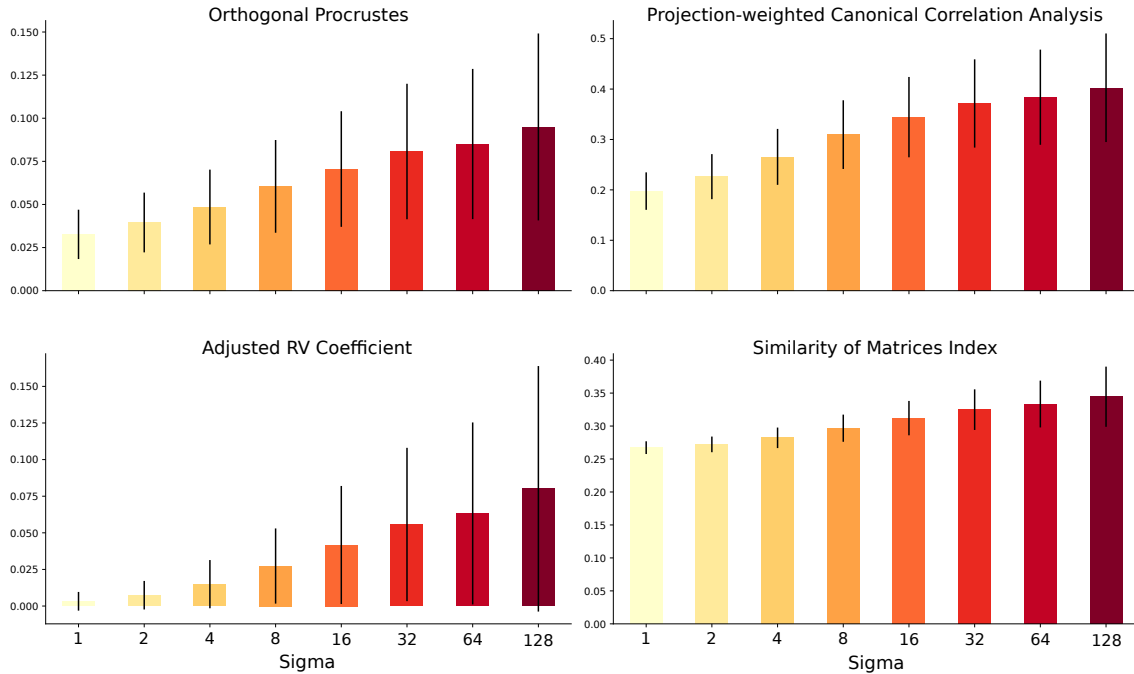
Finally, when grouping by chromagram in Figure 3.2d, the metric shows that the `Chroma`

group has the highest correlation, as expected. However, the `Random` interpolation group performs on par with the `Onset+Chroma` group and outperforms the `Manual` group. This is a surprising result as both the `Onset+Chroma` and `Manual` groups explicitly use chroma-weighted averages in their input sequences while the `Random` interpolations do not.

One major difference between the chromagram and the previous two univariate features is that it is quite smooth over time (due to the feature including a median filter and generally being dominated by the high-energy bass frequencies which often play slower, longer notes). This means that, on average, random interpolations will have a higher correlation with the chromagram than with the sparser, higher-frequency RMS or onset features.

#### 3.4.4. Smoothness Bias

To investigate the correlation bias towards smoother videos, we perform the same experiment as above (with only 100 snippets rather than 500), but with all groups being random, smooth interpolations with varying temporal filter widths. The higher the sigma of the Gaussian filter, the smoother the output video.



**Figure 3.3:** Audiovisual correlation between concatenated features of random interpolation videos generated with different Gaussian filter sigma values.

Figure 3.3 clearly shows the strong connection between smoother videos and higher audiovisual correlation scores. Across all matrix correlation metrics, the same trend is apparent. Multiple different re-weighting schemes are tried to account for smoothness of audio and video features, but none significantly alleviated this issue (see Appendix B).

Thankfully, looking closely at the scale of the orthogonal Procrustes audiovisual correlation, even the highest sigma interpolations, which are almost completely static, have a lower median correlation than the audio-reactive groups in Figure 3.2. This means that despite over-valuing smoothness of the output videos, the proposed audiovisual correlation can give meaningful information on the audio-reactivity of an interpolation video.

**Takeaways:**

- The audiovisual correlation is defined as the orthogonal Procrustes correlation between a set of audio features and a set of video features extracted from an audio-reactive video.
- This correlation measures important aspects of the audio-reactivity and is able to characterize the differences between our groups of interpolations.
- The metric is not perfect. It scores our test set below our expectations and seems to over-estimate the audio-reactivity of smoother videos with less extreme short-term variation.

# 4

## Video Synthesis

Video synthesis can be a memory- and computation-intensive task. We can leverage StyleGAN’s well-organized, smooth latent space to avoid synthesizing a large, low-information pixel tensor directly. Rather, we generate the condensed sequence of latent vectors and noise maps and decode those afterwards with a pre-trained generator. For simplicity, we disregard the possible spatial correlations in the noise maps and only model their change over time. This reduces our task to a pure sequence-to-sequence problem—all of our inputs and outputs are multivariate time series.

To simplify the task, rather than taking the audio waveform as input directly, we extract a number of relevant audio features and use these as input (see Table 3.1 for a full list and description of input features). This gives the model a significant head start towards audio-reactivity as its inputs are already perceptually relevant information from the audio. This comes at the cost of the model becoming unable to discover its own new features that might be even more effective. However, due to the complexity and long-range context length required for learning from an audio waveform directly, this trade-off seems in order.

We introduce four algorithms for audio-reactive video synthesis:

- **Randomizer**: A random generator that synthesizes latent and noise sequences by randomly selecting audio features and decoding strategies. This synthesizer is similar to the prior work described in Section 2.2 but uses an expanded number of input features and leverages the entirety of StyleGAN’s latent and noise hierarchy.
- **Supervised**: A learned model that maps arbitrary audio features to StyleGAN latent and noise sequences. This synthesizer is trained to mimic a collection of hand-made audio-reactive latent interpolations.
- **Self-Supervised**: A second learned model that maps audio features to latent and noise sequences. Unlike **Supervised**, this synthesizer uses an adaptation of the audiovisual correlation as a loss. This allows it to train without any hand-made examples, just a dataset of audio features.
- **HiPPO (High-order Polynomial Projection Operator)**: Another self-supervised synthesizer. Rather than training a sequence-to-sequence model on a dataset, this approach parameterizes the latent and noise sequences directly and optimizes them with a self-supervised loss. This means it can optimize a video that corresponds to a single audio input, rather than requiring a dataset of many examples.

In terms of manual supervision, **Randomizer** is essentially equivalent to previous manually-tuned interpolations. The translation from audio to StyleGAN input sequences being randomized shifts the manual supervision from designing this translation to filtering through the random outputs. **Supervised** also still requires quite a bit of supervision as there needs to be a large dataset of good, hand-made interpolations to train on. While this theoretically only needs to happen once, it is still human supervision.



**Table 4.1:** Comparison of the characteristics of the four synthesizers.

Synthesizer	Pre-training	Dataset	Inference
Randomizer	None	None	Random patch generation
Supervised	Supervised	Large set of paired audio, latent, and noise sequences	Forward pass
Self-Supervised	Self-supervised	Large set of audio sequences	Forward pass
HiPPO	None	None	Self-supervised optimization

Both Self-Supervised and HiPPO allow for true self-supervised video synthesis. The human supervision here comes in the form of the audiovisual correlation loss. The choices of the matrix correlation, audio, and video features does still entail some human design, but broadly the optimization process is guiding itself without explicitly provided samples—it scores on its own what is correct and what is not. Despite this possibility of complete self-supervision, feature weighting with the pairwise feature correlations does still allow for some extra influence over what the Self-Supervised or HiPPO synthesizers actually learn.

## 4.1. Randomizer

The first synthesizer is closest in philosophy to previous approaches by Siegelman [11], Alafriz [30], or Brouwer [17]. The *Randomizer* generates 5-15 random latent and noise sequences each one based on a single audio feature and targeted to a random set of StyleGAN’s layers. The sequences are then randomly merged together into one final “weighted average of weighted averages” sequence. We first give an overview of the algorithm and then discuss the specifics of generating and merging the latent and noise sequences.

What makes the *Randomizer* a stronger baseline than previous work stems from three major points. First the range of audio features it randomly selects from is larger. This includes performing source separation before extracting an audio feature (e.g. to focus on rhythmic or harmonic elements). Second, it randomly chooses to synthesize sequences across the entirety of the StyleGAN latent and noise hierarchy. Third, it generates numerous sequences and combines them in a variety of ways. This allows for generations with a high audio-reactive bandwidth as many scales of structures in the video are reacting to a multitude of different musical features. The downside, of course, is that sometimes the randomly chosen configurations do not work well together.

To help clarify how the *Randomizer* works, we introduce the notion of a “patch” (appropriating the term from analog modular audio or video synthesizers). A patch is the specification of each of the random choices that lead to one possible latent or noise sequence. These patch parameters are drawn randomly beforehand and, given a fixed random seed, deterministically map to a certain output sequence. Each patch consists of a random number of sub-sequences which are combined to form the final output sequences.

Every sub-sequence is driven by one audio feature, translated to latent or noise values using one of three possible strategies. Then, each sub-sequence is combined one-by-one using two possible strategies for each merge. Each strategy for generating or merging

sequences may have multiple parameters governing the exact details of the process. All of these parameters are encompassed by the patch.

While largely similar, generating latent sequences and noise sequences differ in a few points. First, we discuss the latent sequences.

### 4.1.1. Latent Patches

Latent patches are generated using a pre-selected "palette" of latent vectors. These can be completely random or selected carefully beforehand to have a certain overall style.

The process starts by choosing a random number of vectors from the palette and calculating a cubic spline that passes through them spaced equidistantly over the total length of the video. Then, each sub-sequence is generated and merged into this main sequence one-by-one.

#### Generating Sub-sequences

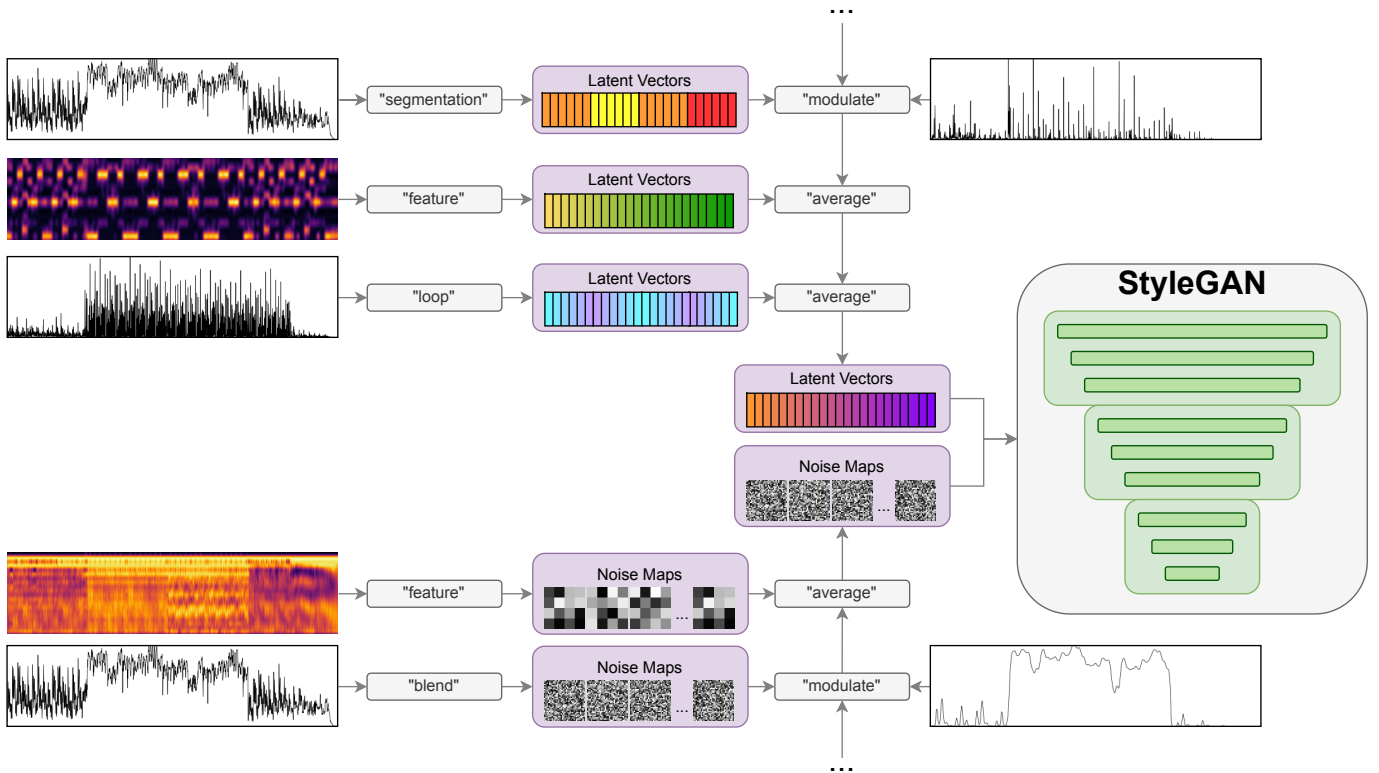
For each sub-sequence, the latent palette is randomly permuted and the first few vectors drawn (the exact number varies by strategy). Next the patch chooses a random depth range in StyleGAN. The layers are divided into three sections (deep, middle, and shallow) and one, two, or all sections are chosen.

