FACULTY OF ENGINEERING AND ARCHITECTURE

Analysing and Improving Visual and Semantic Quality of Realtime Audio Visualisations with Deep Generative Models

Arthur Deleu
Student number: 01711291

Supervisors: Prof. dr. Tijl De Bie, Prof. dr. Dirk Deschrijver
Counsellor: Ir. Lorin Werthen-Brabants

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2022-2023

GHENT UNIVERSITY

Preface

"The Buddha, the Godhead, resides quite as comfortably in the circuits of a digital computer or the gears of a cycle transmission as he does at the top of the mountain, or in the petals of a flower. To think otherwise is to demean the Buddha - which is to demean oneself." – *Robert M. Pirsig, Zen and the Art of Motorcycle Maintenance: An Inquiry Into Values (1974)*

This master's thesis would not have been possible without the help of a number of important people.
First and foremost, I'd like to thank my counsellor Lorin, for the time he took to guide me in this process, for his patience, for the regular meetings and helpful advice. It has been a pleasure working with you.

Secondly, I should thank Kasper for critically questioning the artistic side of this project when joining our meetings, for sharing his inspiring take on our field and offering me the opportunity to perform with the system I created.

I would like to express my gratitude to Jonas Kraasch and Phillipe Pasquier at the Simon Fraser University's School of Interactive Arts and Technology in Vancouver, for kindly introducing me to their work and taking the time to discuss the subject during our phone conversations.

Although our journey together was short, it's thanks to Cedric De Boom, who believed in this project from the start, that I had the chance to work on this subject. By setting up numerous meetings with research groups and labs, he played a pivotal role in helping me connect with the right mentors to kick off the project. I thank him for that.

At last, I'd like to thank some people who stood close to me this past year. These past years.

My parents, for being there, no matter what.
The road's been long and not without obstacles.
I have to thank them for their patience, their advice, their ever-sincere warmth. None of this would have been possible without them.

I want to thank my sisters - Emma & Alice - for showing me a life worth living.
Each in their unique own way.
For always being there, for the tremendous inspiration, for generously sharing their worlds with mine. I can not thank them enough for that.

Tomas, for reading my manuscript, for providing intelligent feedback. For being the close friend he was these last years.
My roommates - Wannes, Britt & Francis - for, each in their own right, creating a place I loved to come home to. It's been a blessing.
Pinto, for the countless hours of brain breaking during summers, winters, springs. For the music.

The Alugang.
The people at the library.
My friends, my family.
I owe you
"Thank You".

Copyright statement

Arthur Deleu
August 14, 2023

Clarification of the oral exposition

This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

Analysing and Improving Visual and Semantic Quality of Realtime Audio Visualisations with Deep Generative Models

Arthur Deleu

Supervisors: Prof. dr. Tijl De Bie
Prof. dr. Dirk Deschrijver
Counsellor: Ir. Lorin Werthen-Brabants

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering
Academic year 2022-2023

Abstract

In the last few years, deep generative modelling made its way to the public. The branch of deep learning specialized in the generation of novel data is already transforming the professional lives of writers, visual artists and musicians worldwide. In this work, a system is designed that employs deep generative models as real-time audio visualisers for live music visualisation, aiming to extend the toolkit of modern video jockeys (VJs). For the first time in literature, we quantify and benchmark the performance of existing systems in terms of latency, frame rate, computational complexity, visual quality, diversity, stability, semantic quality, entanglement, and content coherence, each through their own metrics. In order to quantify content coherence, a novel metric is designed and employed in this project, measuring the difference between the *Root Mean Square Variability* and *Temporal Motion Variability Distance* of audiovisual data. With the insights gathered from quantitative analysis, new design proposals are made targeting the interpolation algorithm, synthesis model, and control mechanisms. The *Hypersphere Interpolation* algorithm proposed in this study, presents a novel technique to create audio reactive walks in the latent space of generative model, through describing a trajectory on the surface of an n-sphere. The technique delivers promising results in terms of latency, stability, and control. Moreover, β-VAE models are implemented as synthesis models for the first time in current literature, with promising results in terms of latency, frame rate, semantic control and latent space disentanglement. Through both quantifying the current state-of-the-art thoroughly for the first time and subsequently formulating design advancements, this project is believed to have contributed to the young field of real-time audio reactive generative modelling and the field of live visualisation as a whole. The thus created software was used for live performances throughout Spring 2023 and shows promising practical results.

Index Terms: Deep generative modelling, real-time audio visualisation, latent space exploration, VJing

Analysing and Improving Visual and Semantic Quality of Realtime Audio Visualisations with Deep Generative Models

Arthur Deleu
Supervisors: Prof. dr. Tijl De Bie, Prof. dr. Dirk Deschrijver
Counsellor: Ir. Lorin Werthen-Brabants

***Abstract*—In the last few years, deep generative modelling made its way to the public. The branch of deep learning specialized in the generation of novel data is already transforming the professional lives of writers, visual artists and musicians worldwide. In this work, a system is designed that employs deep generative models as real-time audio visualisers for live music visualisation, aiming to extend the toolkit of modern video jockeys (VJs). For the first time in literature, we quantify and benchmark the performance of existing systems in terms of latency, frame rate, computational complexity, visual quality, diversity, stability, semantic quality, entanglement, and content coherence, each through their own metrics. In order to quantify content coherence, a novel metric is designed and employed in this project, measuring the difference between the *Root Mean Square Variability* and *Temporal Motion Variability Distance* of audiovisual data. With the insights gathered from quantitative analysis, new design proposals are made targeting the interpolation algorithm, synthesis model, and control mechanisms. The *Hypersphere Interpolation* algorithm proposed in this study, presents a novel technique to create audio reactive walks in the latent space of generative model, through describing a trajectory on the surface of an n-sphere. The technique delivers promising results in terms of latency, stability, and control. Moreover, β-VAE models are implemented as synthesis models for the first time in current literature, with promising results in terms of latency, frame rate, semantic control and latent space disentanglement. Through both quantifying the current state-of-the-art thoroughly for the first time and subsequently formulating design advancements, this project is believed to have contributed to the young field of real-time audio reactive generative modelling and the field of live visualisation as a whole. The thus created software was used for live performances throughout Spring 2023 and shows promising practical results.**

***Index Terms*—Deep generative modelling, real-time audio visualisation, latent space exploration, VJing**

I. Introduction

Video plays a profound role in modern digital society, from video conferences and lectures to streaming services and short clips on various social media. Some researchers even state that recent digital media technology contributes to a rapidly evolving, strongly visual world language [1].

Already since the 1960s, VJing, the art of mixing and generating video in real-time to accompany a live musical performance, has been a part of that movement and is still gaining popularity in the global club and festival scene with ever-more creative and immersive visualisations being created through dedicated VJing software packages.

At the same time, recent advances in the field of machine learning and deep generative modelling made the creation of novel data possible at unprecedented levels, with recent generative models already changing the lives of writers, visual artists and musicians worldwide.

In this study, a VJing system is proposed that generates real-time music visualisations based on audio reactive walks in the latent space of generative models, improving and extending the current state-of-the-art.

In order to do this, a novel metric was designed to measure the content coherence of an audiovisual data: *Root Mean Square Variability - Temporal Motion Variability Distance* (RMSV-TMV). Along with other existing metrics from literature, the RMSV-TMV made it possible to quantify the performance of the current state-of-the-art to an unprecedented extent. Utilizing insights from this quantitative analysis, novel design concepts were proposed to enhance the interpolation algorithm, synthesis model, and control mechanisms. The study introduces the *Hypersphere Interpolation* algorithm, offering a novel approach for creating audio-reactive trajectories within the generative model's latent space by tracing a path on the surface of an n-sphere. This technique demonstrates favourable outcomes in terms of speed, stability, and control. Additionally, the incorporation of β-VAE models as synthesis models offers a novel approach and visual look, while showcasing promising outcomes in latency, frame rate, semantic control, and latent space disentanglement. The created system was put to use on multiple live occasions, with promising audience feedback.

The organization of the paper is as follows. In Section II, we introduce the necessary background to the reader in order to be able to understand every aspect of this study. In section III, we discuss the constraints imposed on a real-time visualisation system and the metrics used to assess the performance of such a system, including the novel *RMSV-TMV distance* to quantify the music-video coherence. Section IV presents the proposed techniques, divided in *Feature Extracting & Signal Processing*, *Hypersphere Interpolation*, *β-VAE as the Synthesis Model* and *Semantic Control*. In Section V, the findings of this study are discussed, and final thoughts are reported.

II. Background

The field of deep generative modelling itself is still quite young, with the first papers on Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) released in 2013 and 2014 respectively [2], [3]. Consequently, the existing work on employing deep generative models for audio visualisation is scarce.

Although different in training procedure, both GAN and VAE models first learn a latent representation of the training data and train a generator network to infer novel images from these representations. Through interpolating between these latent vectors, smooth visualisations can be generated.

Brouwer [4] makes these interpolations audio reactive, by extracting the chromagram, representing the frequency content of an audio input, and matching the 12 notes of the Western musical scale to 12 latent representations in the StyleGAN2 [5]. At every time instant, a weighted sum is calculated of the latent vectors linked to the notes being played. By interpolating in between those weighted sums over time, audio reactive latent interpolations are generated.

Kraasch and Pasquier [6] make this approach real-time, by simplifying the signal processing and audio reactive interpolation algorithm. Instead of extracting the chromagram at each time instant, the *Root Mean Square* (RMS) amplitude is extracted and multiplied with the step size of a random walk in the StyleGAN2 latent space. This way, greater visual change is associated with greater amplitude change. The RMS over an audio fragment gets calculated according to Equation 1.

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\left({x_1}^2 + {x_2}^2 + \cdots + {x_n}^2\right)}, \tag{1}$$

with x_i representing the amplitude at time instant i.

The algorithm introduced by Kraasch and Pasquier [6] will be referred to as the *Latent Space Traversal* algorithm in this study. The performance of this system was assessed on an NVIDIA QUADRO 5000 with 16GB of VRAM, resulting in frame rates of 24 FPS at a resolution of 512 $\times$ 512.

III. Constraints & Metrics

A. Constraints

A VJ accompanies musical performances with live imagery. The most important requirement of that imagery is that it needs to be in sync with the music. The visualisations moreover need to be generated in real-time.

For a video to appear smooth in real-time, two concepts are important: the *critical flicker fusion threshold* (CFFT) and the lip-sync error (LSE).

The CFFT is the threshold above which a sequence of images cannot be perceived on an individual basis any longer, effectively making it appear as smooth video. The CFFT of the *Human Visual System* (HVS) lies between 60 Hz and 90 Hz on average [7].

The *lip-sync error* (LSE) or audio-video delay refers to the relative timing of audio and video fragments.

Stemming from the field of television broadcasting, thresholds have been measured below which an audience cannot discern drift between audio and video. Different standards exist, with the ITU-R BT.1359 stating that audio should not lead video by more than 90 ms before delays are unacceptable and the ATSC stating that video can not lag audio with more than 15 ms [8], [9]. Kraasch and Pasquier keep a middle-ground, and state delay in modern VJing applications should be kept below 50 ms [6].

Apart from being in sync with the music, the visualisations need to have semantic value and should respond to the energy of an audience. For that reason, an artist needs to have the option to control system parameters in real-time.

B. Metrics

In essence, the quality of a VJing system is subjective: if an audience likes what it perceives, a visual performance is considered successful.

However, this study attempts to assess the performance of the designed system in terms of latency (ms), frame rate & computational complexity (FPS), visual quality & diversity (FID), stability, semantic quality, entanglement (PPL), and coherence through empirical measurement. This is done to an extent, unprecedented in existing literature.

To quantify coherence, or the alignment between audio and visualisation, a new metric had to be designed: the *RMSV-TMV distance*. By calculating the distance between the *RMS Variability* (RMSV) and the *Temporal Motion Variability* (TMV), both normalized over the length of the whole audiovisual fragment, an error value is calculated that quantifies the degree of misalignment between both.

The RMSV gets calculated according to Equation 2.

$$\mathrm{RMSV} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{\mathrm{RMS},i} - \bar{x}_{\mathrm{RMS}})^2, \tag{2}$$

with x_{RMS} calculated according to Equation 1.

To quantify the TMV, the Färneback algorithm is used to measure the optical flow between consecutive frames [10]. The motion degree of a video window is calculated as the sum of the norm of all motion vectors between consecutive frames, calculated through the Färneback algorithm.

The average RMSV-TMV distance is formulated in Equation 3.

$$\mathbb{E}[\text{RMSV-TMVD}] = \frac{1}{n}\sum_{i=1}^{n}\left|\mathrm{RMSV}_i - \mathrm{TMV}_i\right|, \tag{3}$$

with n the amount of video fragments.

The performance of the metric was assessed by calculating the RMSV-TMV distance of the first 20 seconds of an audio-video sync test video [11]. The clip shows a white square every other second, combined with a short beeping sound. Audio and video are in perfect sync. From Figure 1 it is clear that the RMSV and TMV values follow the same trends. However, although both are in sync, the RMSV values vary a lot more than the TMV ones. Due to normalization, this leads to some RMSV peaks being completely suppressed with a bigger RMSV-TMV distance as a consequence. This shows that the

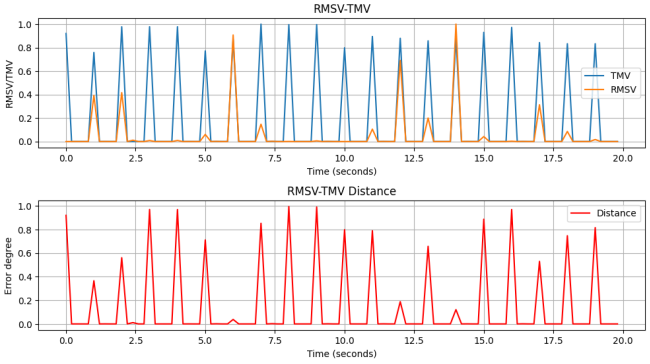


Fig. 1: RMSV-TMV metric assessed on an audio-video sync test clip.

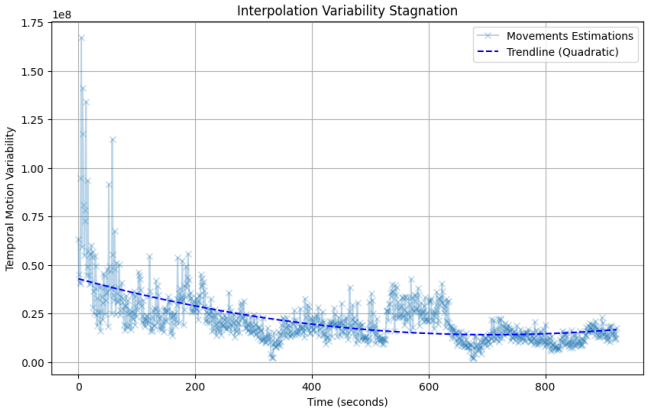


Fig. 2: The TMV between frames shows a downward trend over time, although a constant amplitude input is sent through the system.

RMSV-TMV distance gives a correct idea on the coherence of audio and video, although not perfectly accurate. Combining visual inspaction of the values of RMSV and TMV over time visually with looking at the $\mathbb{E}$[RMSV-TMVD] is the suggested option when evaluating coherence performance during this study. All measurements and visualisations in this project were run on an Nvidia Geforce RTX 3070 TI Laptop GPU with 8GB of VRAM.

IV. Proposed Techniques

A. Feature Extraction & Signal Processing

The audio was processed as a monophonic stream with a chunk size of 1024, a sampling rate of 44100 Hz and a Fast Fourier Transform (FFT) window size of 2048.
The RMS is extracted according to Equation 1 with n being the chunk size. The onset is detected through a real-time algorithm based on recurrent neural networks described by Böck et al., reaching a precision of 0.850 and a recall of 0.787 [12]. The onset refers to the beginning of a musical note or sound event and is used to modulate the noise injected into each layer of the StyleGAN2 model, leading to small jitters when musical events are detected.

B. Hypersphere Interpolation

The Latent Space Traversal algorithm based on random latent walks, proposed by Kraasch and Pasquier has two fundamental flaws, both linked to the inherent random nature of the algorithm [6].
First, although always different, the trajectory travelled by employing this algorithm has no notion of recurrence. However, repetition and recurrence constitute an important part of musical composition. A fissure might appear when the music repeats itself, but the visualisation does not.
The second issue, is the fact that image variation stagnates when the interpolation algorithm has been running for longer periods of time. This effect is visualised in Figure 2, through visualisation of the TMV over time and adding the trend line. In this study, it was concluded that the reason for this stagnation lies in Theorem IV.1.