The three strategies for generating latent sub-sequences are "segmentation", "feature", and "loop".

- **Segmentation** uses a Laplacian segmentation (see Table 3.1) of a randomly selected audio feature with a randomly chosen number of segments,  $k$ , between 2 and 16. The motivation for this sub-sequence type is to allow for changes in visual style between different sections of the song. Laplacian segmentation assigns a label to each frame in the sequence according to the predicted membership in one of  $k$  different sections. For example, if an audio sequence of length 15 follows an ABCBA pattern and  $k$  is set to three, the expected outcome is the sequence 111222333222111. Each section is assigned one latent vector and these are gathered into each of the corresponding frames. Finally, a Gaussian filter is applied to ensure the transitions between sections are smooth.
- **Feature** applies a feature-weighted moving average. Each dimension of the feature sequence is assigned a latent vector, the feature is normalized to sum to one in each timestamp, and then the sequence is generated by taking the sum of latent vectors weighted by the feature over time. This strategy allows latent sequences to change directly with changes in features such as when an onset occurs, different notes are played, or there is a change in the frequency distribution.
- **Loop** taps into the pattern-based structure of music. The tempo of the music is estimated and a random selection of latents is made to loop every  $n$  bars using spline interpolation (with  $n$  chosen randomly from 4, 8, 16, or 32).

#### Merging Sub-sequences

Once the sub-sequence has been generated, it needs to be added into the main sequence. This is either done by the "average" strategy, which simply adds the two sequences and divides by two, or by the "modulate" strategy. The second strategy chooses a random univariate audio feature, normalizes it to lie between 0 and 1, and uses it to blend back and forth between the main sequence and the new sub-sequence.



**Figure 4.1:** Schematic of the Randomizer synthesizer. Each row represents a sub-sequence. A random audio feature is drawn (left) and translated via a random strategy into a sequence. These sequences are then merged into the main sequence one by one (note that the "modulate" merge strategy draws a random audio feature as well). The final latent and noise sequences are decoded with a pre-trained StyleGAN generator.

### 4.1.2. Noise Patches

Noise patches are generated similarly to latent sequences. One important difference is that, due to their spatial dimensions, noise sequences must be generated lazily to preserve memory (naively generating these tensors for a 5-minute video requirements on the order of 300 GB of memory). This means that, rather than generating the full tensor beforehand and iterating through it when rendering, each noise map is a deterministic function of the frame index.

The three noise sub-sequence generation strategies are "feature", "blend", and "loop".

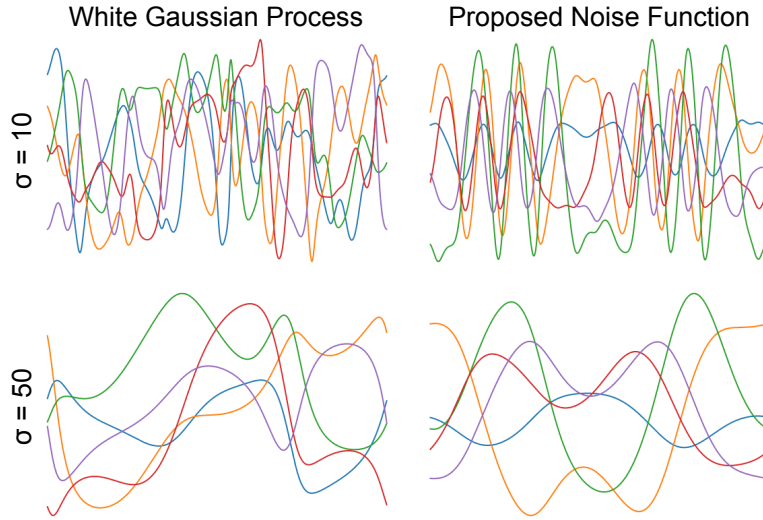
- **Feature** is essentially identical to the latent sub-sequence strategy. Each noise matrix is treated as a flat vector and the feature-weighted average is calculated batch by batch during StyleGAN decoding.
- **Blend** is also similar but uses the weighted average of the audio feature to modulate between a slowly interpolating noise sequence and a quickly interpolating noise sequence. This causes a squirming, quickly interpolating visual effect whenever the audio feature has a high value and more constant movement when it is low.
- **Loop** is slightly different from for latent sub-sequences due to lazy evaluation. The same spline interpolation strategy is hard to adopt as the spline coefficients need to be calculated for a much larger vector (up to 3 million entries even for the shallowest layers). For this reason, a periodic function is used instead. This function is manually

designed to have values similar to a smoothed white Gaussian process. It takes three noise vectors as random seeds, a  $\sigma$  parameter which changes the "smoothness" of the interpolation (similar to  $\sigma$  for a Gaussian process), and an  $L$  parameter which determines how many times the function repeats. The exact definition is:

$$N_0(t) = \sin(\cos(2\pi tL + n_0)/(\sigma/50) + n_1) * n_2$$

$$N(t) = N_0(t)/\sqrt{\bar{N}_0(t)^2}$$

Where  $n_0, n_1, n_2 \sim \mathcal{N}(0, \mathbb{I})$ ,  $\sigma \in \mathbb{R}$ ,  $L \in \mathbb{N}$ , and  $t$  is taken linearly spaced between 0 and 1 for the number of frames. The second step normalizes each frame individually to have roughly normally distributed values.



**Figure 4.2:** Comparison between a random white Gaussian process and the proposed noise function for different sigma values.

Noise sequences are merged in the same way as latent sequences (but done lazily at decoding time, generating each frame in the sub-sequence just-in-time to be added into the main sequence frame).

## 4.2. Supervised and Self-Supervised

The Supervised and Self-Supervised synthesizers are essentially identical except for the training loss. The model is pre-trained on a large dataset of audio(visual) examples. Once fully-trained, the model can be run on new audio feature examples to synthesize a corresponding sequence of StyleGAN inputs. The only difference between the two procedures is the loss used for pre-training: a supervised mean-square-error loss between the model's outputs and a training set of examples or a self-supervised audiovisual correlation between the model's outputs and the audio feature inputs. The design of the model, which we discuss in the following sections, is the same between both synthesizers.

### 4.2.1. Model Architecture

The model architecture consists of two main parts: the envelope generator, which translates the concatenated audio feature time series to an internal representation of a fixed dimension, and a decoder, which translates this internal time series to a sequence for each of the inputs to StyleGAN.

#### Envelope Generator

A variety of sequence to sequence model candidates are compared for the envelope generator:

- GRU [31]
- LSTM [32]
- ConvNeXt [33]
- MLP-ASR [34]
- Transformer [35]
- Sashimi [36]

The exact architectural details and hyperparameters can be found in Appendix A.

#### Decoder

The decoder is responsible for taking the condensed sequence of most audio-reactively-relevant envelopes from the Envelope Generator and transforming it to each of the input sequences to StyleGAN. For simplicity, only a subset of all inputs are modelled with the decoder. For the 1024×1024 pixel pre-trained StyleGAN network that is used, trained by Gonsalves [37], there are 18 latent vector inputs and 16 noise maps. The latent vector inputs are segmented into 3 groups (deep, middle, and shallow) where each layer within a group received the same sequence of latent vectors. Due to computational considerations only the 4×4, 8×8, 16×16, and 32×32 pixel noise maps are used. This means the decoder outputs multiple multivariate sequences ranging from 16-dimensional (4×4) to 1024-dimensional (32×32) for the noise maps and 512-dimensional for each of the latent vector groups.

Two designs are considered for the decoder:

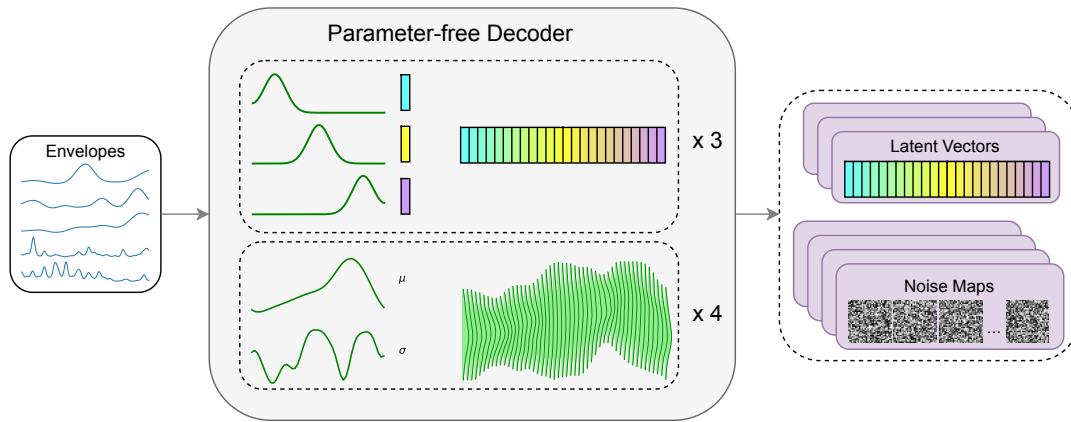
1. **Learned:** a channel-wise multi-layer perceptron, which translates from the internal representation to the output space.
2. **Parameter-free:** a fixed-size weighted average based on a pre-specified palette of output space samples (be they latent vectors or noise). This is essentially the "feature" sub-sequence generator from the `Randomizer` but with the weighting determined by the internal representation from the Envelope Generator (i.e. a learned non-linear function of the audio features).

**Residual Latent Sequences** One important consideration in developing this architecture is that there are many possible sequences of latents and noise that correspond to equally valid audio-reactive videos. This highlights the need for separating the content (visual style) and motion (change over time) during the generation process.

One simple way to approach this in latent space, is to mean-center sequences so that they represent deviations instead. These residual sequences can then be re-centered around any location in latent space to apply a different visual style (by addition of a latent vector with the residual sequence).



This can help make the task easier for the model as the exact values of the latent sequence no longer need to be learned, rather only the relative offset that the incoming audio features should induce.



**Figure 4.3:** A schematic representation of the parameter-free decoder. The input is a multivariate time series where each envelope in the time series is mapped to one of the latent temporal weightings or noise mean and standard deviation values.

**Parameter-free Decoder** The parameter-free decoder takes this idea of content-motion separation one step further. There are still complex linearly-separated image semantics embedded in the 512-dimensional latent space, even when only looking at deviations from a mean. For example, a change in the 34th dimension of the latent vector represents a different visual change, than the 72nd dimension. This means that a large proportion of model capacity must still be used to transform the internal representation to match this output space.

This isn't necessarily the problem we are interested in solving, we just want to generate movement within this space. Therefore, the parameter-free decoder uses a weighted average of a fixed set of latent vectors with the weight of each vector changing over time. This significantly reduces the total parameter count of the model, leaving only the envelope generator model with learnable parameters.

This has the added benefit that the palette of latent vectors can be hand-selected to have the correct visual style beforehand. While the residual sequence can also be "styled" by re-centering it to a different area of latent space, it's movements around that point are harder to adjust and may drift away from the desired style.