Theorem IV.1 (Recurrence of random walks)**.** *A simple random walk in dimension d is recurrent for d = 1, 2 and transient for d >= 3 [13]*

Since the StyleGAN2 latent space counts 512 dimensions, a random walk will never be recurrent, making drift in a certain direction inevitable. However, Theorem IV.1 can not explain the effect by itself. In fact, if the StyleGAN2 latent space would have identical distributions of image variation anywhere in its space, a drifting walk should not lead to stagnating visualisations. We conclude that a possible explanation for this could be found in the manifold hypothesis, stating that real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space [14]. Applying this to the high dimensional latent spaces of GANs: it may be that near the edges of the latent space, the data variance is smaller than towards the centre.
In order to remove the random aspect of the interpolation and to build a notion of recurrence into the algorithm, the *Hypersphere Interpolation* algorithm draws latent trajectories on the surface of an n-sphere, with $n = 511$ when working with StyleGAN2 models. An n-sphere in a general $(n+1)$-dimensional Euclidean space can be described in spherical coordinates according to Equation 4.

$$\begin{aligned} x_1 &= r\cos(\phi_1) \\ x_2 &= r\sin(\phi_1)\cos(\phi_2) \\ x_3 &= r\sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ &\vdots \\ x_{n-1} &= r\sin(\phi_1)\ldots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ x_n &= r\sin(\phi_1)\ldots\sin(\phi_{n-1})\cos(\phi_n), \end{aligned} \tag{4}$$

where r represents the sphere radius and ϕ_i angles ranging over $[0, 2\pi)$. By varying all ϕ_i from 0 to 2π in steps of $\frac{2\pi}{P}$ with P the period, a spherical trajectory can be described in a latent space of any dimension $(n+1)$ with each dimension being traversed.
By interpolating on the surface of a hypersphere, a visuali-

sation loop is now introduced by default. This loop can be synchronized with the beats per minute (BPM) of the playing music, ensuring that a single visual loop aligns precisely with an integer number of beats in the music track. To achieve this, the traversal period P should be related to the BPM according to Equation 5.

$$\mathrm{P} = \frac{\mathrm{FPS}}{\mathrm{BPM} \times 60} \times \mathrm{N}, \tag{5}$$

with BPM the tempo of the music played, FPS the average amount of frames generated per second and N a chosen coefficient. The higher the N value, the more beats will be included in one visual period and the slower the visualisation will run by design.

In order to make the *Hypersphere Interpolation* audio reactive, 3 different visual parameters can be mapped to musical parameters: the period P, the radius r and the hypersphere centre $\mathbf{c} = [x_1, x_2, x_3, ..., x_{n-1}, x_n]$. For testing purposes, the RMS of an audio chunk moved the centre of the hypersphere in certain directions of the latent space, depending on its strength. These translations should happen in a few interpolation steps n_{frames}, in order to avoid changes to appear too abrupt. This leads to the definition of a minimum $\mathbf{LSE_{min}}$, representing the delay between a musical cue and its first visual reaction, and a peak $\mathbf{LSE_{max}}$, representing the delay between the musical cue and the appearance of its maximum visual response. Both are represented in Equation 6.

$$\begin{aligned} \mathbf{LSE}_{\text{min}} &= D_g \\ \mathbf{LSE}_{\text{max}} &= n_{\text{frames}} \cdot D_g \end{aligned} \tag{6}$$

Table I compares both algorithms quantitatively in terms of frame generation time T_g, LSE, frame rate and mean RMSV-TMVD. The *Latent Space Traversal* algorithm slightly outperforms the *Hypersphere Interpolation* algorithm on all accounts except for the $\mathbf{LSE_{min}}$ value, but the differences are small. Both algorithms meet the latency constraints mentioned by Kraasch and Pasquier and the ITU-R in terms of $\mathbf{LSE_{min}}$, they fail to meet the constraints set by the ATSC. Both frame rates fail to exceed the CFFT of 60 FPS, mentioned in Section III-A. The RMSV-TMV error is low for both algorithms, implying they both visualise the music in a coherent and reactive fashion. However, the *Latent Space Traversal* algorithm seems to slightly outperform the *Hypersphere Interpolation* algorithm in terms of coherence. A potential explanation could lie in the looping nature of the *Hypersphere Interpolation* algorithm, leading to omni-present motion, even when no audio input is observed. The hypersphere-based algorithm takes away the problem of transient walks, making the system stable over long time frames and thus making prolonged VJing sets possible. Since the *Hypersphere Interpolation* algorithm is recurrent, it allows the artist to represent repeating patterns in the music by matching the tempo with the interpolation period.

C. β-VAE as the Synthesis Model

Instead of confining inference to StyleGAN2 alone, in this study the potential of β-VAE models as synthesis networks has been researched [2]. By training both a β-VAE architecture and a StyleGAN2-ADA model on the same 128×128 CelebA a comparative performance study was performed, of which the results are visualised in Table II [15], [16].

Both models used *Hypersphere Interpolation*, with the only major difference in functionality being the smaller dimension of the β-VAE latent space. Since the β-VAE latent space counts 128 dimensions, a trajectory is followed on a 127-sphere.

When comparing the performance of both models, it is clear that the VAE model is promising as a synthesis network and worth further research in the context of real-time audio visualisation. In terms of FID, the β-VAE model scores worse than a StyleGAN2 model, with quite high FID scores for β-VAE models. This lies in line with visual inspection, upon which it quickly becomes clear that the images generated by the VAE model are of lower visual quality and less diverse, with blurring effects as the most prominent artefacts. Moreover, β-VAE models fail to perform at high resolutions. On the other hand, lower PPL values are measured for β-VAE models, with the lowest scores reported for high β values.

This shows that the VAE latent space is less entangled than the StyleGAN2 latent space, which is confirmed through visual inspection of the latent space interpolations. This not only creates smoother latent interpolations, but also allows for more intuitive semantic control, as will be further explained in Section IV-D. On top of that, the β-VAE inference process is faster by more than a factor 10 compared to the StyleGAN2 inference process, with frame rates of up to 794 FPS.

This creates opportunity for more extensive signal processing, running visualizations over communication networks, running live visualizations on less powerful GPUs and even running multiple models at the same time, allowing for smooth transitions between models trained on different datasets. When it comes to coherence, the RMSV-TMV distance measures slightly better coherence between the music and the visualizations when a StyleGAN2 synthesis model is used. On a more subjective level, using β-VAE models for audio visualization delivers a unique aesthetic, even though resolutions are low. The artist could make the choice to use this pixelated effect as a visual style, with the gain of having more intuitive live control over the semantic changes and the ability to operate at much higher frame rates, lower lip-sync error and increased reactiveness of the system.

D. Semantic Control

The visualisations generated during live performance should have a form of semantic value: the image sequences need to tell something meaningful. An artist wants to tell a story. Using the algorithms as is, will not lead to a visual narrative. In this study, three mechanisms have been implemented that make it easier for the visual artist to make their visualisations semantically valuable.

1) Finding meaningful directions in the Latent Space: The *Hypersphere Interpolation* algorithm was made audio reactive through translation in a certain latent space direction of the centre under the influence of the RMS. However, not

TABLE I: Performance Comparison of Audio Reactive Latent Interpolation Algorithms.

Algorithm	T_g	$LSE_{\textbf{min}}$	$LSE_{\textbf{max}}$	**Frame Rate**	$\mathbb{E}$**[RMSV-TMVD]**
Latent Space Traversal	**31 ms**	36.5 ms	5.5 ms + 31 ms $\cdot$ n_{frames}	**32 FPS**	**0.187(0.21)**
Hypersphere Interpolation	33.64 ms	**33.64 ms**	n_{frames} $\cdot$ 33.64 ms	30 FPS	0.194(0.16)

TABLE II: Performance Comparison of the Generative Models using Hypersphere Interpolation

Model	**FID**	**PPL$_{\text{min}}$**	**Frame Rate**	**GPU Util.**	$\mathbb{E}$**[RMSV-TMVD]**
StyleGAN2	**17.76**	14.32	77 FPS	**91.35** %	**0.17(0.18)**
β-VAE	116.33	**3.82**	**794 FPS**	95.28 %	0.23(0.16)

(a) Eigenvector 1: age.

(b) Eigenvector 5: eyecolor.

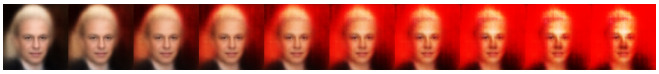
(c) Eigenvector 2: gender.

Fig. 3: 10 equidistant frames of 3 different visualisations where volume was mapped to different eigenvectors/directions in the FFHQ 512 $\times$ 512 latent space.

every direction is even remotely meaningful, and looking for meaningful directions of change in a 512- or 128-dimensional space by hand is like searching for a needle in a haystack.
In this project, an algorithm to find these meaningful directions of change in the StyleGAN2 latent space is used: The Semantic Factorization (SeFA) algorithm proposed by Shen and Zhou [17]. It is a closed-form, unsupervised approach, discovering meaningful factors of variation by only looking at the pre-trained weights of a model generator.
The SeFa algorithm returns a list of 512 eigenvectors that could be viewed as directions of change, sorted from most to least impactful. This way, the artist gets the possibility to choose from 512 pre-calculated direction vectors that are guaranteed to be impactful and semantically meaningful. A couple of eigenvectors and their effects on the FFHQ 512 $\times$ 512 dataset are visualised in Figure 3 [18]. Since these vectors are pre-calculated, they add no overhead on computation time.

As mentioned in the previous section, the β-VAE latent space is less entangled than the StyleGAN2. This disentangled nature of β-VAEs makes for easier control of semantically relevant features, since by design the β-VAE model forces single factors of variation upon single latent space dimensions. Simple cycles through all dimensions before a VJing set, allow the artist to make the music change latent vectors in the latent space directions of most interesting visual change. Moreover, if the model is trained with a high β, these semantic manipulations become additive, making combinations of semantic change

(a) Dimension 42, Degree +10: glasses added.



(b) Dimension 92, Degree= +10: background colour shifted to red.

(c) $\beta = 100$, Dimension 42 & 92, Degree +25 & +10: glasses added and background colour changed shifted to red

Fig. 4: Manipulating multiple latent dimensions at once leads to multiple semantic changes at once.

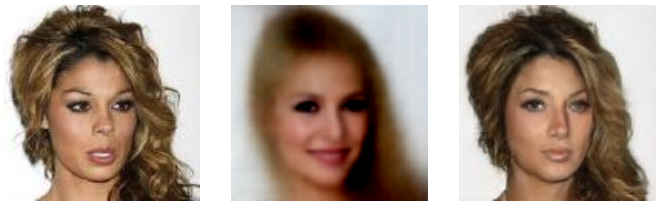
possible by changing multiple dimensions at the same time. This is visualised in Figure 4.

2) Latent Space Projection: Instead of trying to find interesting directions of change, an artist could try to find specific imagery in the StyleGAN2 latent space. To make this possible, GAN inversion was implemented, literally inverting the generator of the StyleGAN2 model [19]. The optimization problem that realizes this inversion is described in Equation 7.

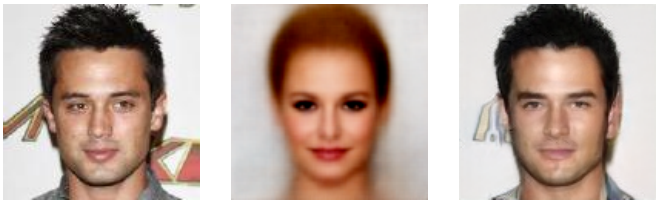
$$z^* = \min_z -\mathbb{E}_x \log[G(z)] \tag{7}$$

The learned filters of the VGG image classification model appear to be qualitative feature extractors and could be formalized as the perceptual loss [20]. This perceptual loss was shown to be a good loss function to perform gradient descent on during the inversion algorithm proposed above. Although slow, this way, the latent representation of a novel image closely resembling the original image can be found in the StyleGAN2 latent space. At the cost of higher perceptual losses, image projection is again more straightforward when using β-VAE models as synthesis models. Due to their encoder-decoder architecture, an image can just be fed through the encoder module to map an image to its latent representation. This is about as fast as inference from the decoder module, only taking 1.26 ms on average. Figure 5 shows results of latent space projection for both β-VAE and StyleGAN2 models. Through visual inspection and inspection of the perceptual losses, it should be clear that, although faster, StyleGAN2 projection outperforms β-VAE projection when it comes to accuracy.

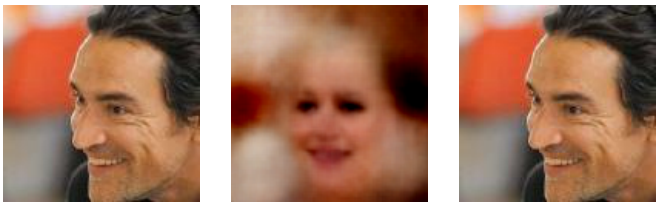
3) MIDI-control: In order to give an artist the ability to adapt model and algorithm parameters in real-time, MIDI-control was implemented in the final system. By mapping buttons and sliders to model and algorithm parameters, like hypersphere centre location, hypersphere radius, period, and noise strength,



(a) Original Image (b) β-VAE, PL=0.420 (c) StyleGAN2, PL=0.110



(d) Original Image (e) β-VAE, PL=0.384 (f) StyleGAN2, PL=0.112



(g) Original Image (h) β-VAE, PL=0.540 (i) StyleGAN2, PL=0.001

Fig. 5: Sample of images (left column) projected into the 128×128 β-VAE (middle column) and StyleGAN2 (right column) latent space, with their perceptual losses.

the artist can adapt the interpolation process intuitively, in real-time. They moreover get the option to switch in between pretrained models, eigenvectors, saved settings and resolutions.

V. Conclusion

In this study, we examined, reviewed and evaluated both the state-of-the-art and novel approaches to deep generative model based real-time audio visualisation, with the aim to add a tool to the arsenal of modern VJs. We designed a novel metric, the RMSV-TMV distance, measuring content coherence. Although the metric gave a fair indication of coherence, it is not perfect and should be subjected to further research. Implementing Dynamic Time Warping (DWT), cross-correlation between RMSV and TMV values and perceptual models all seem reasonable approaches for future fine-tuning of the metric. Real-time feature extraction was implemented to extract musical onset location and RMS. The *Latent Space Traversal* algorithm was thoroughly analysed on its performance and some fundamental flaws were discovered and reported. The novel *Hypersphere Interpolation* algorithm was designed to mitigate these flaws, with promising results in terms of latency, stability, and control. We employed β-VAE models in the context of real-time audio visualisation, for the first time in literature, with promising performance results. At the cost of visual quality and resolution, β-VAE based systems offer higher frame rates, smoother interpolations and less entanglement. They moreover offer more intuitive methods for latent space projection and semantic direction discovery, compared to the SeFA and GAN inversion algorithms needed for StyleGAN2 based systems [17], [19]. We however believe that the impact of this study has value beyond just an artistic one. We believe that by inviting people from different creative backgrounds to experiment with the latent spaces of generative models, novel insights and alternative perspectives on the subject could be cultivated.

References

[1] R. Oppenheimer, "Maximal art: The origins and aesthetics of west coast light shows," Apr 2009. [Online]. Available: https://rhizome.org/editorial/2009/apr/15/maximal-art-the-origins-and-aesthetics-of-west-coa/

[2] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2022, arXiv:1312.6114 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1312.6114

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," Jun. 2014. [Online]. Available: http://arxiv.org/abs/1406.2661

[4] H. Brouwer, "Audio-reactive Latent Interpolations with StyleGAN," Tech. Rep., 2020. [Online]. Available: http://arxiv.org/abs/1912.04958.

[5] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," Dec. 2019. [Online]. Available: http://arxiv.org/abs/1912.04958

[6] J. Kraasch and P. Pasquier, "Autolume-Live: Turning GANs into a Live VJing tool," *Proceedings of the 10th Conference on Computation, Communication, Aesthetics & X*, pp. 168–185, 2022, publisher: University of Porto.

[7] J. Davis, Y.-H. Hsieh, and H.-C. Lee, "Humans perceive flicker artifacts at 500hz," *Scientific Reports*, vol. 5, p. 7861, 2015.

[8] "Rec. itu-r bt.1359-1 1 recommendation itu-r bt.1359-1 relative timing of sound and vision for broadcasting," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:155097085

[9] Advanced Television Systems Committee, "ATSC Implementation Subcommittee Finding: Relative Timing of Sound and Vision for Broadcast Operations," 1750 K Street, N.W., Suite 1200, Washington, D.C. 20006, Technical Report IS-191, June 2003.

[10] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Image Analysis*, G. Goos, J. Hartmanis, J. van Leeuwen, J. Bigun, and T. Gustavsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2749, pp. 363–370, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/3-540-45103-X_50

[11] T. Test. (2016) Video in perfectsync(av sync test). YouTube video, 0:40, Posted October 20, 2016, accessed August 16, 2023. [Online]. Available: https://www.youtube.com/watch?v=d5TT12WUll0

[12] S. Böck, A. Arzt, F. Krebs, and M. Schedl, "Online real-time onset detection with recurrent neural networks," in *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12), York, UK*. sn, 2012, pp. 17–21.

[13] S. Popov, *Two-Dimensional Random Walk: From Path Counting to Random Interlacements*, ser. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2021.

[14] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," 2013.

[15] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training Generative Adversarial Networks with Limited Data," Oct. 2020, arXiv:2006.06676 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2006.06676

[16] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[17] Y. Shen and B. Zhou, "Closed-Form Factorization of Latent Semantics in GANs," Jul. 2020. [Online]. Available: http://arxiv.org/abs/2007.06600

[18] NVidia Corporation, "FFHQ dataset," https://github.com/NVlabs/ffhq-dataset, 2014, gitHub repository.

[19] A. Creswell and A. A. Bharath, "Inverting The Generator Of A Generative Adversarial Network," Nov. 2016, arXiv:1611.05644 [cs]. [Online]. Available: http://arxiv.org/abs/1611.05644

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

Contents

List of Figures

List of Tables

Acronyms

ADA Adaptive Discriminator Augmentation.

AI Artificial Intelligence.

ANN Artificial Neural Network.

ATSC Advanced Television Systems Commitee.

BPM Beats Per Minute.

CFFT Critical Flicker Fusion Threshold.

CLI Command-Line Interface.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

DJ Disk Jockey.

ELBO Evidence Lower Bound.

FFA Fusiform Face Area.

FFT Fast Fourier Transform.

FID Fréchet Inception Distance.

FLOPS number of floating points operations.

FPS Frames per Second.

GAN Generative Adversarial Network.

GPU Graphical Processing Unit.

GUI Graphical User Interface.

KID kernel inception distance.

lerp linear interpolation.

MLP Multi-Layer Perceptron.

OFA Occipital Face Area.

OSC OpenSound Control.

PCA Principal Component Analysis.

PPL Perceptual Path Length.