The parameter-free decoder takes a fixed number of inputs per latent sequence and noise sequence it models. First, two envelopes for the mean and standard deviation of each of the four noise maps. Then an envelope for each latent vector, where groups are averaged together (e.g. a 6-way weighted average for each of the three latent sequences).

### 4.2.2. Training

With an understanding of the shared architecture of the Supervised and Self-Supervised synthesizers, now we discuss the difference in their training procedures.

#### Supervised Loss

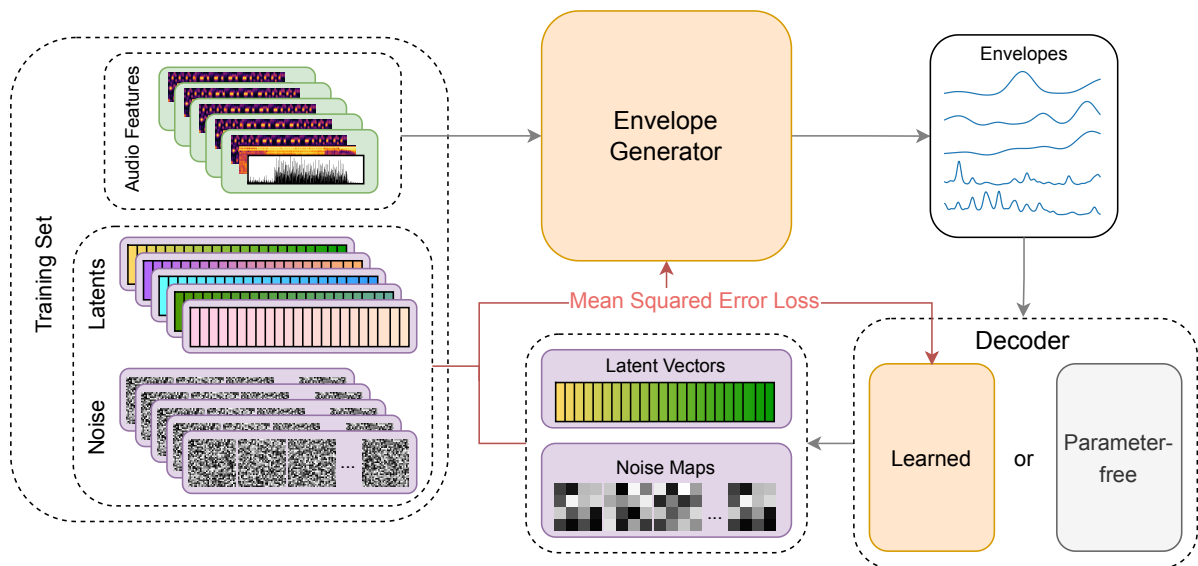
The supervised loss is the most straightforward way of training the learned models. A large dataset with pairs of audio features and the expected output latent and noise se-

quences are used to train the model. For each batch of examples, the audio features are sent through the model and a mean squared error between the model's output and the expected sequences is minimized via gradient descent.

While in theory, this allows the model to learn to synthesize audio-reactive latent and noise sequences of similar quality to the dataset, in practice the amount of data required is intractable. Even with the content-motion decomposition induced by the residual latent sequences or the parameter-free decoder, the supervised loss tasks the model with learning to output a single infinitesimal, knife's-edge path through a multiple-hundred-dimensional space. Multiple augmentation strategies were tried to alleviate this, but none made much of a difference and so are relegated to Appendix B.

During training, input and output sequences are sliced into clips of 8 seconds each offset by 1 second. This led to a training and validation set size of 3072 samples and 704 samples respectively.

Models are trained for 128000 examples with a batch size of 32. The ADAM optimizer [38] is used with a learning rate of  $1e-4$ . The supervised models are trained with a mean squared error loss between the model's output sequences and the true reference sequences from the dataset.

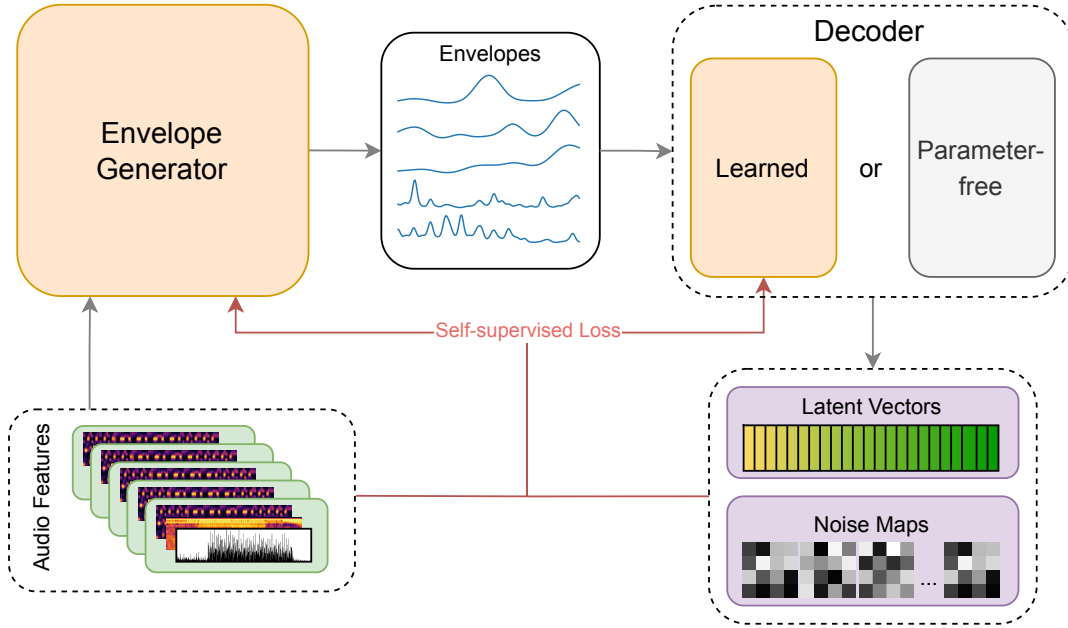


**Figure 4.4:** A schematic of the supervised training process. A large dataset of paired (audio, latent, noise) tuples is required. The audio features are translated to latent and noise sequences by the model. These are compared with the reference sequences via a mean-squared-error loss and the parameters of the model updated via gradient descent (denoted by the red arrows).

### Self-supervised Loss

The self-supervised loss is designed with alleviating the need for a large dataset of examples. Most importantly, it allows multiple different output sequences for the same audio feature inputs to receive a low loss. This is akin to targeting an aircraft carrier rather than a knife's edge. The optimizer's task is thus significantly easier as good areas of parameter-space are much more common and diffuse.

While the metric introduced in Chapter 3 is designed to measure the correlation between audio and video features, to avoid having to render high-definition video during training, the correlation is rather measured between the audio features and the model's output sequences. This makes training significantly more tractable. Intuitively, a high



**Figure 4.5:** The self-supervised training process. A dataset of audio features is required. The audio features are translated to latent and noise sequences by the model. These sequences are compared to the reference audio features with the self-supervised loss (orthogonal Procrustes distance) and the model parameters updated via gradient descent.

correlation on the latent and noise sequences should generate high correlation videos as well, because of the one-to-one correspondence between StyleGAN’s inputs and outputs.

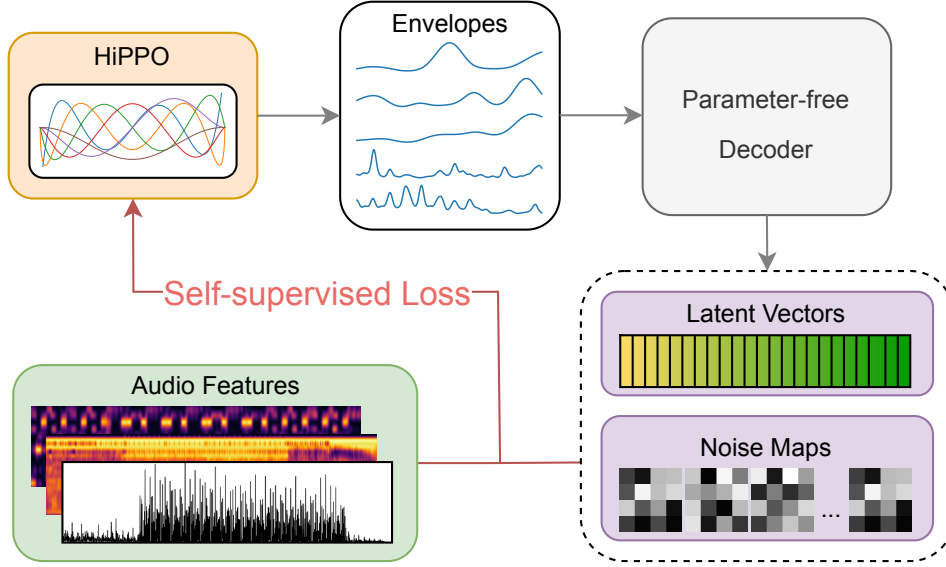
For the self-supervised loss, the model’s output sequences are concatenated and the correlation to the input audio features is maximized ( $d_{op}$  is minimized, see Equation (3.3)). When feature weighting is used, rather than concatenating all input and output sequences, the weighted sum of the pairwise correlations is used.

### 4.3. HiPP0

The final synthesizer is HiPP0. Rather than gathering a large dataset and training for a long time to do well in the average case, we instead directly optimize a single example. Intuitively, the quality can be higher as the interpolation is tailored to the single song at hand rather than all possible inputs in the dataset.

This is made possible by combining the self-supervised loss and the parameter-free decoder. In this case, there is no example output needed and there are no parameters in the decoder that need to be learned. We can directly optimize the inputs to the decoder and place a loss on the audiovisual correlation of the outputs with the audio features.

One issue here is the performance of the optimization. The subspace of envelopes that will generate nice audio-reactive videos is a minute fraction of the total space of multivariate time series. The envelopes should be smooth and continuous to ensure they generate a nice interpolation, but the optimizer does not know that. To imbue the system with a stronger inductive bias, we re-parameterize the envelopes to ensure they are smooth and continuous yet still expressive enough to enable reaching a high audiovisual correlation.

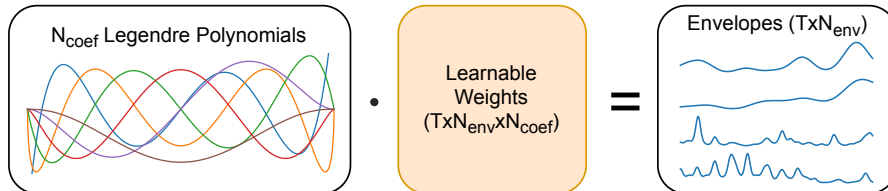


**Figure 4.6:** HiPPO synthesizer optimization process. The envelopes for the decoder are parameterized by the HiPPO module directly. These envelopes are the weighted average of the first  $N$  Legendre polynomials where the weights are learned by gradient descent on the self-supervised loss between the decoded sequences and the audio features.

#### 4.3.1. HiPPO Parameterization

Gu, Dao, Ermon, *et al.* [39] introduce the **H**igh-order **P**olynomial **P**rojection **O**perator as a long-term memory unit for a recurrent neural network. The work is fairly complex mathematically, but for our purposes only a small part of it, the polynomial approximation, is important. The idea is similar to a Fourier transform, where, rather than saving the values of a function directly, a set of coefficients is saved which are decoded to the original signal by multiplying with a fixed set of basis functions. Here, the basis functions are from the series of Legendre polynomials [40]. The HiPPO module can be used to parameterize the full multivariate time series as a weighted sum of these basis polynomials at each time step. The parameters that are optimized are the weights for a fixed set of orthogonal polynomials.

This provides a useful inductive bias as the polynomials are relatively smooth, so the decoded video based on these envelopes will be smooth as well. The number of parameters used for optimization also functions as a control for how smooth the output video is. The more parameters used, the more extreme high frequencies the learned envelopes can contain.



**Figure 4.7:** The internal workings of the HiPPO module. The envelopes are a per-time-step weighted average of a fixed number of Legendre polynomial basis functions.

The ADAM optimizer is used for optimization, starting with a learning rate of  $1e-3$  and using cosine annealing to lower it to  $1e-5$  over the course of optimization. The optimization is run for 2048 steps.