RGB Red, Green, Blue - coding scheme for colored pixels in digital images. Each having values between 0 and 255.

RMS Root Mean Square.

RMSV-TMV Root Mean Square Variance to Temporal Motion Variability ratio.

SeFa Semantic Factorization.

slerp spherical linear interpolation.

VAE Variational Autoencoder.

VJ Visual DJ.

1 Introduction

Artificial intelligence is the talk of town. Although it has been around for more than half a century, current evolutions in data storage and collection methods and the ongoing rise in compute power give rise to ever-more complex and accurate machine learning algorithms.
According to IBM, an early pioneer in the field, artificial intelligence could be defined as the field which combines computer science and robust datasets, to enable problem-solving [1].

The initial artificial intelligence paradigm was rule-based, mimicking the decision-making process of human experts, and required thorough analysis of human rational process to define rules according to which a computer could be programmed. This clearly was a labour-intensive and costly task.
However, the current status of the AI subfields machine learning and deep learning removed the need for expert-defined rules by designing algorithms that can learn the right algorithmic rules from training data. This not only led to a reduced AI cost of implementation, but also increased the accuracy of the models themselves.

AI is not only intriguing in a purely scientific or academic perspective, the AI market is incredibly profitable. According to estimates by Next Move Strategy Consulting, the market of 100 billion U.S. dollars in 2021 is expected to grow twentyfold by 2030 up to two trillion U.S. dollars [2]. This fact pushes innovation from within industry itself.

Whereas initial machine and deep learning applications such as recommender systems and medical classification systems have found their way to the public for a while now already, a new industry breakthrough appeared fairly recently: generative artificial intelligence. Industrial products like DALL-E [3], Midjourney [4] and ChatGPT-3 [5] made generative modelling known and accessible to the larger public during the last few years.

The field of generative modelling is closely tied to the visual arts themselves.
In this study, the potential of generative models as real-time audio visualisers is thoroughly analysed in order to add tools to the arsenal of visual artists and VJs.
This field, uniting classical audio visualisation and generative modelling, is still fairly young, with very little existing research to take inspiration from.

The first quantitive analysis in existing literature of the state-of-the-art was performed in this study by making use of metrics reported in literature and by designing tailor-made ones for the application at hand, namely the *RMSV-TVM* distance metric reported in Section 6.2.4.
Next to existing audio reactive latent space interpolation algorithms, a novel algorithm is proposed based on trajectories on the surface of an $(n + 1)$-dimensional hypersphere. The technique effectively solves some known problems with the existing algorithms and improves overall performance.

In current literature the only synthesis model employed in the context of audio reactive generative modelling, is the StyleGAN2 model [6].
In this study, the potential of β-VAE [7] architecture for real time audio reactive inference has been extensively explored. The results are promising, with β-VAE models outperforming StyleGAN2 based architectures on multiple evaluation criteria, like frame rate and disentanglement.
To top things off, novel techniques for semantic control over the generated visualisations were introduced. This way the toolkit of the visual artist is further expanded and the intuitive the system even further.

1.1 Objective

The goal of this research project is two-fold.
First, the current state-of-the-art of real-time audio visualisation is thoroughly analysed and quantified, by making use of metrics existing in parallel research fields and by designing a new metric specifically designed for the purpose of real-time audio reactive generative modelling.
Performance in terms of computation time, visual quality, coherence and semantic value is reported.
At the same time, multiple alternative design choices are discussed and empirically tested in order to improve the state-of-the-art and expand the artistic toolkit of the artist.
All of this is done with the idea in mind that the technology should be used in live environments by visual artists, so the algorithms and techniques should be robust and user-friendly.
To practical usability of the system has been assessed by using it in multiple live performances throughout Spring 2023. A brief account of these events and setups can be found in Appendix D.2.

1.2 Thesis Outline

The first major part of this research project revolves around giving a background to the reader, in order to understand the goals of real-time audio visualisation, techniques used and research tradition it is rooted in.
To introduce the interested reader to the field of audio visualisation, a brief historical overview of the scientific and artistic field is provided at the start of this report, in Chapter 2.
Subsequently, the job of the modern VJ is described in order to form an idea on what properties a modern VJing tool should possess and what the design constraints are.
To root this research in the field of VJing and audio visualisation itself, a short list is created of existing software packages and their functionality in Chapter 3.
In Chapter 4, the most important concepts from the fields of machine learning and deep generative modelling are explained and sources for further research are listed in order to understand the techniques used in this research project in detail.
Finally, the limited existing work on the subject of real time audio visualisation with generative models is explored and explained to the reader in Chapter 5.
With that background knowledge, the reader should be able to follow the quantitive analysis and proposed improvements of each part of the architecture. In Chapter 6, the research methodology is described, and used metrics are explained in detail. Chapter 7 describes the real-time signal processing techniques used and explains the properties of the extracted musical features. In Chapter 8, three audio reactive interpolation algorithms are presented and evaluated on their performance. In order to add meaningful narrative to the audio reactive interpolations and offer fine-grained control to the artist, Chapter 9 gives an overview of implemented methods to find meaningful directions, latent vectors and controls in the StyleGAN2 latent space. At last, Chapter 10 evaluates the performance of a system where β-VAE models are used as synthesis models, instead of StyleGAN2 models, by quantitively comparing both.
The conclusion, future work and impact are discussed in Chapter 11.

2 Music Visualisation Background

2.1 Historical Overview

For centuries, music has taken up a profound role in human culture. It functions at the same time as a form of expression, leisure, communication, and spirituality. Varying cultures and traditions throughout the ages have included music in spiritual and political expression. Chants as a spiritual elevation of prayers can be found throughout all prominent world religions, from Tibetan Buddhists to the Lutheran church. At the same time, national anthems and military hymns drive feelings of national pride and wariness. It is clear that music has a profound intellectual and emotional impact on mankind as a whole.

Mankind has tried to both understand and control music further and further throughout the ages. As with every other science and craft, understanding pushes control and vice versa. This search for understanding spans the fields of music theory, neuroscience, signal processing and the art sciences and lead to the development of a wide range of musical instruments, recording and playback devices, and visualisation tools.

That last one is what is to our interest in this dissertation, and has taken up a wide range of forms throughout the previous years. Visualisation can help with understanding music better, while at the same time adding a level of expression to a piece of music as well. In what follows a short history of visualisation tools and methods is given, where each form resides on the spectrum between being mainly a form of understanding music, like musical notation, and mainly being a form of expression, like dance performances[1].

[1] Although dance is a form of art in itself, it is very often accompanied by music and could be regarded as a form of visual expression on top of a musical input.

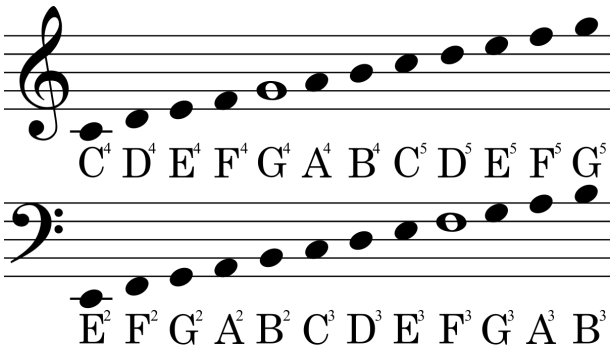


Figure 2.1: The Western standard for music notation: staff notation.

2.1.1 Musical Notation

Maybe the most obvious and albeit the oldest form of music visualisation is musical notation. From a, maybe cumbersome, technical point of view, one could consider the standard musical notation musicians use in today's scores as offline music visualisations. It is clear that the goal of these visualisations is to communicate what an artist should play, making it a type of language of communication rather than an expressive art form based on music. The musical notation should stay as closely as possible to the actual music being played.
In a sense, there is a symbiosis between the visual notation and the music itself: the music drives the notation and vice versa.
The earliest form of musical notation can be dated back to 1400 BCE in the area of Nippur in Babylonia [8], where a tablet was found with an early form of musical notation. From there, musical notation evolved through many forms strongly dependent on musical style and culture. The modern standardized European staff notation framework consisting of the 5 horizontal lines to indicate pitches has been around in its current form since the 20th century [9].
It is used in different musical styles and across different cultures and locations, although of course, different systems are still in use.

2.1.2 Early physical visualisations

Science drives innovation, a fact that is also illustrated in the history of musical visualisation. With the works of the Pythagoreans and their theory on sound frequency ratios and musical consonance, the first mathematical framework for musical theory was introduced around 400 B.C. The Pythagoreans are accounted to have introduced theories around consonant intervals and their frequency ratios and would have introduced the concept of an octave. These observations lead to a physical understanding of sound waves, their frequencies, and their relationship to music [10].

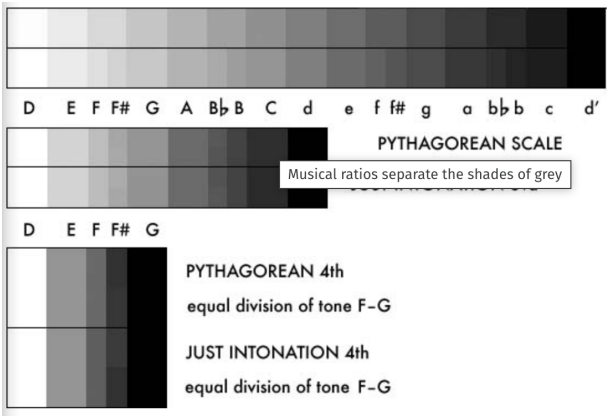


Figure 2.2: Arcimboldo's grey scales, idealized. Adapted from Hutchison [11].

Giuseppe Arcimboldi started studying these Pythagorean proportions of (semi)tones and proposed a relation between the musical scale and colour brightness [11]. Originally a court painter to the Holy Roman Emperors, Arcimboldi also intended to match painting with music. Not only improving the theory of the Pythagoreans, he found a way to create a visual representation of the Pythagorean scale in the form of a grey scale. He matched brightness to pitch, so that low pitches correspond to bright white colours and high pitches correspond to dark colours up to full black. A rise in darkness would occur for each semitone. An idealized understanding of his scale is visualized in Figure 2.2.

In 1704, when Isaac Newton was studying the properties of light, he went a step further en proposed a link between the frequencies of light and sound frequency. He suggested a relationship between the seven colours of the rainbow and the seven notes of the musical scale. He proposed that an increase in light frequency in the colour spectrum from red to violet made a corresponding increase in the frequency of sound in the diatonic major scale [12].

Since then, many others proposed specific mapping schemes between sound frequency and light wavelengths. Controversial physicist Louis-Bertrand Castel introduced a relationship between colour and notes [13]. He motivated the analogy between sound and light by the observation that both are vibrational phenomena, and since notes are special forms of sound and colours special modifications of light, proposed to tie the two together.
In 1743, Castel made this more concrete by inventing the first physical device to transform sound into colour: *the ocular harpsichord*. The machine, although never a completely finished product by the time of Castel's death in 1757, was designed to be a mechanical instrument with keys producing notes. At the same time of a note however, a lantern would light up with a specific wavelength of light. The specific wavelengths of light would be achieved by lighting up lanterns behind coloured glass tiles.
After Castel's death, many other inventors tried to improve upon his ideas and developed their own variations of the harpsichord. A variation of the harpsichord developed by Johann Gottlob

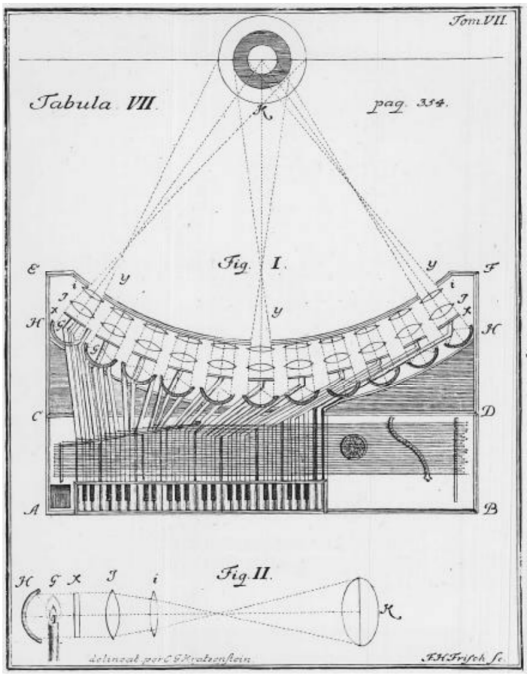


Figure 2.3: Johann Gottlob Krüger's design for an ocular harpsichord. [13].

Krüger in 1743 is depicted in Figure 2.3.

Leaving brightness and colour behind, physician Ernest Chladni, regarded as the father of modern acoustics, chose to study the link between sound and form, giving birth to the field of cymatics. By connecting a sound source to a *Chladni metal plate* on which sand grains are evenly dispersed, patterns emerge when sound is created. Places where sand grains accumulate through the vibrations are called nodes, places from which sand moves away under sound or music vibrations are called antinodes.
The specific locations of nodes and antinodes is characteristic to different frequencies or combinations of frequencies. This way a spatial representation of music is found, where moreover, the exact mapping between the visual and auditive dimension is not decided on by man, but is innate to nature. The same sound would reproduce the same image every time.

Hans Jenny published the first standard work in the field with 'Cymatics, A Study of Wave Phenomena and Vibration' [14]. He takes the experiment further than the Chladni plate alone and states cymatic observations in different fluid, solid and even biological media. He moreover invented the *tonoscope*: a pipe-shaped instrument which causes a diaphragm with quartz sand to move when sang through. He stated that singing at the same pitch results in the same quartz sand shapes, with low tones generating simple patterns and high ones resulting in complex ones [12].

2.1.3 Lighting art and the 20th century.

During the 20th century, the idea of accompanying music with lighting shows began to inspire more artists than it did before. One of these artists was Thomas Wilfred.
He developed a first type of 'light synthesizer' in functionality similar to current-day lighting desks: *the Clavilux* [12]. The machine had six projectors, controlled by a bank of sliders. The projectors would be directed upon prisms painted in different colours and shapes by Wilfred. By moving the sliders, different colours and shapes would shift in front of each other, creating dynamic projections similar to the Northern light [2].

Influenced by Thomas Wilfred's colour organ other artists started combining live visual shows with music and developing mechanical devices to make those possible in the 20th century. Examples are Mary Ellen Bute, Walter Ruttman and Oskar Fischinger.

Jordan Belson paved the way for modern VJing in 1957 when he started to combine visual choreographs with electronic music. By creating a costum built optical bench with rotary tables, variable speed motors and lights of variable intensity, he accompanied early electronic live sets.

An important evolution in the midst of the 20th century was the development of psychedelic light shows to accompany live music shows. Historian Charles Perry states that these were first developed at the San Francisco State College, where art professor Seymour Locks experimented with projection through liquid paints [15] [16]. The technique became a popular feature of live shows during the 1960s and 1970s accompanying psychedelic rock bands, with bands like 'The Grateful Dead', 'Pink FLoyd', 'Jefferson Airplane' and 'The Jimi Hendrix Experience' making abundant use of the technique to further emphasize the psychedelic nature of their music. Famous early liquid light show artists were Glenn McKay, Jerry Abrahams and The Joshua Light Show, by Joshua White, still active today [3]. Some emphasize the connection between the visual aesthetic that these liquid light shows created and the hallucinogenic drug LSD, widely popular during the 60s flower power movement [17].

Media specialist [18] boldly states that these psychedelic light shows originating on the U.S. West Coast were part of the beginnings of a rapidly evolving, strongly visual world language that now seems evident when looking at newer digital media technologies. He states that along with other counterculture activities, light shows evolved as a way to connect people and help raise collective consciousness outside the highly mass-produced and controlled media of television, radio, and cinema.

Around the same time as the liquid light shows, at the U.S. West Coast, a different kind of

[2]Thomas Wilfred - Master of light: `https://www.youtube.com/watch?v=gbs3NQ2mf4c`

[3]An early liquid light show example by The Joshua Light Show: `https://www.youtube.com/watch?v=sVLx1U40BEI&t=96s`

aesthetic was introduced by Andy Warhol and his multimedia project for the Velvet Underground: 'Exploding Plastic Inevitable [4]. Instead of the soft and organic paint visualisation, Warhol chose a more glitchy, dark style with a harder edge and strobe light effects. A style that is still to be found at modern day electronic music events.

Zarate states that, although both visual styles differ, they both reached the immersive effect due to the fact that the visual artist could respond to the energy of the public in real-time.
According to Whitney Museum curator Chrissie Iles, light shows are grounded in the psychological effect of synaesthesia, which she personally describes as the perception of becoming at one with each other, the music and the architectural surroundings of the venue [16].

It is clear that those three elements are still core elements of modern-day concert experiences, and hint that the early light shows of the 60s and the 70s paved a direct way towards modern VJing culture.

2.1.4 The birth of VJing

According to Dekker the term VJ or Visual Jockey originated back in 1981 when the music television station MTV started the non-stop broadcasting of music-video clips, interrupted by presenters under the name of VJs. Although those video clips were meant to boost music sales and give the artist a more public appearance, they had a profound influence on the look and style of live VJing in the years that followed, with VJs saying they were inspired by the visual manipulations and effects in the clips [17].

It was around the mid-90s that the art of the VJ had become as diverse as the different styles of music. With the advent of the digital age and the steadily lowering prices of video mixing hardware, the VJ did not necessarily have to be trained in the art academy no more. A second generation of VJ's was formed from people from different traditions, having roots in computer programming, graphic design, film directing or sound.

2.2 The 21st century and beyond

Fodel states that the history and development of VJing is tied more closely to the history of technology than anything else. The modern VJ has a wide range of tools to its availability ranging from the liquid light shows of the 60s to analogue video synthesizers and tailor-made digital VJing software. It can be observed that all these tools are still widely used in live environments and often combined with one another.

[4]The full 'Exploding Plastic Inevitable audiovisual project: `https://www.youtube.com/watch?v=HsR4ghMfq0U`

2.2.1 The modern VJ: what does he do?

The modern VJ accompanies musical performances with live imagery. The most important requirement of the imagery generated seems to be that it needs to at least seem to an audience that the imagery reacts to or at least is in sync with the music. Apart from that, the VJ can show an audience any content he prefers and has complete artistic freedom.

This goal can be achieved by remixing prerecorded clips with one another on musical cue, or by generating the imagery itself on the spot. For both types of VJing practices, specific tools have been created.

The analogue or digital video synthesizers that have been around in the early club scene are textbook examples of VJing through remixing existing imagery. Early musical visualisations like the harpsichord or the cymatics practices could be seen as musical remixing of existing imagery as well. Although the boundary between generative algorithms is thin.

On the other hand, generative visualisations create imagery on the spot, in the venue itself. This often leads to non-repeatable performances, at least in the sense that one performance will never be the same as a previous one. The early liquid light shows are excellent examples of this. With the advent of the computer, digital generative live performance are widely spread in modern day VJ culture. Through specific software like Max [5] or Touchdesigner [6] or tailor-made algorithms, the artist is able to use music as a data stream input steering a certain visual output.

It is needs to be clear however that, as stated by Zarate, the artist needs to react not only to the music but also the energy of the audience itself. In much the same sense a DJ 'reads' its audience in order to choose between tracks, the VJ needs to wisely react to both the music being played and the audience present in order to keep the performance immersive and elevate the musical performance.

This shows the importance of a human in the loop. Although a generative algorithm could potentially generate complete audio reactive visualisation, the visual artist needs to steer and adjust the algorithm or thoughtfully pick the video clips in his collection based on the audience as well.

The importance of a human in the loop is emphasized by the unpredictability of most musical performances, where a DJ selects tracks and performs transitions between them on-the-fly. The possibility of algorithmic error when dealing with this uncertainty is very real and can be actively dealt with by the artist as a person.

[5] Max website: https://cycling74.com/products/max
[6] Touchdesigner website: `https://derivative.ca/`

Kraasch and Pasquier clearly summarize these requirements in their research paper [20]. A VJing tool should have a physical, tangible interface that is accessible for immediate response and should allow the visual artist to improvise. Moreover, it should allow the artist to have extensive control over visual parameters while keeping usability in mind. Physical controllers like MIDI-controllers can be a tremendous help in adding extra control and can be seen as visual instruments that empower the artist. Last but not least, a system should react to the music and its surroundings and should have the possibility to be unique.

On top of that, VJing systems need to run for whole performances without interruptions. A VJing system should thus be stable: quality should not decrease over time and crashes should be avoided.

2.2.2 Constraints of real-time applications

Frame Rate and the Human Visual System There are two separate elements being important when creating a real-time audiovisual system. The first one is the frame rate of the video itself, without audio. The frame rate needs to be high enough, in order not to perceive individual frames within the video. The *critical flicker fusion threshold* (CFFT) is the threshold above which individual light source flickers can not be distinguished and a smooth continuous video is perceived. The threshold is on average between 60 Hz and 90 Hz, but could in certain situations be as high as 500 Hz [21].

Although the initial standard frame rate for sound film of 24 FPS was below this critical flicker fusion, the images were shown 2 or 3 times in order to exceed the critical flicker fusion threshold [22], arriving at a frame rate of 48 FPS or 72 FPS. To display fast motion however, 24 FPS will fail to completely convince an audience of smooth motion and higher frame rates, preferably above the critical flicker fusion threshold, should be used. Current day movie theatres and broadcasting stations often screen at 48 FPS and 60 FPS, resulting in more realistic motion scenes without motion blur. Often so realistic that it is perceived as unpleasant [23].

Lip Sync Error The second important aspect of real-time audio visualisation is the delay between the audio and the video frame that goes along with the specific audio fragment. This value is also referred to as the 'Audio-to-video synchronization Error' or 'Lip Sync Error', referring to the broadcasting delay between lip movements and speech, since human beings are best at perceiving synchronization errors of that kind. The Lip Sync Error should be below a certain value, in order to be indiscernible for an audience.
The ITU-R BT.1359-1 [24] recommendation for television broadcasting in 1998 states from subjective evaluations that the audio-video delay detectability interval lies between +45 ms to -125ms and the acceptability interval lies between +90ms and -185ms on average, with positive

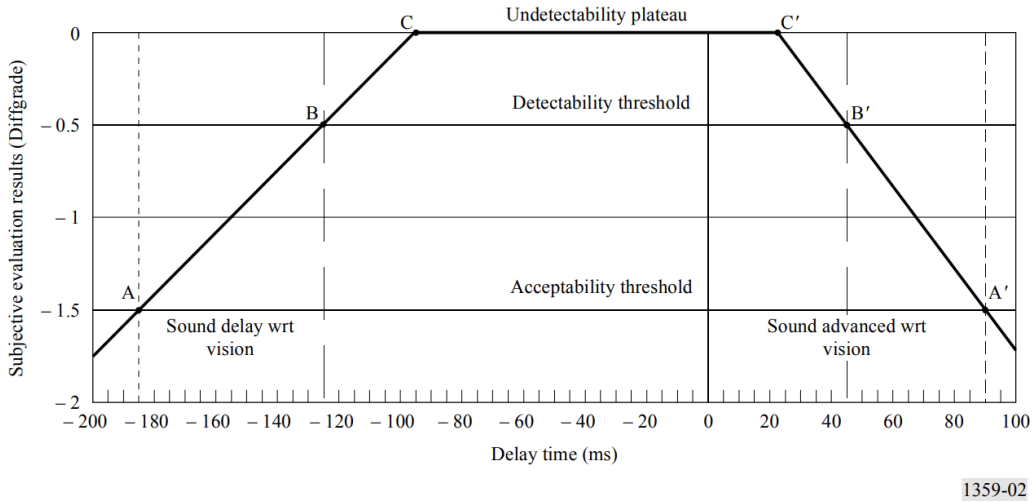


Figure 2.4: ITU-R BT.1359-1 detectability and acceptability thresholds. [24].

values meaning the sound is advanced with respect to vision. When following this reasoning, this means that a video frame that is tied to a certain audio fragment can be delayed by 45 ms before an audience would detect a delay and by 90 ms before the delay would be too big in order to stay acceptable. This recommendation however dates from 1998, an era where multimedia was much less common place than it is now. For that reason, some researchers have opted to re-evaluate the recommendation and generally made it more strict. It would for example make sense that with higher resolution, more details are visible and thus more strict delay requirements need to be met. The Advanced Television Systems Committee (ATSC), an international non-profit developing voluntary standards for digital television, states in their recommendation from 2003 [25] that the ITU-R BT.1359-1 recommendation is found inadequate for audio and video synchronization purposes for DTV broadcasting. They propose tighter bounds and state that sound should never lead video by more than 15 ms and should never lag the video by more than 45 milliseconds.
Kraasch and Pasquier state that modern VJing software at least recommends a delay below 50 milliseconds between audio and image, leading to a frame rate of 20 FPS [20].
An optimal system would generate frames at a rate where individual images are unperceivable to the human eye, leading to a working frame rate of 60 FPS and a delay of 20 milliseconds.
The ATSC standard sets forth the most stringent constraint, while the ITU-R BT.1359-1 recommendation adopts a more loose approach, and Brouwer strikes a balance between the two.

2.3 Existing VJing software and their functionality

To illustrate the state-of-the-art in digital VJing, two software packages and their functionality are discussed in this section, each having specific use cases and specialization areas. To end the section, the concept of Live Coding, a fairly new performing arts paradigm, is introduced. A clear link between the techniques of both live coding and generative visualisation can be found and live coding could end up to be a fruitful framework to embed deep generative audio visualisation in.

2.3.1 Resolume

Resolume [7] is a live video mixing software package, allowing to load prerecorded VJing clips and remix them to musical cue on the spot. The Resolume company was founded in 1998 to create a digital counterpart of the analogue mixing with VHS tapes and an mx50 video mixer, as the team states themselves. It is nowadays one of the standard software packages used in both the context of clubs and festivals. Within Resolume one can map clips to controls on a MIDI-controller and add certain filters, effects, and transitions in real-time. On top of that, Resolume allows for projection mapping and live inputs from cameras or the network. Resolume does not allow for real-time generative visualisation.

2.3.2 Touchdesigner

TouchDesigner [8] is a visual development platform developed by Derivative with a wider application domain than music visualisation alone. It is a visual programming environment offering a graphical user interface which allows artists to manipulate data streams of any kind in order to steer real-time visualisation. It allows to create interactive media systems, projections or live music visuals. In Touchdesigner one also directly code in Python leading to endless possibility when it comes to generative audio visualisation. One can decide to not only let the music steer the visual output but could also use other sources of information like audience movement, facial sentiment or heat maps.

2.3.3 Live Coding

Live Coding is the creative process of programming music or visuals live on stage. Although the idea of live programming generative art has been around since the 90s, the concept was

[7] Resolume website: `https://resolume.com/`
[8] Touchdesigner website: `https://derivative.ca/`

introduced to a larger audience with the creation of the Algorave [9] concept as a global dance event by Alex McLean and Nick Collins. The community is loosely guided by their 2000s 'The Generative Manifesto' [10] stating in 5 statements what the philosophy of a live coder should include. The manifesto asks a visual or musical artist to use code to dive deeper into artistic detail, to keep things real-time, to reflect human narrative, to keep code open and only use software applications written by the community and the artist itself.
It should become clear that the work in this research project lies within line of these guidelines and could thus potentially form a contribution to the field of live coding in general.
The system created in this project was used in a live performance at the Algorave Ghent event in May 2023, described in Appendix D.2.2.

[9] Official Algorave website: `https://algorave.com/`

[10] The Original Generative Manifesto: `https://slab.org/2015/07/28/the-generative-manifesto-august-2000/`

3 A Primer in Machine Learning

3.1 Machine Learning

The goal of this dissertation is to make it possible to employ current state-of-the-art generative modelling to generate visually appealing videos reacting to music in real-time. It thus seems right to start off with the question: what exactly are deep generative models, and how do they work?

In order to understand what generative models are and how they operate, one first needs to understand the objectives of classical machine learning. IBM, one of the pioneering organizations in the field, defines the field of machine learning on their website as follows:

> Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy [27]. *–IBM*

Machine learning thus is a subfield of artificial intelligence, where algorithms are designed to learn a certain objective function from training data. The objective function is a mathematical formulation of the specific goal the machine learning model needs to realize after its training phase. The function is approximated during the training phase as close as possible, minimizing a loss function between the objective function and the realized function at each step. The only way of relating the proximity of the realized function with the objective function is through the realized output data and the input training or validation data. [28]

$$\forall n \quad y_n \approx f(x_n) \tag{3.1}$$

with x_n an input sample, y_n the realized output and $f()$ the objective function.

Approximation is then achieved by minimizing the loss between the realized output data and the real-world data. Different loss functions can be used depending on the specific machine learning problem at hand. Mean-square error loss for linear regression and cross-entropy loss for classification are the most common ones and more important ones for the rest of this discussion.

The goal of the mean squared error is to minimize the average squared difference between the predicted output and the true output.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.2}$$

where n is the number of samples, y_i is the true output, and $\hat{y}_i$ is the predicted output.

When minimizing the cross-entropy, the loss between the predicted probabilities and the true binary labels is minimized. In mathematical formulation, this becomes:

$$\text{Cross Entropy Loss} = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{3.3}$$

where n is the number of samples, $y_i$is the true label (0 or 1), and $\hat{y}_i$ is the predicted probability. Based on the form of the objective function we want to approximate, machine learning can be broadly classified into three categories: supervised learning, unsupervised learning and semi-supervised learning.

3.1.1 Supervised Learning

Supervised learning is occupied with the task of learning the distribution $p(y|\mathbf{x})$ with y being the set of labels and $\mathbf{x}$ the set of data points. Each label is bijectively linked with a data point. The data points are vectors in general [29].
In case of regression, the labels y are real-valued, in case of classification the labels are discrete and categorical. In case of supervised learning, we thus need labelled data, which involves some human intervention in the learning process. We call $p(y|\mathbf{x})$ the posterior distribution.
Supervised learning is the most common setting for training discriminative models. These type of models try to learn a decision boundary that separates different groups of data points. More rigorously, they try to model the posterior probability $p(y|\mathbf{x})$ directly and then use decision theory on these posterior probabilities to perform classification or regression tasks.

3.1.2 Unsupervised Learning

As the name suggests, we do not need the human supervision of labelling datasets when using unsupervised learning. In the absence of data labels y, there is no posterior distribution $p(y|\mathbf{x})$ to be modelled. Instead, we could try to quantify the full data distribution $p(\mathbf{x})$.

This is where generative modelling comes in. If we achieve to model full data distributions, we can start sampling or generating new data from those distributions.

Apart from generation, however, generative models can still be used for discriminative tasks as well. Imagine the situation where data would be perfectly labelled, one could try to model the joint distribution of data points and labels $p(\mathbf{x}, y)$. By making use of Bayes' rule listed in Appendix A.1.2, the posterior distribution $p(y|\mathbf{x})$ can be rewritten as follows [29].

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{3.4}$$

In what follows we will reference $p(y|\mathbf{x})$ as the posterior, $p(\mathbf{x}|y)$ as the likelihood, $p(y)$ as the prior and $p(\mathbf{x})$ as the evidence.

The evidence $p(\mathbf{x})$ can be rewritten as well.

$$p(\mathbf{x}) = \sum_{y} p(\mathbf{x}|y)p(y) \tag{3.5}$$

This formulation hints towards an important concept that will be dived in deeper on in the rest of this discussion: latent variables. Even when there is no labelled data available, one can suppose that there are underlying structures within our data. These categories can take various forms like colour, shape, size or even non-evident features that can only be extracted with extensive neural networks. We will call these hidden variables the latent features of our data. In the evidence formulation these latent variables can be viewed to be represented by the distribution $p(y)$.

In a sense, these latent variables can be seen as a condition on the generation process, where a certain variable will add context information. A conditional model will learn the probability $p(\mathbf{x}|\mathbf{c})$ with $\mathbf{c}$ representing the context at hand. Again, this context can be any hidden or wanted feature. Context adds a sense of control over the generation process. Because of this latent variable and the concept of context conditions, generative modelling can often take a self-supervised form instead of a solely unsupervised one.

3.1.3 Semi-supervised Learning

Semi-supervised learning can be seen as a midway between supervised and unsupervised learning, where an input-output relationship is learned with only a few labelled data points. By first training on a small labelled dataset, then using classification or regression on the unlabelled data with the learned model and finally adding the data points with their generated labels to the training set if their confidence scores are high enough, the time- and labour-consuming task of labelling is minimized.

3.2 Deep Learning Primer

In classical machine learning, feature extraction is a job for a human expert. It is a form of data preparation where an expert in the field at hand goes through the data and tries to manipulate or aggregate it in order to present relevant features to the model to train on. This approach has its limits. Foremost, it is a costly operation to have human experts spend lots of time on engineering the features of a dataset, second of all the features in a dataset are not always clear to the experts themselves. If you for example take the question of generating human faces, what are the exact features of an image that make sure that the generated distribution is the distribution of a human face? Some problems are simply too complex for a human being to engineer.

3.2.1 Artificial Neural Networks

Deep Learning presents a solution for the problem of feature engineering, by not only learning relevant classification or generation rules, but the features as well. Its architecture is grossly modelled after the biological functioning of the human brain: the artificial neuron formally resembles the biological neuron.

For this reason, these networks are called Artificial Neural Networks (ANN). The architecture of an ANN is visualised in Figure 3.1. An ANN consists of an input layer of d neurons, L hidden layers with K neurons, and an output layer of d neurons with d the dimension of the wanted output data. The value of K and L are freely chooseable and tuneable to achieve the highest quality results. Each neuron in the network essentially realizes a linear regression function, where the outputs of neurons in the previous layer are multiplied by a trainable weight value and relayed to a neuron of the current layer. Each node is also fed through an activation function, making non-linear functions possible. The function realized in each neuron is given in Equation 3.6.

$$x_j^{(l)} = \phi(\sum_i (w_{i,j}^{(l)} x^{(l-1)} + b_j^{(l)}) \tag{3.6}$$

with ϕ representing the activation function, $w_{i,j}^{(l)}$ the weight value of node i to j in layer l and $b_j^{(l)}$ the bias value of node j in layer l. Common activation functions are visualised in Figure 3.2.

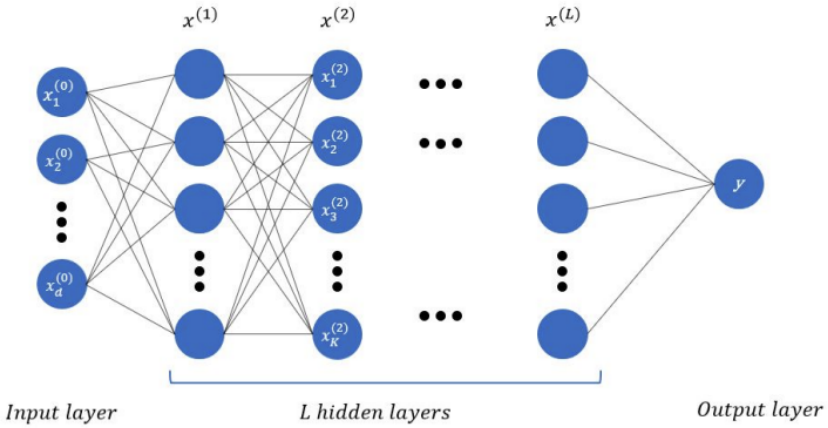
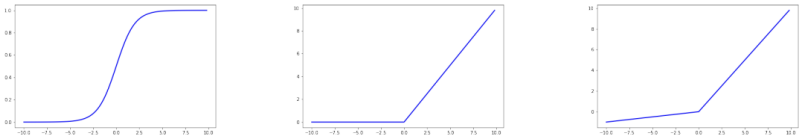


Figure 3.1: The Artificial Neural Network architecture [30].