**Takeaways:** We introduce four algorithms for translating audio features to StyleGAN input sequences.

1. `Randomizer` which randomly generates a wide variety of latent and noise sequences based on our set of audio features.
2. `Supervised` which is trained to explicitly mimic a dataset of paired audio, latent, and noise sequences through supervised learning.
3. `Self-supervised` which uses the audiovisual correlation between its audio feature inputs and latent and noise sequence outputs to learn the translation task in a self-supervised manner.
4. `HiPP0` a model-free approach, which uses the same self-supervised learning strategy as `Self-supervised` but rather optimizes a single audio-reactive interpolation directly.

# 5

## Experiments

In this chapter we perform experiments to evaluate the performance of our proposed methods. First we discuss our experimental setup: the frameworks, dataset, evaluation methods, and baselines. Then we show the results of experiments that compare all methods against each other and experiments that ablate certain design decisions of the individual methods.

### 5.1. Frameworks

All experiments are done using PyTorch [41]. This provided a large assortment of readily available sequence learning models, data loading pipelines (e.g. FFCV [42] and Decord [43]), and auto-differentiation for training. Furthermore, there are many state-of-the-art code repositories online which also use PyTorch and so can be integrated easily.

A number of audio-reactive utility functions, data processing functions, and neural networks from Maua [44], a toolkit for creating art with deep learning, are used. Maua provides wrappers for multiple different implementations of StyleGAN under a unified API that allows for sampling images at arbitrary resolutions and rendering audio-reactive latent interpolations.

Audio feature extraction is done using implementations from librosa [45] and madmom [46]. The features in Table 3.1 are reimplemented in PyTorch to make use of the GPU as well as offer the possibility of using autograd.

Implementations of the matrix correlation metrics are adapted from or reimplemented based on [47] and [48].

All code used in the experiments is made available in an online repository for transparency and reproducibility: <https://github.com/JCBrouwer/self-supervised-audio-reactive>

### 5.2. Dataset

The learned models are all trained on a dataset of mainly electronic music (specifically drum & bass, dubstep, and UK garage). The main factor behind this choice of data is the need for examples of audio-reactive interpolations. There are no large sets of these available online, however, having personally made many audio-reactive interpolations, I have 3.5 hours of material that is used as a dataset.

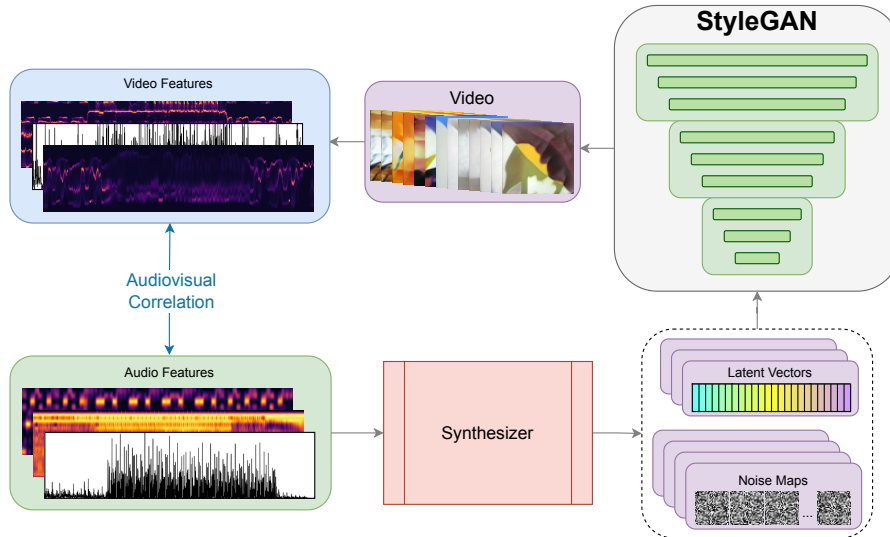
This data is split into three groups, a training set, a validation set, and a test set. The training and validation set are an 80%-20% random split of the hour-long mixes together. The test set consists of 10-15 shorter videos. Due to their short length, the test set videos are the most polished in terms of audio-reactivity due to the relative ease of more detailed refinement during their creation.

The sampling rates of all videos (and so the latent and noise sequences) is fixed at 24 frames per second. For audio feature extraction, the audio is resampled to 24576 samples

per second and processed with a hop length of 1024 samples to create sequences which are also exactly 24 frames per second.

While the self-supervised approaches can use any dataset of audio, for comparison's sake the same dataset is used for this setting as well.

## 5.3. Evaluation



**Figure 5.1:** Schematic representation of the calculation of the audiovisual correlation. Audio features are decoded to a video, from which video features are extracted and finally compared with the original audio features.

We employ a number of metrics to quantify the performance of the video synthesizers.

### 5.3.1. Audiovisual Correlation

The primary metric is the audiovisual correlation as introduced in Chapter 3. To assess this metric, we synthesize videos for the audio of the entire test set (this corresponds to about 250 snippets of video relative to the 500 used in the preliminary analysis in Section 3.4). The audiovisual correlation primarily captures motional aspects of the video.

### 5.3.2. Fréchet SwAV Distance

To quantify the content of the interpolations, we measure use Fréchet SwAV Distance. The Fréchet distance [49] is a distance measure between distributions. The Fréchet Inception Distance [50] is a popular metric for evaluating the similarity between a generative model's outputs and its target distribution. It measures the Fréchet distance between a pre-trained InceptionV3 [51] network's internal representations of a large set (30000-50000) of generated images and a second set of real images.

Recently, there has been evidence that using a self-supervised pre-trained image recognition network (specifically SwAV [52]) instead of InceptionV3 is better suited for evaluating generative models. Morozov, Voynov, and Babenko [53] suggest that SwAV's contrastive training objective, which simultaneously clusters images and enforces consistency between cluster assignments under augmentations, produces internal representations which better capture perceptual importance of images—especially for non-ImageNet

data. Given that our StyleGAN generator [37] is trained on abstract art, this seems to be the better fit.

To evaluate the visual quality of the synthesized audio-reactive interpolations, we measure the FSD between 30000 randomly generated samples from the generator and 30000 random frames sampled from synthesized videos. This evaluation helps to show whether the latent and noise sequences which are generated fall within the distribution that StyleGAN expects, allowing it to generate high-quality samples.

### 5.3.3. Inference Time

The final metric we evaluate is the inference time. This is the amount of time that it takes for the set of audio features to be translated to the set of latent and noise sequences. The time taken to calculate audio features and the time taken to render the actual video are neglected as these are identical across methods.

## 5.4. Baselines

To gain an understanding of how our methods compare with other publicly available approaches, we also run our evaluation on a few baselines. Next to the hand-made test set, which is created using Maua [44], we test **LucidSonicDreams** [13], a Python package which synthesizes audio-reactive latent interpolations (see Section 2.2 for details), and **WZRD.ai** [54], a website which generates audio-reactive latent interpolations. While Maua and LucidSonicDreams are open source, it is not public knowledge how WZRD.ai synthesizes videos. Nevertheless, we include it as it is one of the few tools which are available online.

While both LucidSonicDreams and WZRD.ai provide multiple parameters to tweak the audio-reactivity of their interpolations, for simplicity the default parameters are used with randomly drawn latent vectors. This means that audio-reactivity and visual quality can presumably be improved by refining these parameters as has been done with the test set.

## 5.5. Results

Example audio-reactive interpolation videos can be found in the supplementary material: <https://jcbrouwer.github.io/thesis/supplement>.

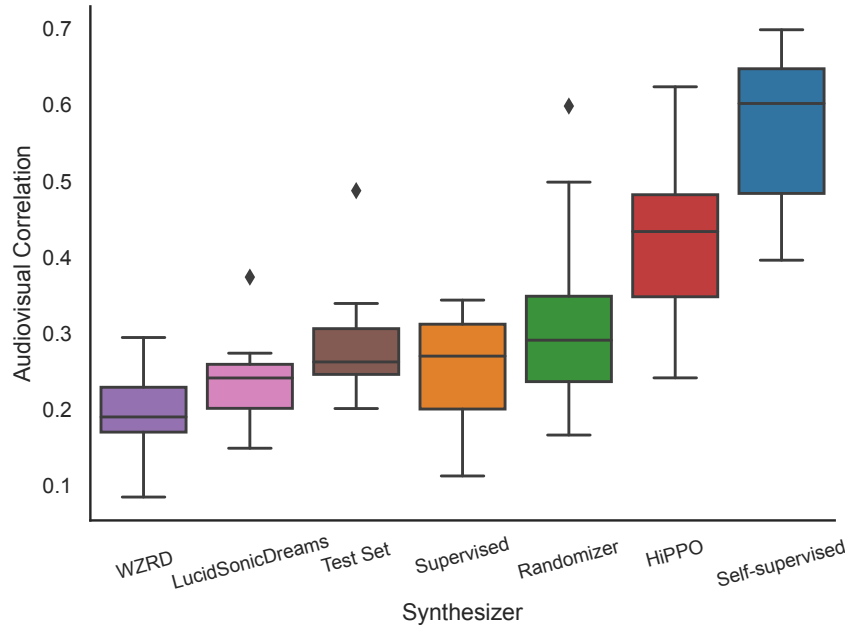
### 5.5.1. Audiovisual Correlation

First we analyze the audio-reactive quality of the different synthesizers. Figure 5.2 shows the distribution of audiovisual correlation scores (as described in Section 5.3.1) of each of the different synthesizers that are evaluated. The scores for all models are shown for the best-performing variants (see Section 5.5.4 and Section 5.5.5).

There is a large difference in performance between the self-supervised methods (HiPP0 and Self-Supervised) and the rest. This confirms that maximizing correlation between the audio features and StyleGAN’s input sequences results in videos that are correlated with the audio features as well. This shows that the input sequence representations are rich enough that they can serve as proxy for the audio-reactive properties of the output video.

Continuing left towards the lower correlation methods, we see that the randomizer’s median correlation is close to that of the supervised model, test set, and LucidSonicDreams baseline. However, there are multiple videos with significantly higher correlation.





**Figure 5.2:** The distribution of the concatenated audiovisual correlations on the test set for each of the synthesis approaches.

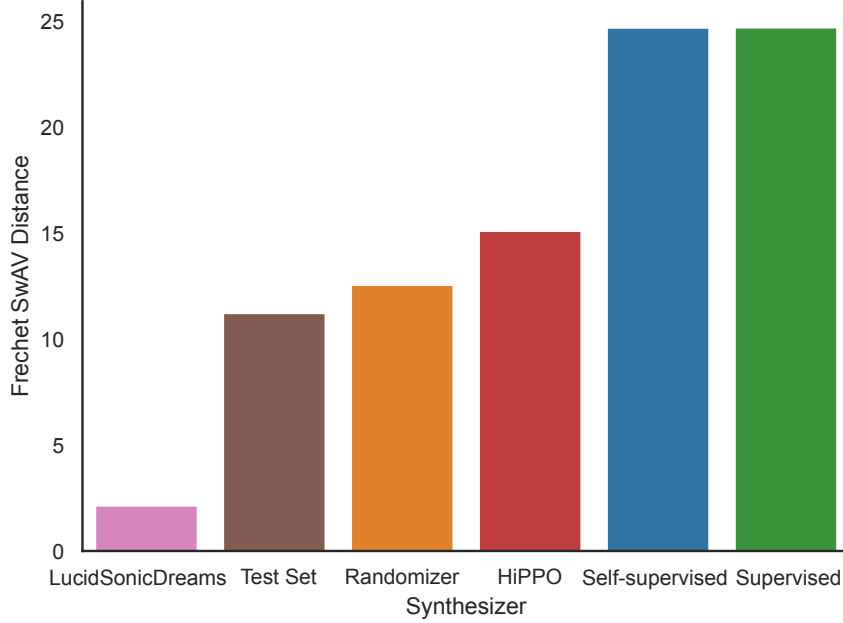
This shows that the `Randomizer` is capable of generating high-quality audio-reactive results (albeit rarely). It is clear that the focus on increasing audio-reactive bandwidth does improve the `Randomizer`’s results over the baselines, even if the difference in averages is not large.