(a) Sigmoid (b) ReLU (c) leaky ReLU

Figure 3.2: Common activation functions [30].

An ANN where every neuron in a layer is connected to all neurons in the previous layer is called a Multi-Layer Perceptron or MLP. Since no connections are made from later layers to earlier ones, the ANN architecture is also called a feed-forward neural network.

Training the weight and bias parameters of an ANN happens through a process similar to gradient descent with a loss function similar to the ones sketched in Equations 3.2 and 3.3. Since now multiple neurons are connected to each other, the loss needs to back propagate through the whole ANN. Although important in the field of Deep Learning as a whole, we will not go further into the training details of an ANN here, since the training process will not matter for the tuning of the Deep Generative Modelling of real-time visuals.

It has been proved, that MLPs act as universal function approximators for continuous functions [31].

3.2.2 Convolutional Neural Networks

Although Neural Networks have proved to be immensely successful in many regression and classification of one-dimensional data, they show to be insufficiently efficient on image and video data where data points (often RGB-values) are highly correlated with each other in the spatial (images) and temporal (video) domain.

Convolutional Neural Networks (CNNs) are specifically developed neural networks for images and video as input data. Since the goal of this research is to generate artificial images and video, all tailor-made generative models that will be tackled in the next section incorporate the use of CNNs for feature extraction.

CNNs can handle 2D input data with multiple channels. In the case of images, the 2D pixel grid of 3 colour channels serves as an input to the network. The architecture of a CNN is visualised in Figure 3.3.

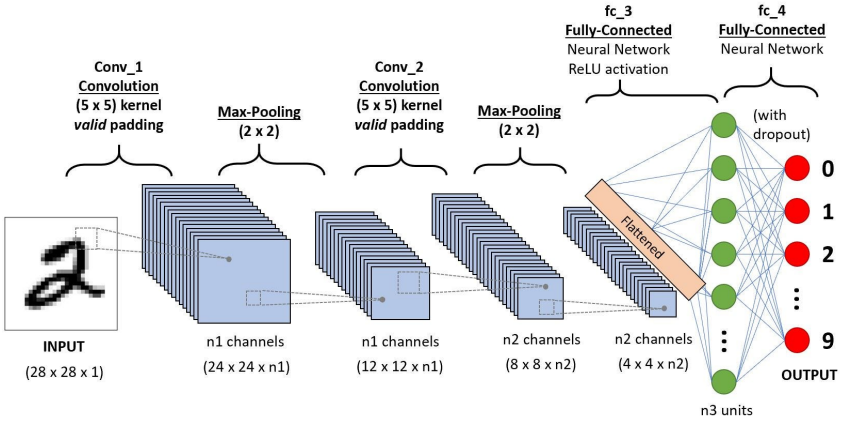


Figure 3.3: Convolutional Neural Network architecture [32].

The way a CNN retains the spatial correlations between pixels is through the mathematical operation of convolutions. During a convolution, a kernel of certain size (NxN) is slid over the input image with a certain stride. The stride represents the amount of pixels a kernel centre is moved between each convolution. During a convolution each pixel is multiplied with the kernel value and the resulting values are summed together leading to a feature map of convolved features in each layer. This can be seen in Figure 3.4.

$$x^{(1)}_{n,m} = \sum_{k,l} (f_{k,l} \odot x^{(0)}_{n-k,m-l}) \tag{3.7}$$

The convolution operation can be expressed mathematically as in Equation 3.7. Where $\odot$

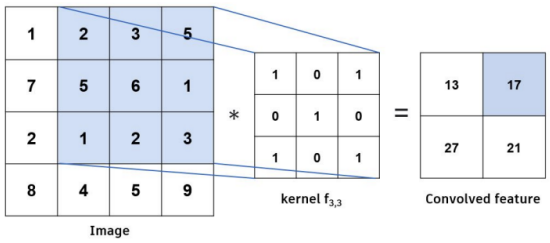
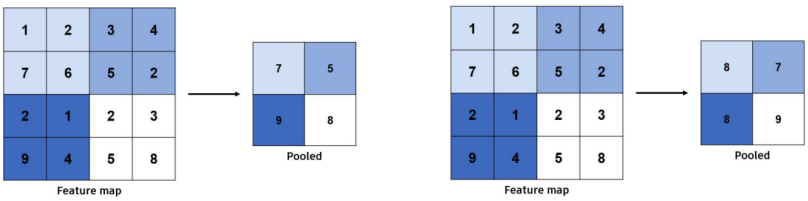


Figure 3.4: A convolution operation of size with kernel size (3×3) and stride 1 [30]

represents the Hadamard product and the superscripts (0) and (1) stand for the first and second layers respectively.

By stacking convolutional layers, the network can learn more and more complex features. Whereas the first convolutional layer will just get separate pixels as an input, following layers will get feature maps as inputs that can be combined through convolutions on their turn again.

Most often, convolutional layers are interleaved with pooling layers. These layers realize a downsampling operation, where the resulting feature map after pooling is smaller than the feature map before pooling. Several pooling techniques are possible, the two most common ones max and average pooling are sketched in Figure 3.5.



(a) 2x2 max pooling

(b) 2x2 average pooling

Figure 3.5: Two kinds of pooling operation: max and average pooling. Filter sizes of 2×2 [30].

4 Deep Generative Models

In the remainder of this research, only generative modelling will be discussed further. The goals of generative modelling have been sketched above. The question remains how the goal of modelling the joint distribution $p(\mathbf{x}, y)$ is achieved. The objective can be formulated similarly as we did earlier in Equation 3.1: one needs to minimize the distance between the real data distribution and the generated data distribution.

$$\min_{\theta \in \mathcal{M}} d(p_{\text{data}}, p_\theta) \tag{4.1}$$

with p_{data} the real data distribution and p_{theta} the modelled distribution. $\mathcal{M}$ represents the collection of all possible data distributions. Important questions now become how probability distributions can be compared to minimize their respective distance, how $\mathcal{M}$ should be specified and how minimization should be solved.

Other questions are how to model the joint distribution of many random variables that are not independent of each other. Latent features of the data influence each other. For example, most of the cars have four wheels, while most bicycles only have two. The hidden feature 'car' in a dataset is thus tied to the hidden feature of 'wheel count' in a dataset of vehicle images. A goal of generative modelling is to represent all relevant features of the data as compactly as possible, while still delivering qualitative results.

It is clear that answering these questions requires complex algorithmic thinking with many variables, especially when generating multidimensional data with lots of hidden variables like image and video. This brings us to the *deep* part of deep generative models, by making use of deep neural networks current generative models can achieve state-of-the-art performance.

4.1 Autoregressive models

As stated in Section 3.2 neural networks act as universal function approximators. We should thus be able to create a (C)NN that approximates the conditional distributions generated through using the chain rule, stated in Appendix A.1.1, on the distribution one wants to model.

$$p(X_1, X_2, X_3, X_4) = f(X_1)g(X_2|X_1)h(X_3|X_1, X_2)k(X_4|X_1, X_2, X_3) \tag{4.2}$$

where each of the functions f, g, h and k can be approximated through the use of a (C)NN. Equation 4.2 shows an innate structured form, where x_t at each time step t is predictable base on all previous data points $[x_1, x_2, ..., x_{t-1}]$ through Equation 4.3.

$$x_t = c + \sum_{i=1}^{p}(\delta_i x_{t-i} + \sigma_t) \tag{4.3}$$

These autoregressive models can generate sequential data, where one data points depends on previous data points.

However, images do not have an innate sequential dependency between pixels. Pixels do not only correlate with previous pixels in a certain direction but with certain pixels in all directions. For that reason, a different approach is necessary.

Fully Visible Sigmoid Belief Networks (FVSBNs) and Neural Autoregressive Density Estimation (NADE) are autoregressive architectures that are able to generate images.

4.2 Variational Autoencoders

The autoregressive paradigm sketched in Section 4.1 assumes that each pixel depends on the previous one. It is clear this is not the best approach for generating images, where correlation between pixels is not sequential in nature. A more general formulation is necessary for the generation of images.

$$p(\mathbf{x}) = \int_{\mathbf{Z}} p(\mathbf{x}|\mathbf{Z}) \cdot p(\mathbf{Z})\, d\mathbf{Z} \tag{4.4}$$

where $\mathbf{Z}$ represents the latent or hidden variables within the dataset. These latent variables can represent any relevant any hidden information in the data like age, gender, or ethnicity that

could prove useful for modelling the distribution $p(\mathbf{x})$. When choosing appropriately, $p(\mathbf{x}|\mathbf{Z})$ could be much easier to model than modelling $p(\mathbf{x})$ directly. It is however clear that finding the set of latent variables $\mathbf{Z}$ is a complex task.

The aim of a variational autoencoder [33] is to generate images from the distribution $p(\mathbf{x}|\mathbf{Z})$ where $\mathbf{Z}$ represents the mean and variance of the image distribution to be modelled.

It does this by combining the concept of autoencoders and variational inference. It has a 'diabolo'-shaped[1] architecture consisting of an encoder mapping the input image to a lower dimensional latent representation of the mean and variance parameters of a multivariate Gaussian and a decoder generating an image by sampling off of this latent probabilistic representation. The architecture is visualised in Figure 4.1.

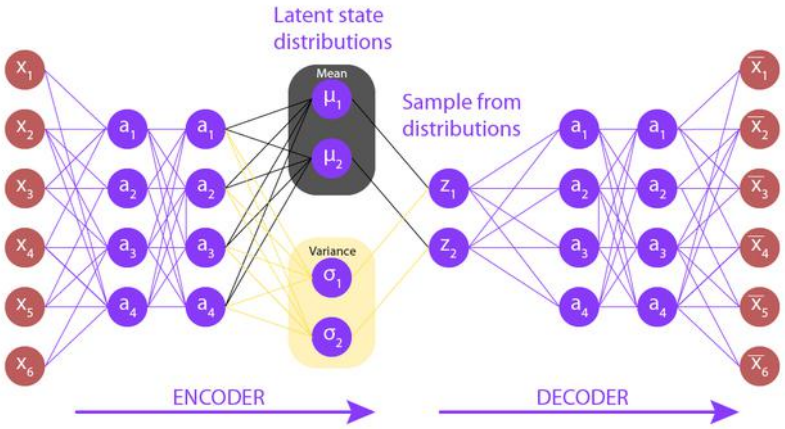


Figure 4.1: The variational autoencoder architecture.

Put more rigorously, the encoder approximates the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$, while the decoder learns the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ with θ the set of model parameters. At first glance this seems like a problem easy to solve by making use of neural networks, however $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable.

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{z}, \mathbf{x})}{\int_{\mathbf{z}} p_\theta(\mathbf{z}, \mathbf{x})} \tag{4.5}$$

Here is where variational inference comes in. Since $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, a family of distributions $q_\phi z$ is picked with their own variational parameters ϕ. If we can make q as close as possible to p, we can just infer from q instead of p. Since we can choose q, we can make it tractable.

[1] A diabolo toy, often simply called a "diabolo", is a type of juggling or skill toy that consists of an hourglass-shaped object with two cups or dishes at each end, connected by an axle or spindle.

4.2.1 KL Divergence

To measure the proximity of q and p and use the value as a loss function, the KL-divergence in Equation 4.6 is used.

$$D_{KL}[q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})] = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\log\left(\frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right)\right] \tag{4.6}$$

We cannot directly minimize this value since we do not know $p(\mathbf{z}|\mathbf{x})$ but by rewriting D_{KL} we discover it is equivalent to the sum of the negative Evidence Lower Bound (ELBO) and the log evidence.

$$\begin{aligned} D_{KL}[q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})] &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\log\left(\frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right)\right] \\ &= \mathbb{E}_q[\log q(\mathbf{z})] - \mathbb{E}_q[p(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_q[\log q(\mathbf{z})] - \mathbb{E}_q[p(\mathbf{z},\mathbf{x})] + \log p(\mathbf{x}) \\ &= -(\underbrace{\mathbb{E}_q[p(\mathbf{z},\mathbf{x})] - \mathbb{E}_q[\log q(\mathbf{z})]}_{\text{ELBO}}) + \underbrace{\log p(\mathbf{x})}_{\text{log evidence}} \end{aligned} \tag{4.7}$$

As the name suggests, the ELBO is the lower bound on the evidence, the derivation is given in Equation 4.8.

$$\begin{aligned} \log p(\mathbf{x}) &= \log \int_{\mathbf{z}} p(\mathbf{x},\mathbf{z}) \\ &= \log \int_{\mathbf{z}} p(\mathbf{x},\mathbf{z})\frac{q(\mathbf{z})}{q(\mathbf{z})} \\ &= \log\left(\mathbb{E}_q\left[\frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]\right) \\ &\geq \mathbb{E}_q\left[\log\frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right] \\ &= \mathbb{E}_q[p(\mathbf{z},\mathbf{x})] - \mathbb{E}_q[\log q(\mathbf{z})] \end{aligned} \tag{4.8}$$

Through Equations 4.8 and 4.7 it becomes clear that minimizing the KL divergence is equivalent to maximizing the ELBO, of which we can compute all constituents.

4.2.2 Amortized Inference

In VAE's we choose $q_\phi(\mathbf{z}|x^{(i)})$ as Gaussians with different means $\mu^{(1)},...,\mu^{(N)}$ and use amortized inference so that we just learn a single function $f(\mathbf{x}^{(i)}) = \mu^{(i)}$. So that we just learn $q_\phi(\mathbf{z}|\mathbf{x})$.

4.2.3 Training

Both the generative parameters θ and the inference parameters ϕ need to be trained simultaneously through gradient ascent on the ELBO loss.

$$\nabla_{\theta,\phi}\mathcal{L}_{\text{ELBO}} = \nabla_{\theta,\phi}\left[\mathbb{E}_{q_{(\phi)}}[\log p_\theta(\mathbf{z},\mathbf{x})] - \log q_\phi(\mathbf{z})\right] \tag{4.9}$$

The gradient with respect to θ can be calculated by making use of Monte Carlo Estimation, as is done in Equation A.1.5.

$$\nabla_\theta[\mathbb{E}_{q_{(\phi)}}[\log p_\theta(\mathbf{z},\mathbf{x})]] \approx \frac{1}{K}\sum_{k=1}^{K}\nabla_\theta \log p_\theta(\mathbf{z}^{(k)},\mathbf{x}) \tag{4.10}$$

4.2.4 The reparametrization trick

Calculating the gradient with respect to ϕ is harder, since the expectation itself depends on ϕ. It is not possible to back propagate gradients of the stochastic nodes through the network. The training process requires sampling from the approximate posterior distribution, introducing non-determinism and thus no efficient way for backpropagation.
The reparametrization trick moves the stochastic nature of the latent variable outside the network by rewriting the latent variable z $\sim q_\phi(\mathbf{z}|\mathbf{x})$.

$$\mathbf{z}^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)} \text{ with } \epsilon^{(l)} \sim \mathcal{N}(0,\mathbf{I}) \tag{4.11}$$

with ϵ thus being stochastic noise sampled from a standard Gaussian distribution. By taking the stochastic nature away from μ and σ and placing it outside the network, back propagation becomes possible.
An efficient encoder module can learn the approximate posterior through a CNN architecture. The decoder architecture in essence tries to implement the inverse function of the encoder and could be implemented through an MLP part followed by some deconvolutional layers.

4.2.5 Latent Space Entanglement

Latent space entanglement is the situation where the learned latent representation of a data distribution does not exhibit a clear separation of the underlying factors of variation. In other words, multiple features are not being independently encoded nor encoded in different dimensions

or subspaces of the latent representation. In a more rigorous formulation, the entanglement problem can be stated as a deficit of the mapping function $q(z|x)$ between the real-world data distribution X and the latent space distribution Z, realized by the encoder. This deficit is inherent.
Disentangling the latent space of generative models results in a higher interpretability of the generated results and thus the ability of developing fine-grained control over the generated data. By varying and combining different factors of variation through latent dimensions, the generated data can be steered into a wanted direction.
In an entangled latent space, this is much harder, since varying one feature could vary one or more features at the same time.

4.3 β-VAE

Although the VAE latent space is already fairly disentangled in nature, the concept of β-VAE's [7] improves disentanglement further.
By introducing a hyperparameter β during training, disentanglement can be further encouraged.

To understand how the β parameter influences disentanglement, the ELBO loss should be rewritten once again.

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log p_\theta(\mathbf{z},\mathbf{x}) + \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log p_\theta(\mathbf{z},\mathbf{x}) + \log p(\mathbf{z}) - \log p(\mathbf{z}) + \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{reconstruction}} + \underbrace{D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]}_{\text{regularization}}\end{aligned} \tag{4.12}$$

Formulated in this fashion, one can view the first term as the reconstruction loss and the second one as a regularization term between the posterior and the prior distribution. By adding a regularization parameter β to the expression, one can put a constraint on the latent information bottleneck.

$$\mathcal{L}_{\text{BETA}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log p_\theta(\mathbf{x}|\mathbf{z})] + \beta \times D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \tag{4.13}$$

By making the β parameter larger, one can enforce the model to learn sparse latent vectors, where semantic variation is centred in one dimension at a time. This aspect of β-VAE's will prove fruitful in the generation of real-time audio reactive interpolations later on.

4.4 Generative Adversarial Networks

A few years later than the VAE paper by Kingma and Welling, the paper by Goodfellow et al. was a deal-breaker [34]. By using a remarkably intuitive approach based on game-playing, high-quality images could now be generated by a generative model.

In their approach, Goodfellow et al. trained two networks with opposite goals: a *Generator* and a *Discriminator*. As its name suggests, the discriminator is a classifier, trained to get good at discriminating between actual real-world images and fake images created by the *Generator*. This *Generator* on the other hand is being trained to generate images growing in realism with every training epoch: it tries to get better at fooling the *Discriminator* by improving the similarity between the real image dataset and the generated one. Figure 4.2 graphically shows the architecture.

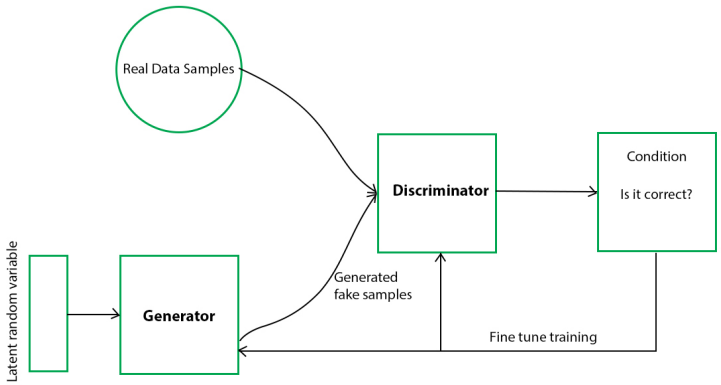


Figure 4.2: The model architecture of a generative adversarial network [34].

This process can be seen as a minimax game being played between the generator model and the discriminator model, a concept originally formulated in the concept of zero-sum game theory [35]. The optimization loss is formulated in Equation 4.14:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))] \tag{4.14}$$

Although the GAN-architecture leads to impressive results in terms of image quality and performance. It does not offer in-depth control over which images are generated. There is no control over the image-style.

4.5 StyleGAN

The problem of inadequate control is tackled in the work by Karras, Laine, and Aila, where a specific architecture is proposed in order to improve on some initial GAN-functionality while also adding control over the style of generated images [36].

Karras, Laine, and Aila state that the initial Generative Adversarial Networks designed by Goodfellow et al. grossly stayed a black-box, which led to a poor comprehension of the structure of the latent space and thus a lack of control over the semantics of generated images. There is no efficient way to control the properties of generated images, and no hierarchical way to control low-level features like background or brightness and high-level ones like eye colour or freckles in human faces separately. In what follows, these features will be referenced to as the image "style". Apart from control, original GAN does not efficiently tackle the problem of entanglement.

The Architecture

StyleGAN delivers an adaptation of the original GAN architecture with the goal of separating high-level attributes from low-level ones in an unsupervised manner. Apart from that, a separation is introduced between these deterministic facial features and stochastic ones like freckles or hairstyle.

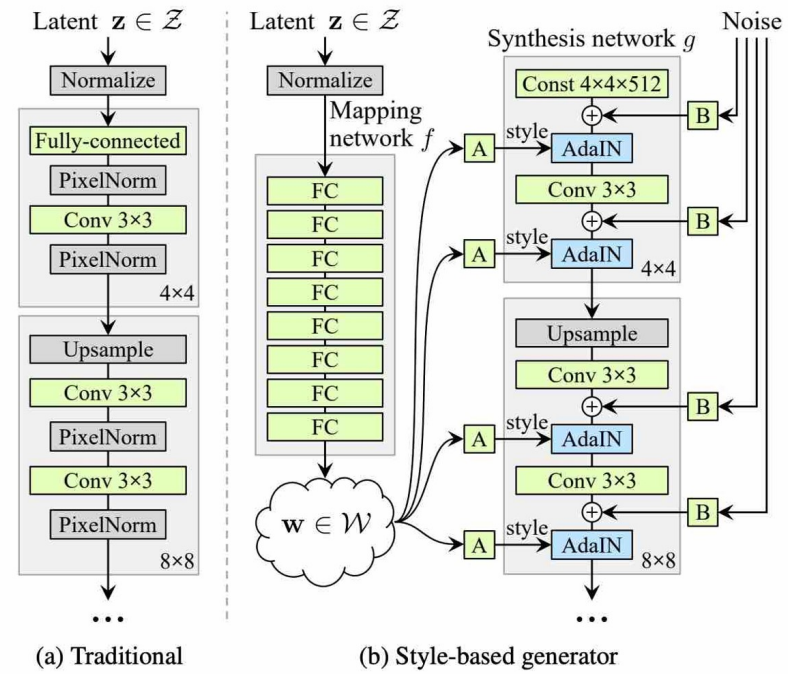


Figure 4.3: Initial StyleGAN architecture [36].

Figure 4.3 shows a comparison between the traditional GAN generator architecture and the

style-based generator proposed by [34]. The discriminator has exactly the same architecture and functionality in StyleGAN as in traditional GAN. Where traditional GAN generated a new image by feeding a random z latent directly through the trained generator network, with an image as a result, the StyleGAN generator appears to be more complex.

StyleGAN is developed as an extension of the progressive growing GAN [37], which proposed a new training method leading to significantly better visual results and reduced training time. Starting from a low resolution in both generator and discriminator, new layers are added that model increasingly fine details as training progresses.

The StyleGAN architecture utilizes the exact same discriminator as the original GAN architecture did, but improved the Generator architecture to such an extent that overall distribution quality metrics scored better, interpolation properties improved, and feature entanglement was reduced.

In order for the image style to be controllable, an auxiliary mapping network, an 8-layer Multi-Layer Perceptron, is introduced which maps a 512-dimension latent space point to a 512-dimension style vector. The initial latent vector is said to reside in the Z-space and is transformed to the W-space.

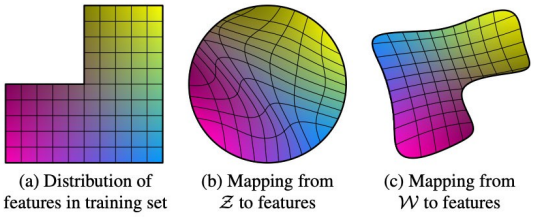


Figure 4.4: $\mathcal{Z}$ to $\mathcal{W}$ transformation.

The style vector is transformed and incorporated into each block of the generator model after the convolutional layers via an operation called adaptive instance normalization or AdaIN.

The AdaIN operation involves first standardizing the output of the feature map to a standard Gaussian, then adding the style vector as a bias term.

$$AdaIN(x_i, y) = y_{s,i} \times \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \tag{4.15}$$

The original Latent Point Input in the GAN architecture is replaced by a $4 \times 4 \times 512$ constant value input.
As a final tweak, Gaussian noise is added to the output of each convolutional layer prior to AdaIN operations. The noise is broadcasted to all feature maps using learned per-feature scaling factors, and then added to the output of the corresponding convolution. This noise addition adds style-level variation at each level of detail.

Both the feature vectors and noise, will prove helpful in making the Audio-Reactive interpolations in the next section.

4.5.1 Disentanglement

By introducing the mapping network from the $\mathcal{Z}$ to the $\mathcal{W}$ space, Karras, Laine, and Aila state that the GAN latent space becomes less entangled. To measure this, two new metrics are proposed: the Perceptual Path Length from Equation 6.2 and the linear separability. The first one is the Perceptual Path Length used to measure the smoothness of linear interpolations in the latent space with the assumption that a disentangled latent space should result in smoother interpolations between all endpoints the exact formulation of the metric is formulated in Equation 6.2. The second one tries to measure how separable points representing different features are in the latent space via a linear hyperplane, so that each corresponds to a different binary attribute of the image. Karras, Laine, and Aila state that the $\mathcal{W}$ space consistently results in better disentanglement than $\mathcal{Z}$ in terms of both PPL and linear separability.

4.6 StyleGAN2

Karras et al. improved the quality of the generated images by their initial StyleGAN architecture by redesigning the generator normalization, revisiting the progressive growing architecture and regularizing the generator to further encourage good conditioning in the mapping from latent codes to images. This not only improves image quality and latent space smoothness, but makes the generator significantly easier to invert, making it possible to project images to latent codes ànd attribute a generated image to a specific generator network. The exact improvements made and architectural tweaks done in StyleGAN2 are less of a concern for this research, but can be found in the paper by Karras et al. [6].

4.7 StyleGAN2-ADA

A final important adaptation to the StyleGAN architecture has been made by Karras et al. to tackle the problem of scarce datasets to train data-intensive models like GAN and StyleGAN. A GAN requires on average 10^5 to 10^6 training images in order to generate high-quality, high-resolution images. For many applications and image themes, it is a very costly operation to these kinds of image quantities.
The problem with using small datasets lies in the fact that the discriminator model starts to overfit to the training samples. This is a classical machine learning problem, which causes a

model to insufficiently generalize.
Karras et al. solve this problem by introducing the concept of augmentation in the training process of StyleGAN.
In training of discriminative models, augmentation is the process of transforming the existing dataset in terms of translations, rotations, noise injections and colouring to create extra data for a model to be trained on. It makes the model more robust and forms the standard solution against overfitting.
However, Karras et al. state that classical augmentation, as used for discriminative models, cannot directly be applied to generative models since the augmentations "leak" to the generated samples. This would mean that for example an augmented dataset by means of colouring or noise injection, would also generate discoloured or noisy data samples.
The solution according to Karras et al. is to slide a filter over the discriminator, the filter distorts the input image to the discriminator in such a way that augmented and non-augmented samples cannot be distinguished from each other. The exact implementation of this pipeline is called Adaptive Discriminator Augmentation (ADA) and is further explained in their 2020 paper [38]. By using ADA Karras et al. were able to both improve the quality of images generated by models trained on small datasets from scratch and those of models fine-tuned with limited data from existing models. Since Karras et al. state that the FID from Equation 6.1 does not serve as an optimal metric when using small datasets, they use the Kernel Inception Distance (KID) [39], a more sensitive unbiased kernel-based quality metric as well to report quality improvements. Just like FID, a lower KID indicates higher quality of generated images. The exact scores and setup details can be read in [38].

5 Related Work

5.1 The Latent Space

A fundamental property of both GAN and VAE latent spaces to realize the goal of this project is the ability to interpolate between different points in the latent space. During such an interpolation, one latent vector is transformed into another via several steps, with the possibility to generate an image from the latent vector at that specific time step. This way, a sequence of slightly differing images can be generated to morph one image into another via the latent space. This way, a smooth video sequence can be generated, if the amount of images generated per second is high enough. Any trajectory between two points can be followed, but common interpolation techniques are linear (lerp) or spherical linear interpolation (slerp).

5.1.1 Latent Space Arithmetic

First observed in the context of natural language processing, Mikolov et al. discovered that the coding space of learned word representations showed highly showed rich linear structure [40]. This property of word embeddings allowed to manipulate words in a semantically meaningful way through simple vector arithmetic operations. A canonical example is given in Equation 5.1 where the arithmetic on the word embeddings of “king”, “man” and “woman” led to the word embedding of the word “queen”.

$$E[\text{'Queen'}] = E[\text{'King'}] - E[\text{'Man'}] + E[\text{'Woman'}] \tag{5.1}$$

Radford, Metz, and Chintala observed similar structure in the latent space of their DCGAN, where simple arithmetic on latent vector representations would lead to semantically meaningful visual manipulations [41]. Using these insights, latent interpolations can be made meaningful by

looking for meaningful directions of change in the latent space and adding these directions to a latent vector one wants to change in a certain way, visualised in Figure 5.1. This leads to the possibility of editing images through latent space arithmetic, as formulated by Shen et al. in Equation 5.2 [42].

$$\text{edit}(G(\mathbf{z})) = G\left(\mathbf{z}'\right) = G(\mathbf{z} + \alpha\mathbf{n}) \tag{5.2}$$

with G representing the generator function, $\mathbf{z}$ the latent vector, $\mathbf{n}$ the direction vector and α some manipulation intensity.

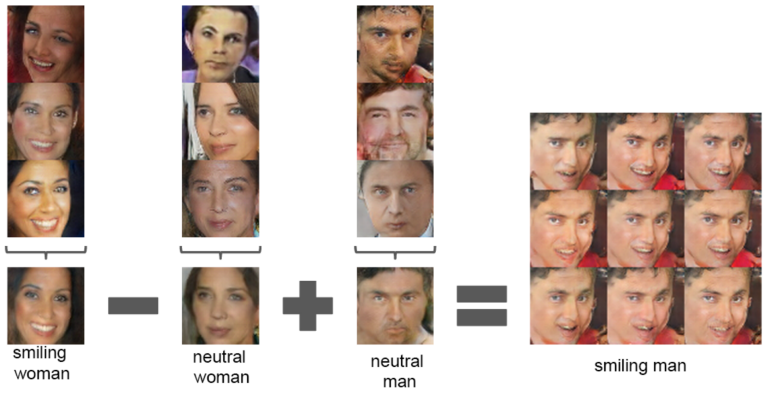


Figure 5.1: Latent space arithmetic as reported by Radford, Metz, and Chintala in the DCGAN paper [41].

5.2 Audio-reactive Latent Interpolations

The first research into making latent interpolations audio reactive was done by Brouwer in his paper from 2020 [26]. Brouwer however, confined his research to offline audio reactive latent interpolation where the music should not be processed in real-time and some human intervention is possible to match the visual and musical features. In other words, the research of Brouwer was meant for the generation of music videos, often confined to the accompaniment of a single song.

5.2.1 Musical Feature Extraction

Brouwer divides the musical features of a song into short- and long-term features. For the short-term features, Brouwer defines the chromagram and onset envelopes as being the most important musical features in terms of the amount of information they convey.

The chromagram divides the frequency spectrum into categories by musical pitch, being 12 notes in the Western musical scale. This results in an envelope per pitch which is high when a certain note is being played and low when a note is absent.

The onset envelope is a single envelope that peaks when a sudden change in the audio spectrum

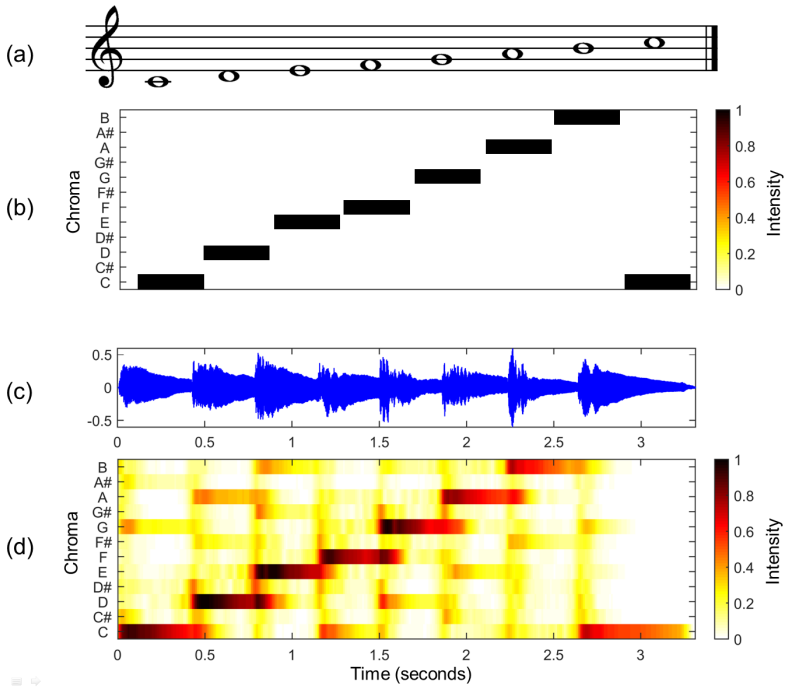


Figure 5.2: (a) Musical score of a C-major scale. (b) Chromagram obtained from the score. (c) Audio recording of the C-major scale played on a piano. (d) Chromagram obtained from the audio recording. [43]

occurs, for example when a drum hits. The chromagram thus conveys the pitch information of the song, while the onset envelope conveys the rhythmic information. On a longer term, Brouwer states that the Root-Mean Square (RMS), representing the average energy across the entire frequency spectrum, can give an indiction on different sections of a song based on the volume change. Moreover, he states that Laplacian segmentation can be used to analyse the hierarchical pattern of music to detect different sections automatically [45]. Brouwer defines two possible audio reactive interpolation methods. The first way is to select a set of latent vectors and loop in between them according to fixed looping patterns. These looping patterns could be synchronized to a fixed amount of bars of the song.

The second way is to match each note of the Western scale to a certain latent vector and take chromagram-weighted sums of the latent vectors at fixed time instances. This way, reoccurring chord progressions will always be visually represented in the same way, keeping long-term consistency.

Apart from the latent vectors, one can also modify the noise maps injected into each convolutional layer of the network. Brouwer proposes to multiply the drum onset at each time instant with

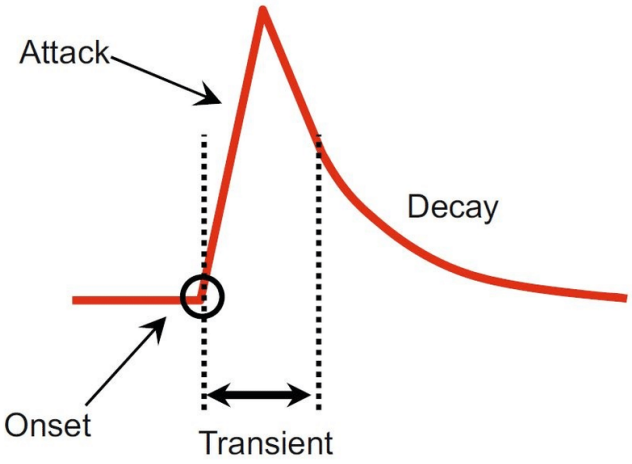


Figure 5.3: Visual representation of a note onset. [44]

the noise map, resulting in a chaotic spasm when drums hit. Brouwer states the advantage of the hierarchical structure of StyleGAN2 by proposing to map different parts of the frequency spectrum to different parts of the StyleGAN2 network hierarchy. One could for example make changes in the lower frequencies have an impact on large-scale/low-resolution visual aspects, while changes in melody could lead to more fine-grained visual changes.
A final technique proposed is the idea of applying transformations to intermediate outputs before passing them to the next layer, a technique called Network Bending [46]. More precisely, network bending is the process of inserting transformation layers in the CNN stack, so that selected CNN feature activation maps can be translated, rotated or scaled. Even morphological transformations like erosion or dilation are possible [47].