The supervised model has a slightly higher audiovisual correlation than the test set on average, however, it also has a number of results which are significantly worse than any of the test set samples. This can likely be attributed to two factors: (1) the training set being based on longer audio-reactive interpolations and thus being less audiovisually correlated than the test set and (2) the model struggling to learn the supervised loss due to the challenging loss landscape (the knife’s edge problem mentioned in Section 4.2.2).

Finally, we analyze the baseline results. `WZRD`, `LucidSonicDreams`, the test set, and the `Randomizer` are all essentially following the same strategy to synthesize videos. The `Randomizer` has the most diverse set of audio features and so performs best on the overall concatenated audiovisual correlation. `LucidSonicDreams` uses the smallest subset of features: only the maximum energy frequency band in the spectrogram, although separately on the harmonic and rhythmic components of the full audio signal. `WZRD` performs worst of all the methods in terms of audiovisual correlation. However, `WZRD` provides an interface for fine-grained scheduling of latent vectors over the course of an interpolation which can greatly increase the artistic control over the video. Spending some time fine-tuning interpolations via the interface can probably significantly improve the audiovisual correlation and visual quality at the cost of manual work.

### 5.5.2. Fréchet SwAV Distance

Next we analyze the visual quality of the synthesized videos. Figure 5.3 shows the FSD (as described in Section 5.3.2) between generated audio-reactive interpolation videos and



**Figure 5.3:** The Fréchet SwAV Distance between random generated images and audio-reactive video frames. Lower is better.

the regular generator distribution. We are unable to calculate the FSD for WZRD as we do not have access to a large set of random samples from the generator. Here it becomes apparent that there is a cost for the higher audiovisual correlation of the self-supervised models.

The frames in interpolations synthesized by LucidSonicDreams are closest to the native distribution of the generator. This is an expected result as LucidSonicDreams operates in the unmapped latent space,  $\mathcal{Z}$ . Thus, vectors in the latent sequence are always mapped into a valid section of the mapped latent space,  $\mathcal{W}$ , because StyleGAN’s mapping network normalizes each vector to have unit variance. LucidSonicDreams also does not use the noise maps, which eliminates another possibility for going out of distribution.

The remaining methods all operate in  $\mathcal{W}$ -space which allows for targeting audio-reactive effects to different areas of the StyleGAN latent hierarchy. However,  $\mathcal{W}$ -space is a complex, nonlinear re-mapping of  $\mathcal{Z}$ -space and so it is not as easy to ensure that linear combinations of latent vectors remain within the distribution that StyleGAN’s synthesis network expects. On top of this, the mean and standard deviation of the noise maps are also modulated over time for these methods. The effect on the individual frames will also increase the distance to the generators natural distribution.

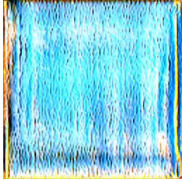
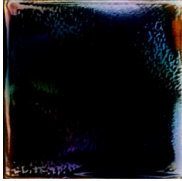

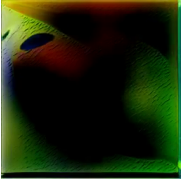
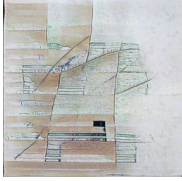
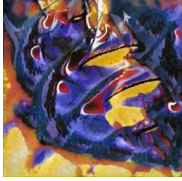
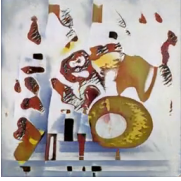
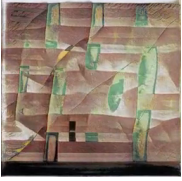
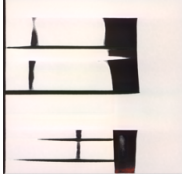

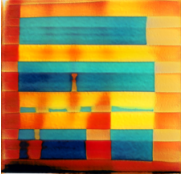

While the difference seems extreme, we note that the test set is the product of a process of iterative refinement for qualitatively better visual aesthetics and audio-reactivity. The distribution of these frames differs from the natural distribution of the generator, but this does not necessarily imply a visually important difference.

The two learned models are another story. Those with experience pushing StyleGAN’s latent space to its limits will recognize the saturated, flat, colored artifacts that are sometimes present in the videos generated by Supervised and Self-Supervised. Interestingly, the two seem to be relatively equal in their distortion of latent space. This is despite the fact that Self-Supervised receives no supervision regarding the real distribution of latents

while Supervised is trained on hours of examples.

Table 5.1 provides some examples of generated frames. The Supervised samples are cherry-picked to highlight extreme, out-of-distribution generations. Note that despite the higher FSD, the randomly selected Randomizer samples are still visually similar to the random  $\mathcal{Z}$ -space samples.

**Table 5.1:** Comparison of images of the Supervised and Randomizer synthesizers with random generations from the StyleGAN generator. Supervised samples are cherry-picked to highlight extreme, out-of-distribution generations, the other two rows are not curated.

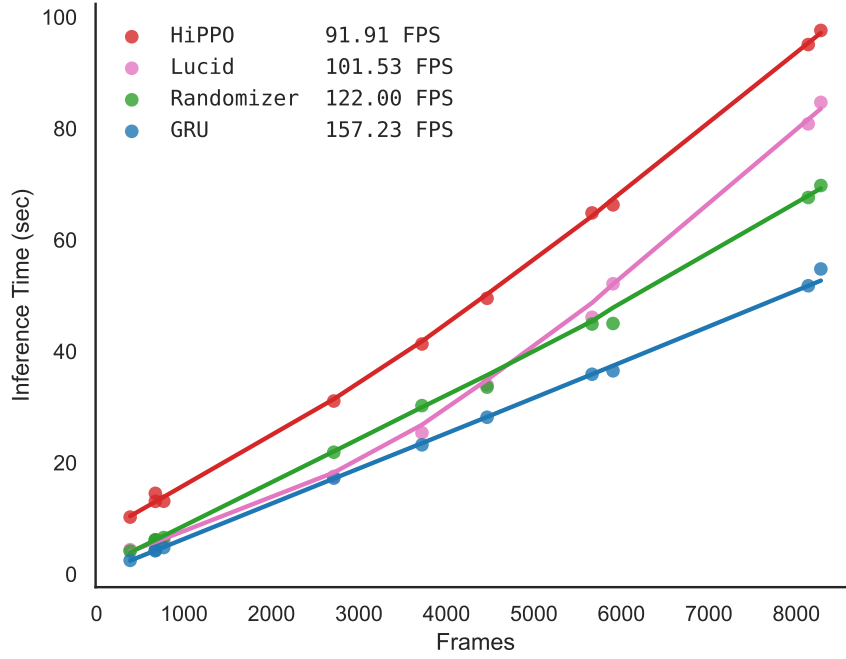
Synthesizer	Example Frames			
Supervised				
Randomizer				
Natural Distribution				

### 5.5.3. Inference Time

Our final point of comparison is the time it takes to perform inference based on audio features of a given length. Figure 5.4 shows the time taken to output latent and noise sequences for a variety of different video lengths. The trend lines are the LOWESS (Locally Weighted Scatterplot Smoothing) [55] fit of the data to help make the scaling trend clear. The FPS (frames per second) values are the inverse slope of the linear regression (imperfect for trends with curvature). We do not analyze the test set as these videos are generated with multiple different scripts that do not all have the same inference time. Supervised and Self-Supervised only differ in their training procedure and so have the same inference time, denoted in the figure as GRU. The experiment is performed on a machine with an RTX 3090 GPU (24 GB of VRAM), a 12-core AMD Threadripper 1920X CPU, and 64 GB of RAM.

The GRU is fastest of the proposed methods, scaling linearly with sequence length. This requires training first, though, which requires between 25 and 40 minutes on the aforementioned machine (the training time is longer when the supervised loss and/or learned decoder are used).

The next fastest method is either the Randomizer or LucidSonicDreams depending on the length of the interpolation. LucidSonicDreams scales super-linearly with the sequence length while Randomizer scales linearly.



**Figure 5.4:** The time taken to translate audio features to latent and noise sequences for different number of frames.

Finally, the slowest method is HiPP0. This method requires a one-off optimization for each video which takes a significant amount of time. Throughout all experiments the number of optimization steps is fixed at 2048 (based on preliminary convergence experiments). However, presumably a more aggressive learning rate and/or fewer steps can make this more competitive.

#### 5.5.4. Supervised vs. Self-Supervised

Next we compare performance of the learned models with different architectural design and training procedures. Each of the audiovisual correlations are found using the procedure explained in Section 5.3.1.

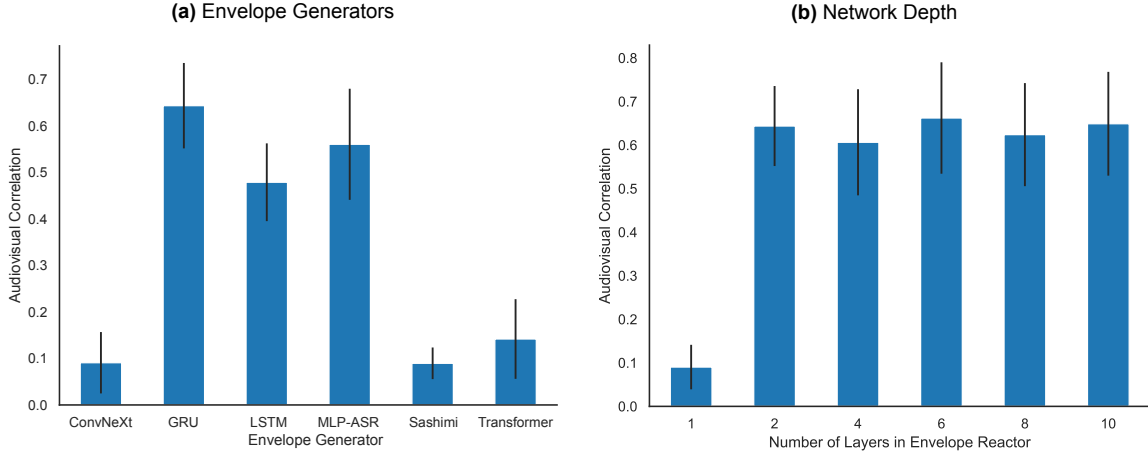
**Table 5.2:** The median audiovisual correlation of the Supervised and Self-Supervised synthesizers with different decoders and residual settings.

	Learned		Parameter-free	
	Supervised	Self-supervised	Supervised	Self-supervised
<b>Non-residual</b>	0.2159	0.4610	0.2202	0.3912
<b>Residual</b>	0.2512	0.3494	0.2744	0.5522

Table 5.2 shows a comparison of the audiovisual correlation on the test set of the Supervised and Self-Supervised models with different combinations of decoder and residual latent sequences. We see that across the board the Self-Supervised outperforms Supervised.

Residual latent sequences do not have a significant effect on the performance of

Supervised. However, they do influence Self-Supervised. When the learned decoder is used (top row), residual latent sequences actually hurts the performance, but when the parameter-free decoder is used (bottom row), residual sequences helps it significantly.



**Figure 5.5:** The audiovisual correlation of the Self-Supervised synthesizer with (a) varying envelope generator and (b) varying depth of the envelope generator.

Next to design of the decoder, the other major component in our learned models is the Envelope Generator. This part of the design is responsible for the temporal analysis—translating a sequence of audio features to a sequence of internal representations to drive the decoder. We compare the audiovisual correlation of several sequence-to-sequence models in Figure 5.5a. Each model is configured with the same depth and such that the number of learnable parameters is within 5% of each other (around 15k learnable parameters). Each model is trained with the parameter-free decoder, residual latent sequences, and the self-supervised loss.

There are two major groups: those sequence models that converged successfully (GRU, LSTM, and MLP-ASR) and those that failed to learn to synthesize audiovisually correlated videos (ConvNeXt, Sashimi, and Transformer).

In the group of successful models are two recurrent networks, the GRU and LSTM. The GRU is a more recent, slightly simpler recurrent design. Apparently in this case Occam’s razor gives the GRU the edge. MLP-ASR is a speech recognition adaptation of the recent glut of MLP-Mixer [56] style models. These models revive the power of simple linear layers by alternating between temporal operations and channel-wise operations with residual connections. It is interesting to see that such a simple model without recurrence or a temporal inductive bias can perform so well.

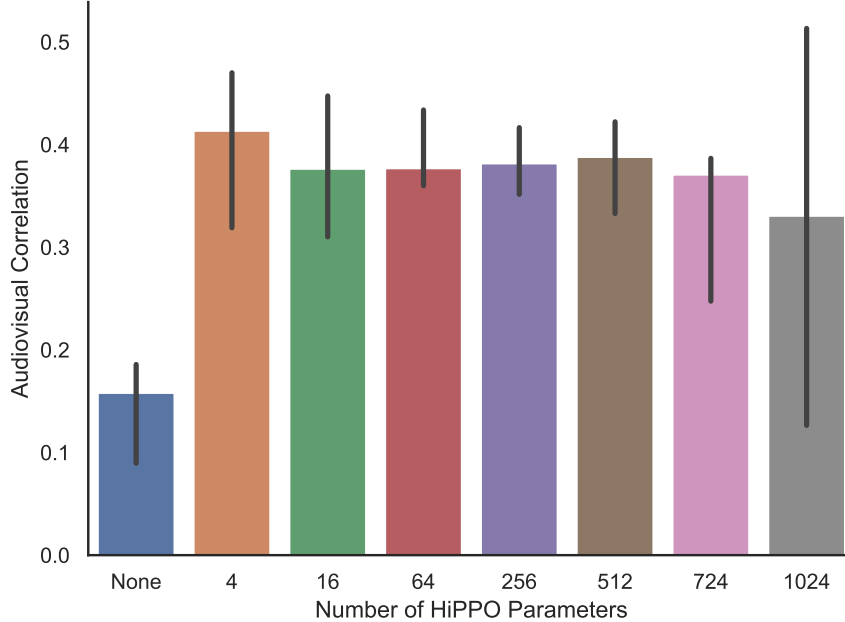
While it might be possible to find better hyperparameter configurations that allow the failing group of networks to converge, there are some plausible reasons why each of these does not perform well in our task. ConvNeXt is a network adapted from 2-dimensional convolutional image recognition to use 1-dimensional, temporal convolutions. It is similar to the residual structure of MLP-ASR, but replaces the alternating temporal and channel-wise linear layers with convolutions. Apparently, this naive translation does not immediately confer the same strengths. Perhaps the architecture is simply not suited for sequence-to-sequence learning. Sashimi is a state space model which follows upon the work of the HiPPO module [39]. The main task considered is learning from raw audio waveforms. It is possible that the inductive biases in the network are specifically suited to these kinds

of low-information, high-noise tasks and so struggles to accurately model our smoother audio-reactive task. Finally, the Transformer is well-known to be data hungry. The architecture especially excels after training for a long time on vast datasets, therefore it seems to not be well-calibrated for this small scale setting with few learnable parameters.

The final major parameter that we investigated is the depth of the envelope generator network. Figure 5.5b shows that single layer networks do not have sufficient capacity to translate audio features successfully. However, more than two layers does not make a significant difference in the performance.

### 5.5.5. HiPPO

Figure 5.6 shows the audiovisual correlations on the test set with a different number of HiPPO coefficients. Directly optimizing the envelope tensor shows significantly worse performance than using even four HiPPO coefficients. There does not seem to be a huge correlation between number of coefficients and audiovisual correlation until going above 512. At this point the median starts to deteriorate and the variation increases markedly. At this point the optimization task has probably grown too complex for the amount of learning signal the self-supervised loss can provide. While not significantly higher, using a small amount of coefficients seems to improve performance slightly. This corresponds to the smoothest results, though, as a smaller series of polynomials to choose from, fewer coefficients also means less flexibility for more extreme, quick changes. It is possible this boost in score can be attributed to the smoothness-bias in the audiovisual correlation. These low number of coefficients also seem to have a high variance in scores. The most consistent (lowest variance) number of coefficients is 256. Perhaps this is evidence that this is the sweet spot in terms of representational power and smoothness inductive bias.



**Figure 5.6:** The audiovisual correlation of the HiPPO synthesizer with different numbers of learned coefficients. None represents optimizing the envelope tensor directly without using HiPPO.

Finally, we analyze the effect of feature weighting in the self-supervised loss. While this is also possible with the learned models, for computation reasons we perform this ex-

periment with the HiPP0 synthesizer. Table 5.3 shows the median audiovisual correlation over the test set with different feature weights. Each row corresponds to training with one audio feature weighted 10 times stronger than the rest. Each column shows the median pairwise score for the labeled audio feature.

Within each row, the correlation with the weighted feature is generally strongest (highlighted on the diagonal). This shows that weighting certain audio features in among all pairwise correlations does have an effect on the video feature correlation of the output. This is encouraging as it gives more fine-grained control over the self-supervised learning approaches which are otherwise a black-box.

Notably, the videos synthesized with extra onset-weighting do not have the highest correlation, unlike all other features. This is surprising as it is in fact the lowest onset correlation of all weightings, despite having the highest score within the row.

**Table 5.3:** Median audiovisual correlation of HiPP0 on the test set with different feature weightings.

Feature Weighting	Drop Strength	Onsets	Chromagram	Spectral Flatness
10x Drop Strength Weight	<b>0.526</b>	0.314	0.346	0.227
10x Onsets Weight	0.107	<b>0.152</b>	0.110	0.104
10x Chromagram Weight	0.264	0.181	<b>0.401</b>	0.172
10x Spectral Flatness Weight	0.225	0.181	0.249	<b>0.394</b>

# 6

## Conclusion

Our results show that we have succeeded in taking the first steps towards automated audio-reactive video synthesis. We have been able to develop a metric that is a strong indicator of a video’s audio-reactivity. While the audiovisual correlation is still sometimes misled by spurious correlations and does not always score manually-crafted audio-reactive interpolations as highly as expected, it clearly captures at least some parts of our intuitive sense of audio-reactive quality. This has allowed us to design and compare multiple synthesis approaches that can synthesize audio-reactive videos without audiovisual training data.

Currently, these methods still have notable drawbacks. For some algorithms, the chance of synthesizing highly audio-reactive results is low, otherwise the visual quality may be low, and there are not always explicit controls to change the output when it is not satisfactory. However, in terms of maximum attainable audio-reactive quality and variation, the methods exceed prior work and illuminate multiple paths forward for even better results.

### 6.1. Summary of the Contributions

First, we introduce a metric, the audiovisual correlation, that can measure the notion of audio-reactivity in videos. The idea is centered around measuring the degree to which temporal patterns in the video match temporal patterns in the audio. We propose two sets of features to extract from audio and video and investigate different ways of measuring their correlation.

Our second contribution is the design of four different algorithms which generate expressive audio-reactive interpolation videos based on a wider range of audio inputs and GAN inputs than prior work. The algorithms each have different drawbacks and benefits ranging from computational efficiency, data efficiency, manual control, and automation. The core novelty is to use our audiovisual correlation metric for self-supervised learning. This allows us to learn a model that can synthesize audio-reactive videos without needing a dataset of example videos. We even extend this procedure to synthesize individual videos to match a single audio input, rather than needing to train a model.

Returning to our research questions, we show that our audiovisual correlation metric can, in fact, measure the difference between groups of videos with different audio-reactive characteristics. Our metric is able to characterize intuitive differences between the groups. For example, groups that have a high response to rhythmic elements in the music also have high audiovisual correlations for the rhythmic features. We also see that the metric provides a useful training signal to learn latent sequence models.

We compare our algorithms with other GAN-based audio-reactive video synthesis approaches and show favorable performance in terms of audio-reactivity and inference speed. The proposed synthesizers are able to leverage the wide range of audio features



to produce a highly audio-reactive video. Using our self-supervised loss and parameter-free decoder, we are able to train models without needing video examples and even with only a single audio input. We also find that our synthesizers consistently have higher audiovisual correlation than a random baseline with the same expanded audio feature set.

## 6.2. Future Work & Recommendations

For the audiovisual correlation, there is one glaring issue apparent in our preliminary study: a bias towards smoother interpolations. Solving this issue could improve the audiovisual correlation significantly by re-calibrating the metric toward recognizing truly relevant aspects of audio-reactive videos. Furthermore, a more thorough analysis of the effect of feature choice would also be useful. Perhaps there are other combinations of audio and video features which capture audio-reactivity even better. It could be interesting to add deep feature representations of audio and video as well. These might integrate more high-level conceptual patterns such as instrumental compositions in music or styles of motion in video. Another candidate would be to apply more elaborate source separation of the music such as using Spleeter [57] or Unmix [58] to extract individual instruments. This kind of source separation could also be used to make univariate features multivariate which might improve the results of different matrix correlations. Finally, another promising way of aligning the audiovisual correlation with human intuition is to add some kind perceptual importance weighting. This could be in the form of focusing on more salient sounds in the music, integrating sectional analysis into the metric, or investigating what kind of audiovisual patterns are most likely to be perceived as audio-reactive.

There are also many improvements that could be made to the synthesizers themselves. Using a longer sequence length at training time might make a large difference in how well the audiovisual correlation can evaluate the similarity of sequences. Another option would be to use or include the raw waveform as input for the model rather than taking a feature extraction approach. This could allow the synthesizers to learn better audio-reactive representations within the task context rather than relying on generic features which were developed for other specific use cases. For computational feasibility and experimental iteration speed, the self-supervised loss was chosen to evaluate the correlation between the audio features and the inputs to StyleGAN. However, with some tricks to ensure the size of videos does not grow to be intractable, it might be significantly more effective to use video features for self-supervision during training.

It could also be fruitful to find alternative ways of incorporating the audiovisual correlation into synthesizers rather than using it as a loss function for gradient descent. A simple post-filtering improvement to the `Randomizer` would be to synthesize multiple videos and then only return the one with the highest audiovisual correlation. Similarly, the `Supervised` model could be trained on an automatically curated set of high-correlation random examples. This could allow creating an arbitrarily large dataset which might not suffer from the knife's-edge problem. Another item on the controllability wishlist is to allow the learned models to generate multiple different videos for the same audio input. This could be achieved by adding an extra motion latent seed to the architecture to alleviate the deterministic, black-box nature of the current models.

There is also a large design space to be explored of different decoders. The proposed parameter-free decoder is only one possibility. The `Randomizer` consists of multiple more elaborate strategies for decoding audio features to latent and noise sequences. It might be possible for instance to use stochastic inference to learn to bias the underlying ran-

dom distributions of the `Randomizer` to generate better results. Alternatively, applying evolutionary algorithms to the patch specifications could also improve the performance. In fact, the decoder might even be designed to use a completely different visual synthesis system. The decoder might as well control fractal rendering software or a complex blender scene, as long as there is some intermediate differentiable representation that the self-supervised loss can be evaluated on.

### 6.3. Human-in-the-loop Workflow

While there is still much left to improve, the proposed audio-reactive synthesizers are already practically useful in their own right. A prototype, human-in-the-loop workflow has been designed that can assist in synthesizing high-quality audio-reactive interpolations for user-supplied music with little technical knowledge. The workflow centers on the iterative refinement of interpolations generated by the `Randomizer`.

First, the song is segmented with Laplacian segmentation or along user-defined timestamps. Then for each segment, one-by-one, the user receives a randomly generated interpolation. Based on the interpolation they received, the user can choose a number of adjustments to improve it. Because each patch deterministically maps to an interpolation, the same patch can be resynthesized with some parameters changed to be more to the user's liking. For example, the sigma values of all noise maps can be increased for more noisy frames, the amplitude of all envelopes can be decreased for less extreme audio-reactive motion, or the latent palette can be resampled in the shallow layers for a different color scheme. Once the user is satisfied with an interpolation for a given section, they can move on to the next. Finally, after going through the process for each section, the full video can be stitched together from the individual interpolations and rendered at high resolution.

While this workflow largely relies on the `Randomizer`, the other synthesizers can easily be incorporated as well, simply by adding them as sub-sequence generators. This can help improve results by ensuring that individual sub-patches are already more audio-reactive before merging into the main sequence. This synergy offers a great way to fine-tune the generation quality of the proposed workflow by adding pre-trained models which are more tailored to the specific music being processed. Multiple sequence models trained with different feature weightings could be used. This tool offers a glimpse into the future possibilities for assisting creativity with artificial intelligence.