5.3 Making it real-time: the Autolume-Live project

The first attempt at making the concept of audio reactive latent interpolations work in a real-time setting has been made by Kraasch and Pasquier in 2022. Kraasch and Pasquier researched the possibility of creating a real-time video synthesizer solely based on deep generative modelling and succeeded in that remarkably well. We chose to deeply analyse the work by Kraasch and Pasquier and improve upon their work at all different performance levels, since their research paper accounts for the current state-of-the-art no in real-time audio reactive generative modelling. By thoroughly analysing the needs of the modern VJ, Kraasch and Pasquier developed the first live VJing system for controllable video generation, which extracts musical features like amplitude, pitch, and onset strength and maps those to trajectories in the latent space. Further manipulation of the visual output can be done through the use of a MIDI-control which allows for

real-time adjustment of the latent space trajectories and even the network parameters themselves through Network Bending, allowing for artists to be actively involved in the creative process.

5.3.1 The architecture

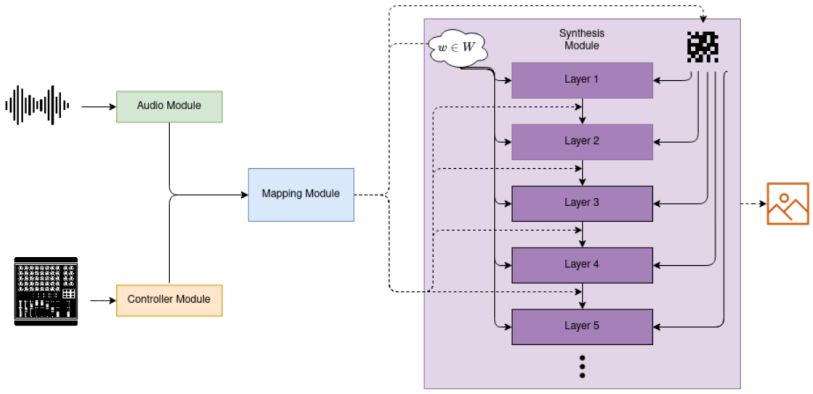


Figure 5.4: The Autolume-Live architecture. [20]

The Autolume-Live architecture is depicted in Figure 5.4. It consists of four separate modules: the audio module, the controller module, the mapping module and the synthesis module. The audio module is occupied with the musical feature extraction like pitch, amplitude, and onset strength. The controller module processes all interactions with the MIDI-controller. Both send their outputs to the mapping module that uses the two signals to decide on the precise latent trajectories to follow. It outputs a $\mathcal{W}$ latent that is sent through the synthesis module, in essence just a pretrained StyleGAN model, resulting in an image. This process repeats itself at a rate equal to the required frame rate.

5.3.2 The Synthesis Network

Brouwer used StyleGAN2 to generate an image at every time step of the visualisation. The reason for this being the lightweight nature of the models compared to more recent techniques like Diffusion Models, making real-time generation possible while delivering robust and qualitative image generation.
For training, the advantage of requiring smaller datasets lead Brouwer to employ StyleGAN2(-ADA), stating that dataset size is important for VJing since it offers artist the opportunity to train their own datasets faster and with smaller datasets. This gives artists the opportunity to create visualisations from smaller datasets and use their own work.
In order to reach the timing constraints necessary for real-time applications as stated in Section 2.2.2, Brouwer compressed the standard StyleGAN2 model by making use of content-aware

GAN compression [48], reducing the complexity of the network counted in number of FLOPS by a factor of 11.

5.3.3 Audio Analysis

To make the extraction of audio-features run live, Kraasch and Pasquier use a monophonic audio stream with a chunk size of 1024 and a sampling rate of 44100. They compute the mel-spectrogram, chromagram and perform local onset strength detection. The FFT window size for the chromagram and spectrogram calculation is 2048. The mel-spectrogram is used instead of the normal spectrogram, since it better represents perceived amplitude by the human ear. Onset detection and strength approximation is done by calculating the average RMS over the last second. If the amplitude is higher than the average RMS over the previous second, an onset is detected.

5.3.4 The Interpolation Algorithm

The Autolume-Live project offers 3 different mapping algorithms between the musical features and the $\mathcal{W}$ latent vectors: chroma-based interpolation, latent space traversal and latent space interpolation.
The latent space traversal and interpolation algorithms are both chroma-agnostic: they do not use chroma information in the process of selecting a certain latent vector at each time instant. The latent space traversal algorithm chooses a random direction vector, normalised so that its step size is always one. The latent space interpolation algorithm uses a predefined set of these direction vectors, possibly selected by the artist itself, and cycles between them during runtime. The link between the audio and the visualisation in both algorithms is made through multiplying the direction vectors with the average amplitude at a certain time step and with the difference between the average amplitude of the current time step and that of the previous time step. This leads to bigger visual differences when the volume at a certain point is high and when a drastic change in volume happens at a certain time step.
The chroma-based interpolation algorithm operates in much the same way as it did in the work of Brouwer, now in real-time.
Next to these mapping algorithms, the approach of Brouwer for noise modulation was followed, multiplying the onset strength at each time step instead, with the standard deviation of the noise injected to the StyleGAN2(-ADA) network. This leads to visual jitter upon onset detection.

5.3.5 Control

By connecting a MIDI-controller, the artist is able to efficiently adjust parameters of the interpolation algorithms and the network in real-time. Previously decided upon latent presets could for example be loaded by the artist to differentiate the latent interpolation algorithm over different sections of the performance.
Apart from this, however, Kraasch and Pasquier made it possible to perform Network Bending as also described by Brouwer through the use of a MIDI-controller. By manipulating the StyleGAN2 network architecture in real-time, the VJ can manipulate frames in much the same fashion as traditional VJing software like Resolume allows, adding rotations, zoom-effects and filters. This allows an artist to react to an atmosphere in real-time.

Brouwer also state that finding meaningful direction vectors in the latent space is a cumbersome task, given its 512-dimensional nature. To tackle this, they implemented GANSpace by Härkönen et al. in order to automate this process for the artist [49]. GANSpace is an unsupervised algorithm that identifies interpretable directions in the latent space by making use of Principal Component Analysis (PCA). It returns a list of direction vectors that result in interpretable visual changes, without extra overhead.

5.3.6 Achieved Performance

All measurements were made by testing a standard StyleGAN2 architecture on an NVIDIA QUADRO 5000 with 16 GB of VRAM. On this setup, Brouwer report a frame rate of 60 FPS when generating images at a resolution of 128×128. When the resolution is increased, the frame rate drops to 24 FPS at 512×512 and 40 FPS at 256×256.

6 Methodology

The Autolume-Live project introduced in Section 5.3 is unique in its kind and offers the first functional generative model-based VJing software, while setting the VJ up with a wide range of parameters to change on the fly to their liking [20].

The goal of this project is to analyse overall performance of the current state-of-the art offered by Autolume quantitively and analyse if other algorithms and design choices could potentially lead to performance improvements, new opportunities or enlarged creative potential. Performance is measured in terms of latency, frame rate, computational complexity, visual quality, diversity, stability, semantic quality and content coherence. This was done by reverse engineering the work done by, Kraasch and Pasquier and by critically analysing the design decisions that were made at each level of the pipeline [20].

6.1 Outline

The same system architecture is used as Kraasch and Pasquier used, depicted in Figure 5.4. The three modules: the *Audio Module*, the *Controller Module*, the *Mapping Module* and the *Synthesis Module* were all implemented as described by Kraasch and Pasquier [20].

With precise knowledge of their performance and functionality, alternative designs were implemented and tested for each module. A quantitative evaluation and qualitative discussion of both the initial system by Kraasch and Pasquier and the alternative system design choices are reported at the end of each module section [20].

The exact technical implementation of each module and the proposed design changes are reported in Chapters 7, 8, 9 and 10.

6.2 A word on Evaluation Criteria & Metrics

The quality of a VJing system is inherently subjective: if an audience thinks the visualisations are pleasing and elevate the music, a VJing system should be considered performant. In this research however, we attempt to quantify precisely how performant the VJing system is in terms of latency, frame rate, computational complexity, visual quality, diversity, stability, semantic quality and content coherence. Since all these criteria have an influence on the experience of the audience.

In what follows, an overview is presented of the metrics and evaluation methods used. Some of these metrics have been proposed in literature before, others have been specifically designed for this research project. In the end, some evaluations were performed by purely visually inspecting the visualisations themselves. In those cases, a subjective qualitative report is given in the discussion of the section.

6.2.1 Latency, Delay & Frame Rate

As stated in Section 2.2.2, real-time video with audio is subject to timing constraints in order to deliver the right experience to an audience. In order to measure the lip sync error, or audio-video delay, and consequently the frame rate, timing measurements have been performed. The time between the start of frame generation and the frame being rendered to screen will embody the lip sync error and is measured in milliseconds for different setups. The frame rate is reported in FPS or Hz. At the same time, this delay quantifies the computational complexity of the system.

6.2.2 Visual Quality: FID

Measuring the visual quality of the images generated by a generative model is known to be a hard task. Several methods have been proposed, but perhaps the most widely used metric to score individual images is the Fréchet Inception Distance (FID) [50].

The FID quantifies the realism and diversity of the images generated by a generative model. A realistic image would be one that closely follows the distribution of the training dataset. Image diversity implies that the generated images are novel and original compared to the training dataset.

Initially proposed by Heusel et al. to measure the quality of GANs, the FID algorithm extracts feature representations of a model using the Inception-v3 model trained by Google [51] and calculates the Fréchet distance of the feature representations of different images.

For any two probability distributions μ, ν over $\mathbb{R}^n$ having finite mean and variances, their Fréchet distance is given by Equation 6.1.

$$d_F(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x - y\|^2 \, \mathrm{d}\gamma(x, y) \right)^{1/2}, \tag{6.1}$$

where $\Gamma(\mu, \nu)$ is the set of all measures on $\mathbb{R}^n \times \mathbb{R}^n$ with marginals μ and ν on the first and second factors respectively. (The set $\Gamma(\mu, \nu)$ is also called the set of all couplings of μ and ν.).

Since the FID is a distance metric, the lower the FID score, the better the image quality generated by the trained generative model.

6.2.3 Entanglement: PPL

As stated in Section 4.2.5 latent space entanglement will have a profound impact on content coherence. It could for example be that a small musical feature change leads to a big visual change because of a dense local manifold in the latent space.

To measure latent space entanglement, the Perceptual Path Length (PPL) is used.
The PPL is a metric introduced in by Karras, Laine, and Aila in the original StyleGAN paper [36] to measure the smoothness of the interpolation between two points in the latent space.

As a basis for the metric, a perceptually-based pairwise image distance [52] is used that is calculated as a weighted difference between two VGG16 [53] embeddings, with weights fitted so that the metric agrees with human perceptual similarity judgements.

In the latent $\mathcal{Z}$-space the perceptual path length over all possible endpoints becomes

$$l_{\mathcal{Z}} = \mathbb{E}\left[\frac{1}{\epsilon^2} d\left(G\left(\mathrm{slerp}\left(\mathbf{z}_1, \mathbf{z}_2; t \right) \right), G\left(\mathrm{slerp}\left(\mathbf{z}_1, \mathbf{z}_2; t + \epsilon \right) \right) \right) \right], \tag{6.2}$$

,

where $\mathbf{z}_1, \mathbf{z}_2 \sim P(\mathbf{z}), t \sim U(0, 1)$, G is the generator, and $d(\cdot, \cdot)$ evaluates the perceptual distance between the resulting images. The interpolation technique used is spherical linear interpolation (slerp).
In the latent $\mathcal{W}$-space, this similarly becomes

$$l_{\mathcal{W}} = \mathbb{E}\left[\frac{1}{\epsilon^2} d(g\left(\mathrm{lerp}\left(f\left(\mathbf{z}_1 \right), f\left(\mathbf{z}_2 \right); t \right) \right), g\left(\mathrm{lerp}\left(f\left(\mathbf{z}_1 \right), f\left(\mathbf{z}_2 \right); t + \epsilon \right) \right) \right], \tag{6.3}$$

, with the only difference being that now lerp is used instead of slerp, since vectors in $\mathcal{W}$ space are not normalised.

In this project the heuristic version of the PPL is used, namely the endpoint PPL, where the PPL is calculated over the difference between the two endpoint embeddings instead of the embeddings of the whole interpolation.

6.2.4 Reactiveness & Coherence: Root Mean Square Variance to Temporal Motion Variability distance (RMSV-TMV)

Perhaps the most important quality measure of real-time audio visualisation is how reactive the imagery is to changes in the music. The visualisation is said to be coherent with the music when slowly varying sections of the music lead to slowly varying visualisations and high-variant sections lead to strongly varying visualisations.
In the context of real-time audio visualisation, there is no consensus on the methods or metrics to measure the coherence between the audio and video signal. For that reason, a novel metric is proposed, coined the *Root Mean Square Variance to Temporal Motion Variability ratio* or RMSV-TMV in short.
As the name suggests, the difference between the normalised variance of the RMS over an audio fragment and the temporal motion variability gets calculated to reflect how coherent and reactive the system is. The rationale behind this is that sections with fast changing volume should give rise to high-motion visualisations.

The Root Mean Square Variance gets calculated according to

$$\text{RMSV} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{\text{RMS},i} - \bar{x}_{\text{RMS}})^2, \tag{6.4}$$

with x_{RMS} calculated according to

$$x_{\text{RMS}} = \sqrt{\frac{1}{n}\left({x_1}^2 + {x_2}^2 + \cdots + {x_n}^2\right)} \tag{6.5}$$

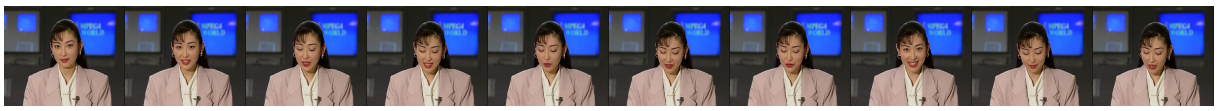
To calculate the Temporal Motion Variability, the optical flow between consecutive frames is calculated. This is done by employing the Färneback algorithm [54], through the `calcOpticalFlowFarneback` [1] method of the OpenCV [55] Python library.

The Färneback algorithm analyses the pixelwise movement between two consecutive frames and outputs directed displacement vectors for each pair of consecutive frames, translating the amount and type of motion each pixel undergoes in between frames.

An indication of the amount of motion over an entire video fragment is could then be calculated through summing the absolute norms of those displacement vectors between each pair of consecutive frames.

[1] Python-OpenCV documentation: `https://docs.opencv.org/4.5.4/dd/d00/classcv_1_1superres_1_1FarnebackOpticalFlow.html`

As a performance benchmark, this motion estimation approach has been performed on the `akiyo_cif` and `football_cif` benchmark video clips provided through the Xiph.org Test Media dataset [56], popular for video compression and processing. The `akiyo_cif` clip is known to have very little movement, while the `football_cif` clip has a high degree of motion. This is clear through visual inspection of both clips, and is reflected in the motion degrees calculated through the Färneback optical flow algorithm, visualised in Figure 6.2.



(a) `akiyo_cif`

(b) `football_cif`

Figure 6.1: 10 equidistant frames extracted from the two Xiph.org Test Media benchmark clips.

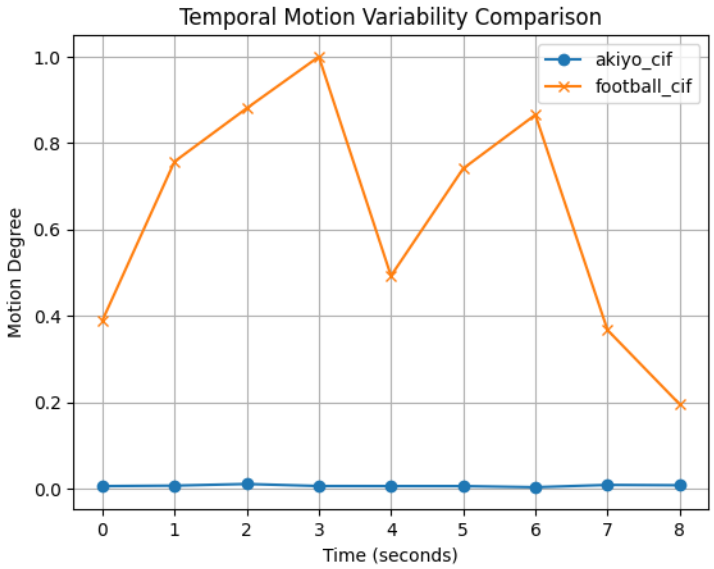


Figure 6.2: Färneback optical flow motion degree comparison between `akiyo_cif` and `football_cif`

By normalizing both the RMSV and the TMV, the MAE can be calculated between both, which leads to the expression for the RMSV-TMV error as described in Equation 6.6.

$$\mathbb{E}[\text{RMSV-TMVD}] = \frac{1}{n}\sum_{i=1}^{n} |\text{RMSV}_i - \text{TMV}_i| \tag{6.6}$$

with n the amount of separate fragments to calculate the RMSV-TMV over.