### 6.4. Final Thoughts

We have explored many paths towards synthesizing audio-reactive interpolations with reduced manual supervision. This has led to a metric which can be used to evaluate the audiovisual correlation in a video. We used this metric to teach multiple different machine learning systems to generate sequences of inputs for a pre-trained image generation GAN. The videos generated by these systems excelled over prior art in terms of audio-reactive quality as well as required amounts of manual tweaking. While at present the synthesizers sometimes push the pre-trained generator out of its natural distribution of images, given the abstract nature of the generated videos, this is a worthwhile trade-off.

We note that the proposed audiovisual correlation is quite generic and can capture a broad range of possible correlations in videos by carefully selecting an appropriate set of audio and video features. This opens up multiple paths of research towards extracting useful information from videos as well as providing supervision for machine learning models.

We introduce a number of methods that span the Pareto frontier of training and inference time, audio-reactive quality, visual quality, manual supervision, and data requirements—allowing practitioners to choose the approach which is best tailored to their needs. Our self-supervised models are easy to train even on single audio examples in as few as two minutes. This makes the methods accessible even on lower-end consumer hardware, offering stronger audio-reactive interpolation video synthesis options through services like Google Colaboratory [59].

While this work may seem to be automating the creative process of an audiovisual artist, we show that it is possible to use these tools to enrich artists' workflows rather than replace them. There are many options to explore that can allow for greater control over these complex, powerful algorithms.

*Finally, from the bottom of my heart I'd like to thank Lydia and Cynthia for guiding me through these last months, Michael for joining my examination committee, and Wiep, Thore, and Ryan for their feedback and support.*

# References

- [1] R. Geiss. “MilkDrop.” (), [Online]. Available: <https://www.geisswerks.com/milkdrop/>.
- [2] B. Foundation. “Blender - Free and Open 3D Creation Software,” blender.org. (), [Online]. Available: <https://www.blender.org/>.
- [3] Derivative. “TouchDesigner,” Derivative. (), [Online]. Available: <https://derivative.ca/>.
- [4] C. Reas and B. Fry. “Processing: A flexible software sketchbook and a language for learning how to code within the context of the visual arts.,” Processing. (), [Online]. Available: <https://processing.org/>.
- [5] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8110–8119. [Online]. Available: [https://openaccess.thecvf.com/content\\_CVPR\\_2020/html/Karras\\_Analyzing\\_and\\_Improving\\_the\\_Image\\_Quality\\_of\\_StyleGAN\\_CVPR\\_2020\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2020/html/Karras_Analyzing_and_Improving_the_Image_Quality_of_StyleGAN_CVPR_2020_paper.html).
- [6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative Adversarial Nets,” p. 9,
- [7] A. I. Humayun, R. Balestrierio, and R. Baraniuk, “Polarity Sampling: Quality and Diversity Control of Pre-Trained Generative Networks via Singular Values,” p. 10,
- [8] R. Balestrierio and baraniuk, “A Spline Theory of Deep Learning,” in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 3, 2018, pp. 374–383. [Online]. Available: <https://proceedings.mlr.press/v80/balestrierio18b.html>.
- [9] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, “On the Number of Linear Regions of Deep Neural Networks,” Jun. 7, 2014. arXiv: 1402.1869 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1402.1869>.
- [10] R. Luxemburg. “Culture Shock: Audio-reactive Interpolation Script,” Gist. (), [Online]. Available: <https://gist.github.com/rolux/48f1da6cf2bc6ca5833dbacbf852b348>.
- [11] M. Siegelman. “The Deep Music Visualizer: Using sound to explore the latent space of BigGAN,” Medium. (Oct. 10, 2019), [Online]. Available: <https://towardsdatascience.com/the-deep-music-visualizer-using-sound-to-explore-the-latent-space-of-biggan-198cd37dac9a>.
- [12] M. Siegelman, *Deep Music Visualizer*, Jun. 10, 2022. [Online]. Available: <https://github.com/msieg/deep-music-visualizer>.
- [13] M. Alafriz, *Lucid Sonic Dreams*, Jun. 9, 2022. [Online]. Available: <https://github.com/mikaelalafriz/lucid-sonic-dreams>.

- [14] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," presented at the International Conference on Learning Representations, Sep. 27, 2018. [Online]. Available: <https://openreview.net/forum?id=B1xsqj09Fm>.
- [15] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 1, 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. [Online]. Available: <https://doi.org/10.1007/s11263-015-0816-y>.
- [16] D. Ellis. "Chroma Feature Analysis and Synthesis." (), [Online]. Available: <https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/>.
- [17] H. Brouwer. "Audio-reactive Latent Interpolations with StyleGAN." (Dec. 11, 2020), [Online]. Available: <https://jcbrouwer.github.io/audio-reactive-stylegan>.
- [18] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," p. 10,
- [19] T. Karras, M. Aittala, S. Laine, *et al.*, "Alias-Free Generative Adversarial Networks," p. 12,
- [20] T. Broad, F. F. Leymarie, and M. Grierson, "Network Bending: Expressive Manipulation of Deep Generative Models," Mar. 12, 2021. arXiv: 2005.12420 [cs]. [Online]. Available: <http://arxiv.org/abs/2005.12420>.
- [21] D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba, "Rewriting a Deep Generative Model," Jul. 30, 2020. arXiv: 2007.15646 [cs]. [Online]. Available: <http://arxiv.org/abs/2007.15646>.
- [22] A. Davis and M. Agrawala, "Visual rhythm and beat," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, Aug. 10, 2018, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3197517.3201371. [Online]. Available: <https://dl.acm.org/doi/10.1145/3197517.3201371>.
- [23] P. Robert and Y. Escoufier, "A Unifying Tool for Linear Multivariate Statistical Methods: The RV- Coefficient," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 25, no. 3, pp. 257–265, 1976, ISSN: 0035-9254. DOI: 10.2307/2347233. JSTOR: 2347233.
- [24] A. K. Smilde, H. a. L. Kiers, S. Bijlsma, C. M. Rubingh, and M. J. van Erk, "Matrix correlations for high-dimensional data: The modified RV-coefficient," *Bioinformatics (Oxford, England)*, vol. 25, no. 3, pp. 401–405, Feb. 1, 2009, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btn634. pmid: 19073588.
- [25] A. Morcos, M. Raghu, and S. Bengio, "Insights on representational similarity in neural networks with canonical correlation," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. [Online]. Available: <https://papers.nips.cc/paper/2018/hash/a7a3d70c6d17a73140918996d03c014f-Abstract.html>.
- [26] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar. 1, 1966, ISSN: 1860-0980. DOI: 10.1007/BF02289451. [Online]. Available: <https://doi.org/10.1007/BF02289451>.
- [27] F. Ding, J.-S. Denain, and J. Steinhardt, "Grounding Representation Similarity Through Statistical Testing," presented at the Advances in Neural Information Processing Systems, May 21, 2021. [Online]. Available: [https://openreview.net/forum?id=\\_kwj6V53ZqB](https://openreview.net/forum?id=_kwj6V53ZqB).

- [28] U. G. Indahl, T. Næs, and K. H. Liland, "A similarity index for comparing coupled matrices," *Journal of Chemometrics*, vol. 32, no. 10, e3049, 2018, ISSN: 1099-128X. DOI: 10.1002/cem.3049. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cem.3049>.
- [29] G. Mordant and J. Segers, "Measuring dependence between random vectors via optimal transport," *Journal of Multivariate Analysis*, vol. 189, p. 104912, 2022, ISSN: 0047-259X. [Online]. Available: <https://dial.uclouvain.be/pr/boreal/object/boreal:254444>.
- [30] M. Alafriz. "Introducing "Lucid Sonic Dreams": Sync GAN Art to Music with a Few Lines of Python Code!" Medium. (Mar. 14, 2021), [Online]. Available: <https://towardsdatascience.com/introducing-lucid-sonic-dreams-sync-gan-art-to-music-with-a-few-lines-of-python-code-b04f88722de1>.
- [31] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: <https://aclanthology.org/D14-1179>.
- [32] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 15, 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [33] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," Mar. 2, 2022. arXiv: 2201.03545 [cs]. [Online]. Available: <http://arxiv.org/abs/2201.03545>.
- [34] J. Sakuma, T. Komatsu, and R. Scheibler, "MLP-ASR: Sequence-length agnostic all-MLP architectures for speech recognition," Feb. 17, 2022. arXiv: 2202.08456 [cs, eess]. [Online]. Available: <http://arxiv.org/abs/2202.08456>.
- [35] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: <https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- [36] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's Raw! Audio Generation with State-Space Models," Feb. 19, 2022. arXiv: 2202.09729 [cs, eess]. [Online]. Available: <http://arxiv.org/abs/2202.09729>.
- [37] R. A. Gonsalves. "Creating Abstract Art with StyleGAN2 ADA," Medium. (Oct. 23, 2021), [Online]. Available: <https://towardsdatascience.com/creating-abstract-art-with-stylegan2-ada-ea3676396ffb>.
- [38] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 29, 2017. arXiv: 1412.6980 [cs]. [Online]. Available: <http://arxiv.org/abs/1412.6980>.
- [39] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Re, "HiPPO: Recurrent Memory with Optimal Polynomial Projections," Oct. 22, 2020. arXiv: 2008.07669 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/2008.07669>.
- [40] *Legendre polynomials*, in *Wikipedia*, Jun. 12, 2022. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Legendre\\_polynomials&oldid=1092784128](https://en.wikipedia.org/w/index.php?title=Legendre_polynomials&oldid=1092784128).

- [41] A. Paszke, S. Gross, F. Massa, *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” p. 12,
- [42] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry, *FFCV: Fast Forward Computer Vision (and other ML workloads!)* FFCV, Jun. 15, 2022. [Online]. Available: <https://github.com/libffcv/ffcv>.
- [43] D. M. L. Community, *Decord*, Distributed (Deep) Machine Learning Community, Jun. 15, 2022. [Online]. Available: <https://github.com/dmlc/decord>.
- [44] H. Brouwer, *Maua: Deep learning toolkit for image, video, and audio synthesis*, Maua, Jun. 14, 2022. [Online]. Available: <https://github.com/maua-maua-maua/maua>.
- [45] B. McFee, C. Raffel, D. Liang, *et al.*, “Librosa: Audio and Music Signal Analysis in Python,” 2015. DOI: 10.25080/MAJORA-7B98E3ED-003.
- [46] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new Python Audio and Music Signal Processing Library,” May 23, 2016. arXiv: 1605.07008 [cs]. [Online]. Available: <http://arxiv.org/abs/1605.07008>.
- [47] R. Hataya. “Moskomule/anatome:  $\square\alpha\tau\omicron\mu\eta$  is a PyTorch library to analyze representation of neural networks.” (), [Online]. Available: <https://github.com/moskomule/anatome>.
- [48] K. H. Liland, *Matrix Correlation Coefficients - MatrixCorrelation*, Apr. 18, 2022. [Online]. Available: <https://github.com/khliland/MatrixCorrelation>.
- [49] *Fréchet distance*, in *Wikipedia*, Apr. 10, 2022. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Fr%C3%A9chet\\_distance&oldid=1081884554](https://en.wikipedia.org/w/index.php?title=Fr%C3%A9chet_distance&oldid=1081884554).
- [50] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium,” p. 12,
- [51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.
- [52] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments,” Jan. 8, 2021. arXiv: 2006.09882 [cs]. [Online]. Available: <http://arxiv.org/abs/2006.09882>.
- [53] S. Morozov, A. Voynov, and A. Babenko, “On Self-Supervised Image Representations for GAN Evaluation,” presented at the International Conference on Learning Representations, Sep. 28, 2020. [Online]. Available: <https://openreview.net/forum?id=NeRdBETionN>.
- [54] X. Steenbrugge. “WZRD.” (), [Online]. Available: <https://wzrd.ai/>.
- [55] W. S. Cleveland, “Robust Locally Weighted Regression and Smoothing Scatterplots,” *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, Dec. 1979, ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.1979.10481038. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/01621459.1979.10481038>.
- [56] I. Tolstikhin, N. Houlsby, A. Kolesnikov, *et al.*, “MLP-Mixer: An all-MLP Architecture for Vision,” Jun. 11, 2021. arXiv: 2105.01601 [cs]. [Online]. Available: <http://arxiv.org/abs/2105.01601>.

- [57] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, "Spleeter: A fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, Jun. 24, 2020, ISSN: 2475-9066. DOI: 10.21105/joss.02154. [Online]. Available: <https://joss.theoj.org/papers/10.21105/joss.02154>.
- [58] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - A Reference Implementation for Music Source Separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, Sep. 8, 2019, ISSN: 2475-9066. DOI: 10.21105/joss.01667. [Online]. Available: <https://joss.theoj.org/papers/10.21105/joss.01667>.
- [59] Google. "Google Colaboratory." (), [Online]. Available: [https://colab.research.google.com/?utm\\_source=scs-index](https://colab.research.google.com/?utm_source=scs-index).
- [60] Z. Liu, *A ConvNet for the 2020s*, Meta Research, Jun. 16, 2022. [Online]. Available: <https://github.com/facebookresearch/ConvNeXt>.
- [61] P. Wang, *Lucidrains/x-transformers*, Jun. 16, 2022. [Online]. Available: <https://github.com/lucidrains/x-transformers>.
- [62] K. Goel and A. Gu, *Structured State Spaces for Sequence Modeling*, HazyResearch, Jun. 15, 2022. [Online]. Available: <https://github.com/HazyResearch/state-spaces>.
- [63] A. Gu and T. Dao, *HiPPO*, HazyResearch, Jun. 14, 2022. [Online]. Available: <https://github.com/HazyResearch/hippo-code>.
- [64] F. Mémoli, "Gromov–Wasserstein Distances and the Metric Approach to Object Matching," *Foundations of Computational Mathematics*, vol. 11, no. 4, pp. 417–487, Aug. 2011, ISSN: 1615-3375, 1615-3383. DOI: 10.1007/s10208-011-9093-5. [Online]. Available: <http://link.springer.com/10.1007/s10208-011-9093-5>.
- [65] T. Vayer, R. Flamary, R. Tavenard, L. Chapel, and N. Courty, "Sliced Gromov-Wasserstein," Dec. 11, 2020. arXiv: 1905.10124 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1905.10124>.
- [66] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive Learning for Unpaired Image-to-Image Translation," Aug. 20, 2020. arXiv: 2007.15651 [cs]. [Online]. Available: <http://arxiv.org/abs/2007.15651>.
- [67] J. Paul, B.-S. Michael, M. Pedro, *et al.*, "PSA-GAN: Progressive Self Attention GANs for Synthetic Time Series," Aug. 2, 2021. arXiv: 2108.00981 [cs]. [Online]. Available: <http://arxiv.org/abs/2108.00981>.
- [68] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised Scalable Representation Learning for Multivariate Time Series," Jan. 3, 2020. arXiv: 1901.10738 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1901.10738>.
- [69] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 11 531–11 539, ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.01155. [Online]. Available: <https://ieeexplore.ieee.org/document/9156697/>.



- [70] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," in *Computer Vision – ECCV 2018*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11211, Cham: Springer International Publishing, 2018, pp. 3–19, ISBN: 978-3-030-01233-5 978-3-030-01234-2. DOI: 10.1007/978-3-030-01234-2\_1. [Online]. Available: [http://link.springer.com/10.1007/978-3-030-01234-2\\_1](http://link.springer.com/10.1007/978-3-030-01234-2_1).
- [71] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, "RoFormer: Enhanced Transformer with Rotary Position Embedding," Oct. 8, 2021. arXiv: 2104.09864 [cs]. [Online]. Available: <http://arxiv.org/abs/2104.09864>.

# A

## Model Details

### A.1. Learned Decoder

The learned decoder is a 2-layer MLP. This consists of a linear layer which doubles the number of channels, then a GELU activation, and finally a linear layer which maps to the number of channels in the latent vectors (512).

The implementation uses PyTorch’s functional API to apply this operation directly for multiple latent sequences per batch element (generally there are 3 sequences per interpolation).

### A.2. Parameter-free Decoder

The parameter-free decoder must receive the correct number of envelopes based on the number of output sequences that are used. Across all experiments the parameter-free decoder is configured to output 3 latent sequences (shallow, middle, and deep) with a 3-way average per sequence and 4 noise sequences (4x4, 8x8, 16x16, and 32x32). This means that 17 envelopes are needed to synthesize all output sequences.

### A.3. Envelope Generator

For all envelope generators, unless stated otherwise, a depth of 4 is used. The hidden size is set to 17 to be compatible with the parameter-free decoder.

- **GRU** PyTorch’s implementation of the Gated Recurrent Unit [31].
- **LSTM** PyTorch’s implementation of the Long-Short Term Memory recurrent network [32].
- **ConvNeXt** FAIR’s implementation [60] of ConvNeXt [33], adapted to use 1-dimensional temporal convolutions rather than 2-dimensional spatial convolutions.
- **MLP-ASR** the MLP-Mixer based architecture introduced by Sakuma, Komatsu, and Scheibler [34], reimplemented manually. Configured with a channel multiplier of 2 in the Convolutional Gating Unit.
- **Transformer** Transformer [35] encoder implementation from lucidrains’s x-transformers [61]. Configured with 4-headed attention with a dimension of 4 each (to ensure parameter count is more comparable with other models).
- **Sashimi** HazyResearch’s standalone implementation [62] of Sashimi [36]. Configured with no channel number expansion in the base model or feedforward layers (to ensure parameter count is more comparable with other models).

## A.4. HiPPO

HazyResearch’s standalone implementation [63] of HiPPO [39] is used. Tests are done to encode audio features like the onsets and RMS using both the scale and translation invariant versions. The scale invariance is better at reproducing the input envelopes and so is used in all experiments.

To prevent extreme values near the start and end of envelopes (which are caused by extreme values near the edges of some Legendre polynomials) the envelope parameterizations are padded on both sides with zeros before encoding and cropped back to the correct shape on the forward pass.

# B

## Negative / Inconclusive Results

**Gromov-Wasserstein Distance** The Gromov-Wasserstein distance [64] (and variants such as the Sliced GWD [65]) is a distance between data in different metric spaces based on optimal transport theory. This seemed like a natural fit for an audiovisual distance metric. However, early experiments similar to Section 3.4 seemed to show an inverse correlation with expectations.

**Patch Contrastive Loss** The first experiments were run using a patch contrastive loss as introduced by Park, Efros, Zhang, *et al.* [66]. The idea is to compare slices of the full timeseries and learn to optimize internal representations such that patches that overlap have a high similarity and patches that do not overlap have low similarity. This was sidelined early on due to the implementation complexity but seems like a very promising loss to adapt to the final models.

**Timeseries GAN (PSAGAN)** During supervised training experiments using the Progressive Self-attention GAN (PSAGAN) was investigated as a way to improve performance [67]. GANs tend to perform significantly better than simple MSE loss training when their training can be stabilized correctly. However, GAN training is also much more computationally expensive and so this path was dropped in favor of a quicker iteration time.

**Context FID Evaluation** The PSAGAN paper [67] introduces an FID-like metric called the Context FID to evaluate the quality of timeseries generations. The metric is based on the unsupervised timeseries training framework introduced by Franceschi, Dieuleveut, and Jaggi [68] which means that it can be tailored to any timeseries dataset without requiring a pre-trained network like the regular FID (such a pre-trained network is harder to find for timeseries as the similarity across different types of timeseries is less than between different image domains). This option was discarded due to requiring a large dataset of good examples to compare with, which is already a weakness of the supervised methods in general. The results of the unsupervised network training were also inconclusive and probably require a better search for correct hyperparameters which is expensive.

**Latent Augmentor** An idea for alleviating the problem of the small supervised dataset was to augment the training data with randomly generated audio-reactive interpolations. The Latent Augmentor was essentially a simple version of the `Randomizer` synthesizer that was run during training. This was unsuccessful as the quality of the random interpolations was not very high and so not a useful training signal for the model.

**Latent Channel Permutation** Another augmentation method. The channels of the latent sequences were permuted. The idea was that these would be equally valid latent

sequences just rotated to move in different directions in latent space. Unfortunately, the different channels of the latent vector do not all have the same range and statistics and so results generated with permuted latents did not resemble the regular latent distribution enough to achieve high quality image results.

**Re-weighting the Audiovisual Correlation** To try to offset the bias towards smoother audio features observed in Figure 3.3, multiple re-weighting schemes were investigated. The idea was to penalize the correlation score of features which were smooth and boost the correlation score of features that were not. One scheme was to multiply the correlation value with the standard deviation of the feature over time. Another scheme was to multiply by the mean of the absolute difference over time of the audio feature. However, both of these schemes suffered from large differences in scale between different features and it was unclear how to normalize the scaling factors to be comparable. A final re-weighting scheme divided the correlation score by the median value of the normalized auto-correlation matrix. While this scheme produced correlation weightings that seemed to properly penalize and boost scores by their smoothness, the results of the experiments in Section 3.4 did not differ significantly.

**Absolute Difference Self-supervised Loss** To encourage matching changes in the output video to changes in the audio features, the self-supervised loss was computed on the absolute difference of the features rather than the features themselves. Models trained with this loss seemed to function correctly (i.e. not completely broken), but were not pursued further as it was intuitively unclear what would make this formulation of the loss preferable.

**Pyramidal Convolutional Envelope Generator** An envelope generator which used a U-net-like sequence of 1-dimensional temporal convolutions with downsampling operations until half the depth and upsampling operations in the second half. This network would be able to incorporate different scales of temporal correlations in its different layer and hopefully benefit from this hierarchical separation of analysis. This envelope generator might be especially suited to longer sequence lengths which were not explored in our experiments.

**Convolutional Learned Decoder** The learned decoder used in all experiments was a channelwise multilayer perceptron. This does not incorporate temporal correlations in its translation of the hidden space timeseries to the output latent space timeseries. Another learned decoder was investigated that used a two-layer 1-dimensional convolution instead. This was not pursued extensively in final experiments due to its higher implementation complexity. Early experiments seemed promising, but not better than the parameter-free decoder.

**Efficient Channel Attention** Efficient Channel Attention blocks [69] were added to the envelope generator and learned decoder to improve channelwise mixing of information.

**Convolutional Block Attention** Convolutional Block Attention [70] was added to the envelope generator and learned decoder to improve channelwise mixing of information.

**Rotary Positional Embeddings** Rotary positional encoding [71] was used in the Transformer encoder layers. This is a positional encoding that encodes relative positional information via 2-dimensional rotations of the input sequence. There is some evidence that suggests it improves Transformer convergence and long-term sequence generations.

**Spatial Correlations in Noise** The first learned decoders for noise sequences used 3-dimensional convolutions to also learn spatial correlations in the noise maps. This might allow the synthesizer to learn to segment different audio-reactive elements to different spatial locations. However, due to the noisy nature of the noise maps, the training data did not have much correlation to learn from. Furthermore, these 3D-convolutional models were very expensive to train.

**Softmax Groups in Parameter-free Decoder** To try to add an inductive bias to the parameter-free decoder towards larger changes in style between sections, the latent averaging envelopes were grouped together and the softmax operation applied. This would ensure that within each group, there was one dominant latent visually without the envelopes needing to be as extreme without the softmax operation. The results were not significantly different from the decoder without softmax operation.

**Section Segmentation Loss with Allocation Solver** Another self-supervised loss was designed which worked by comparing section segmentations of the output sequences and the audio features. The problem with these segmentations is that the assigned label values would not be the same across different feature segmentations. There for the maximum agreement permutations of the labels was found using a Linear Assignment Problem solver, the auction algorithm. While the loss seemed theoretically promising and was implemented to be differentiable, the LAP solver had numerical instability issues, did not always converge to a valid permutation, and was significantly slower than the audiovisual correlation.

**Gradient Normalization** Gradient normalization is a technique where the gradients are normalized by their L1 norm at a certain point in the backwards pass. This can help ensure that the scale of gradients from multiple different optimization targets are free from scaling issues. This was used to equalize the gradient contribution of the individual noise and latent sequences (which had different scales due to the difference in size and value of the sequence tensors). Unfortunately there was no clear advantage over the non-normalized loss.

**Prediction Similarity Penalty** To prevent output sequences from all tending towards having the same autocorrelation, a penalty was added that enforced each noise and latent sequence to have a different autocorrelation. This penalty was the absolute cosine correlation between each sequence's autocorrelation. This did seem to encourage a more varied reaction within each sequence, however, it did not have an impact on the overall audio-reactivity.