The performance of the metric was assessed by calculating the RMSV-TMV distance of the 20 first seconds of an audio-video sync test video [2]. The clip shows a white square every second combined with a short beeping sound, they are in perfect sync.

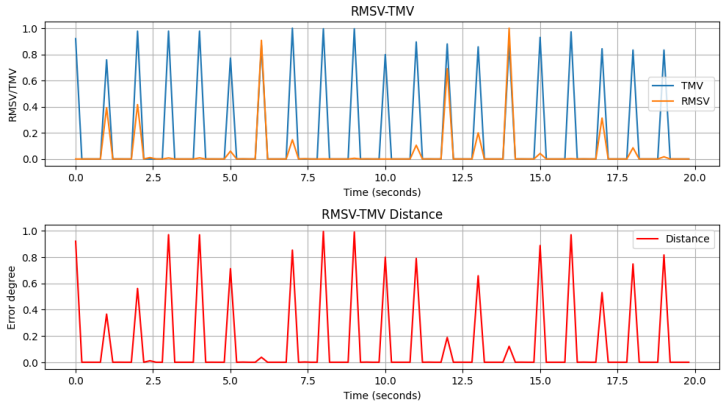


Figure 6.3: RMSV-TMV metric assessed on 20 seconds of an audio-video sync test clip.

From Figure 6.3 it is clear that the RMSV and TMV values follow the same trends. However, the RMSV values vary a lot more than the TMV ones. Due to normalization, this leads to some RMSV peaks being completely suppressed with a bigger RMSV-TMV distance as a consequence. This shows that the RMSV-TMV gives a correct idea on the coherence of audio and video, but has its imperfections. Comparing the values of RMSV and TMV over time visually in combination with looking at means is the suggested option when evaluating coherence performance during this study.

All measurements and visualisations in this project were run on an Nvidia Geforce RTX 3070 Ti Laptop GPU with 8 GB VRAM.

[2]Test Test. "VIdeo in perfectsync(AV sync Test)" YouTube video, 0:40, Posted October 20, 2016, `https://www.youtube.com/watch?v=d5TT12WUll0` (accessed August 16, 2023).

7 Feature Extraction & Signal Processing

When an audio visualisation run starts, the first objective to fulfil is the extraction of meaningful features from the audio stream. This process needs to continue up until the termination of the system. Depending on the complexity of the feature, extracting musical features from audio generally requires expensive signal processing. Brouwer could afford these more expensive operations, like splitting the audio stream on harmonic and percussive elements, since they were operating offline, as explained in Section 5.2 [26]. Since the system needs to operate in real-time, the feature extraction needs to stay relatively simple and needs to be strongly optimized. In this section, the exact methods of operation are explained to extract the volume of an audio stream and detect onsets and their strengths.

The audio input stream is created through the PyAudio Python package [57]. A monophonic stream was used with a chunk size of 1024, a sampling rate of 44100 Hz and an FFT window size of 2048.

7.1 Volume extraction

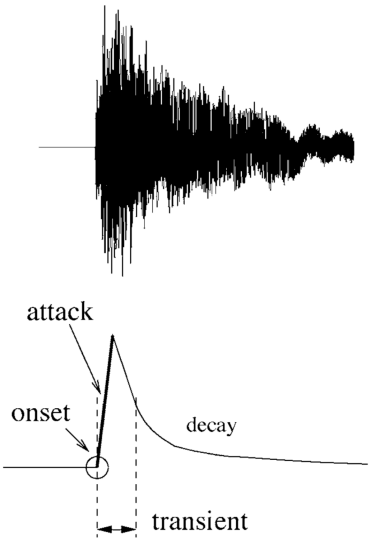
The volume is an informative feature of a piece of music and a fairly lightweight one to extract. In a mathematical sense, it is reflected in the average root-mean-square (RMS) of an audio chunk. For each audio chunk of 1024 audio samples the average RMS of the amplitude is calculated according to Equation 6.5, with n now being equal to the chunk size of 1024.
In this project, the RMS was calculated through the `librosa.feature.rms` [1] method present in the Librosa Python package [58].

[1] Documentation: `https://librosa.org/doc/main/generated/librosa.feature.rms.html`

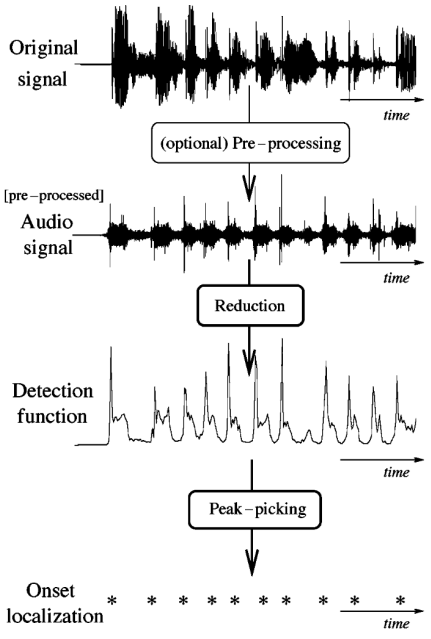
7.2 Onset detection & strength estimation

The goal of music visualisation, in essence, is to visualise important musical events. Although the volume is an indication of this, another way to detect important musical events is to perform music note onset detection.
In order to understand the concept of the *onset* of a musical note, one needs to understand the concepts of *attack* and *transient* as well.

- The *attack* of a note is the time interval during which the amplitude envelope increases.
- The *transient* is more difficult to define precisely, but is most often described as the short interval during which the signal evolves quickly in a non-trivial or relatively unpredictable way.
- The *onset* is in most cases defined as the earliest time at which the transient can be reliably detected.



(a) Visual representation of the concepts of transient, attack and onset.



(b) Flowchart of a standard onset detection algorithm.

Figure 7.1: From the works of Bello et al. [59].

Figure 7.1b shows the standard onset detection pipeline according to Bello et al. In a first step, the signal is pre-processed in order to further pronounce some features that could help in the detection of the onset. Examples are background noise filtering, normalization and pre-emphasis.

The pre-processed signal is then reduced to a detection function, which is a highly subsampled version of the audio signal manifesting the occurrence of transients in the original signal. If a well-designed reduction algorithm is used, the detection function should clearly visualise local maxima or peaks. By making use of a robust peak-picking algorithm, the onsets can finally be detected.
The exact details and algorithms to choose from to perform each step of the detection pipeline can be found in the paper by Bello et al. [59].

Real-time onset detection however is still a though nut to crack. The reason for this is that most onset detection algorithms use a peak-picking algorithm that relies on future information to determine the location of an onset. Böck et al. proposed a real-time approach only using causal audio signal information by employing an RNN [60]. Although slightly less performant than its offline counterpart, the approach by Böck et al. reaches a precision of 0.850 and a recall of 0.787. The work approach by Böck et al. was implemented in the `madmom.features.onsets.RNNOnsetProcessor` [2] method included in the madmom [61] Python library.

During this research, the audio chunks were preprocessed by first denoising and then performing normalization. Before feeding the signal through the `RNNOnsetProcessor`, the signal is sent through a pre-emphasis filter. The pre-emphasis filter is an essence a first-order high-pass filter that emphasizes higher frequencies relative to lower ones.
The output of the `RNNOnsetProcessor` is sent through an expansion filter, emphasizing high onsets and suppressing low ones. The quality of the pipeline was evaluated visually through Figure 7.2.

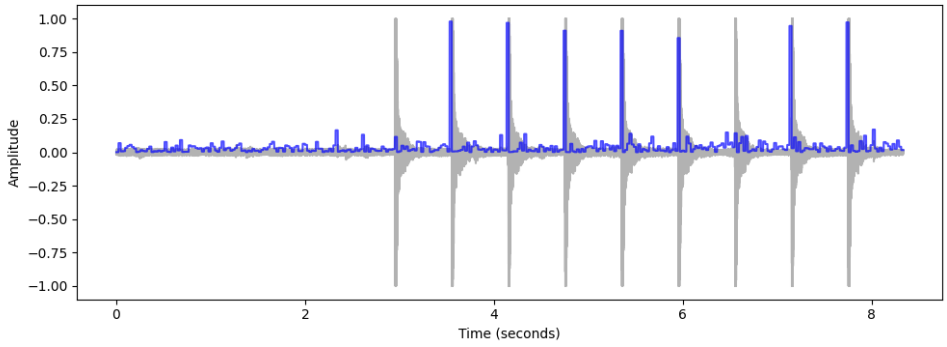
It should be clear that the onset estimation is far from perfect, but seen the real-time constraints, it is acceptable for the current application and could be improved upon in later work.

7.3 Performance

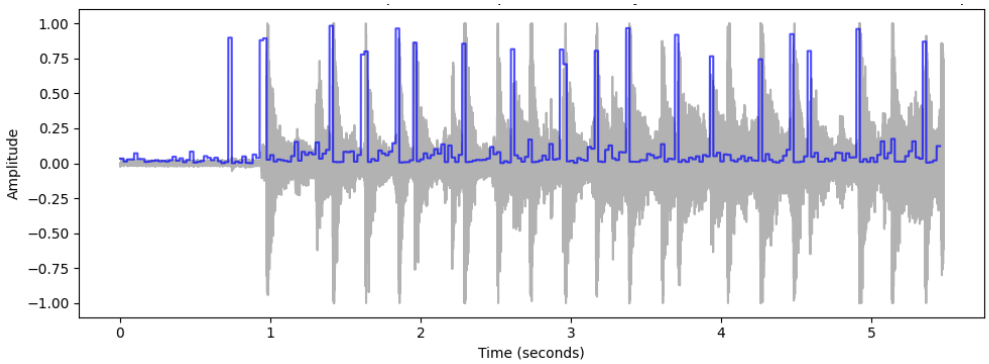
Since audio analysis is a crucial part of the real-time system, its performance should be both qualitative and efficient. By using multiprocessing, the process of musical feature extraction can run in parallel with image generation and rendering, but could slow down the overall system.
In order to quantify the slow-down degree of extracting different features, an audio visualisation run has been timed in terms of lip sync error using the *Hypersphere Interpolation* algorithm explained later on in Chapter 8. The synthesis model used for testing is the FFHQ config-f 512 $\times$ 512 pretrained StyleGAN2 model [3].

[2] Documentation: `https://madmom.readthedocs.io/en/v0.16/modules/features/onsets.html#id20`

[3] Justin Pinkney. (n.d.). Awesome Pretrained StyleGAN2 Models. GitHub Repository. `https://github.com/justinpinkney/awesome-pretrained-stylegan2`



(a) Waveform visualisation with onset estimations overlaid of a 100 BPM metronome real-time recording.



(b) Waveform visualisation with onset estimations overlaid of a real-time recording of a contemporary dance song.

Figure 7.2: Waveform visualisations with Onset Estimations

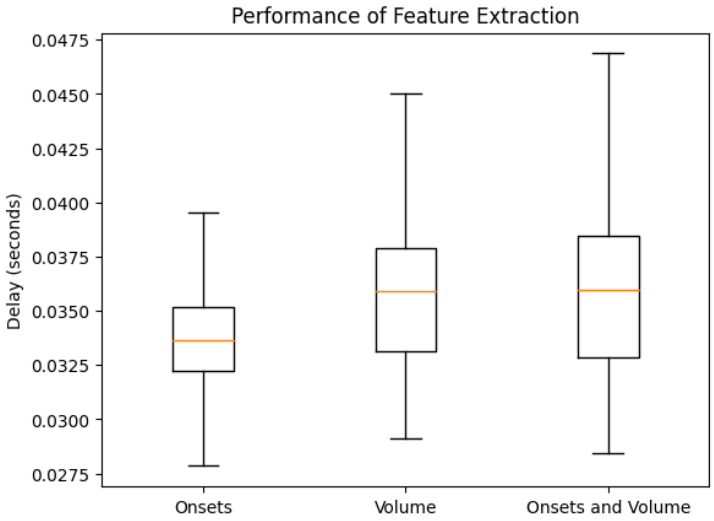


Figure 7.3: Timing measurements of generation times by generating 10,000 frames through hypersphere interpolation on the FFHQ config-f 512 × 512 StyleGAN2 model.

It should be clear that the nature of the feature being extracted only has a very slight impact on the mean performance of the system, with onset extraction being more lightweight than volume extraction. A volume-base system leads to higher variability in delay, which might lead to inconsistent frame rates.
As shown in Figures 7.2a and 7.2b, onset extraction is not completely reliable. Apart from that, the delay due to volume extraction is similar to the delay caused by volume-and-onset-extraction. It thus appears to be the best choice to combine volume and onset extraction in order to maximize the information extracted from the music while keeping the incurred delay to a minimum.

7.4 Open Sound Control

Given the inherent unreliability of real-time signal processing, as illustrated in Figure 7.2, the software created in this project also allows communicating musical cues through the use of Open Sound Control or OSC [4].

By exchanging OSC-messages over the network, information on the music at hand could be sent from the musicians' system to the system of the visual artist with minimal latency. A setup like this was experimented with and is presented in Appendix D.2.2.

[4]Official Open Sound Control website: https://opensoundcontrol.stanford.edu/

8 Constructing audio reactive Latent Walks

Kraasch and Pasquier could not afford to use expensive audio processing and interpolation algorithms, seen constraints on the frame rate and audio-video delay or lip sync error as discussed in Section 2.2.2 [20]. For that reason, the interpolation technique used by Kraasch and Pasquier was kept simple.

In this study, two of the algorithms proposed by Kraasch and Pasquier are implemented and evaluated in order to have a baseline of comparison. This is necessary since the paper by Kraasch and Pasquier delivered a qualitative rather than a quantitative report, where reported performance results are kept to a minimal. The only reported performance measurement is the FPS, but that measurement is strongly system dependent as well. By implementing their algorithms and running them on the same machine as the improved algorithm, a comparison can be made between their proposed approach and the one described here. In the following chapter, the exact functionality of three different interpolation algorithms are explained, quantitative results of each are given at the end of each subsection.

8.1 Latent Space Traversal

Algorithm 1 Algorithm for Latent Space Traversal

procedure LATENT_SPACE_TRAVERSAL
  while Music is playing **do**
    Choose a random source latent vector $\mathbf{x}_s$. ▷ Initialization
    Choose a random direction latent vector $\mathbf{d}$.
    Normalize $\mathbf{d}$ so that it leads to a step size of 1.
    Extract the RMS and onset of the audio. ▷ Feature extraction
    Multiply $\mathbf{d}$ by the RMS.
    Add $\mathbf{d}$ to $\mathbf{x}_s$ to find the destination latent $\mathbf{x}_d$.
    for $i = 0$ to n_{frames} **do**
      $\mathbf{x}_i = \mathbf{x}_s \cdot \frac{i}{n_{\text{frames}}} + \mathbf{x}_d \cdot \left(1 - \frac{i}{n_{\text{frames}}}\right)$ ▷ Interpolation
    end for
    for $i = 0$ to n_{frames} **do**
      Multiply onset value with noise vector $\mathbf{n}$. ▷ Noise Injection
      Feed $\mathbf{x}_i$ through the synthesis network. ▷ Synthesis
      Render frame to the screen. ▷ Rendering
    end for
  end while
end procedure

The first implemented algorithm is an interpretation of the latent space traversal algorithm proposed by Kraasch and Pasquier as sketched in Section 5.3.4 and as presented in pseudocode in Algorithm 1.

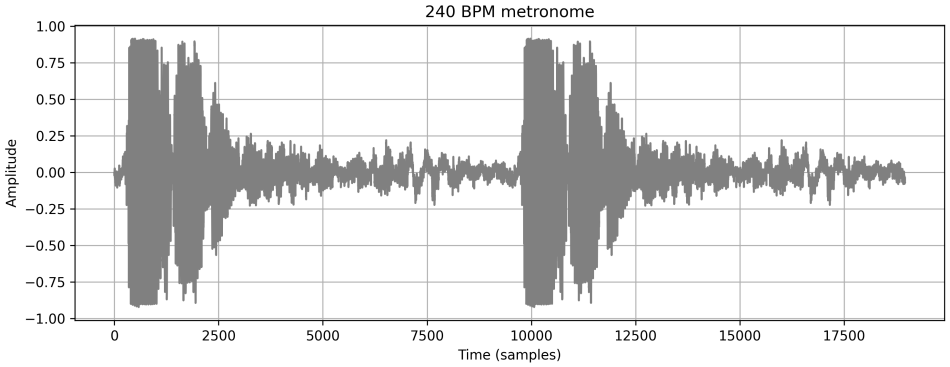
Upon startup, the system chooses a random source latent vector and a random direction latent vector. The direction latent vector is normalized so that it always leads to a step size of 1. The direction latent vector is multiplied by the RMS extracted by the audio-handler and added to the source latent in order to find the destination latent. Subsequently, n_{frames} intermediary latent vectors are calculated through linear interpolation.

$$\mathbf{x}_i = \mathbf{x}_s \times \frac{i}{n_{\text{frames}}} + \mathbf{x}_d \times \left(1 - \frac{i}{n_{\text{frames}}}\right), \quad i = 0, 1, \ldots, n_{\text{frames}} \tag{8.1}$$

where $\mathbf{x}_s$ is the source frame, $\mathbf{x}_d$ is the destination frame, and i ranges from 0 to n_{frames}.

When all latent vectors have been calculated, they are fed through the synthesis network one by one. From the moment all frames have been rendered to screen, the process repeats itself.

The onset value is used for noise modulation. As described by Brouwer and implemented by Kraasch and Pasquier, one modulates the stochastic variation of the generated images by multiplying the onset with the noise injected to the different convolutional layers. Since StyleGAN2 uses a progressive growing training algorithm, one can choose at which resolution the stochastic variation should be modulated by the onset. Figure 8.1 shows the effect of noise modulation on the second convolutional layer.



(a) 240 BPM metronome track.

(b) 10 equidistant frames.

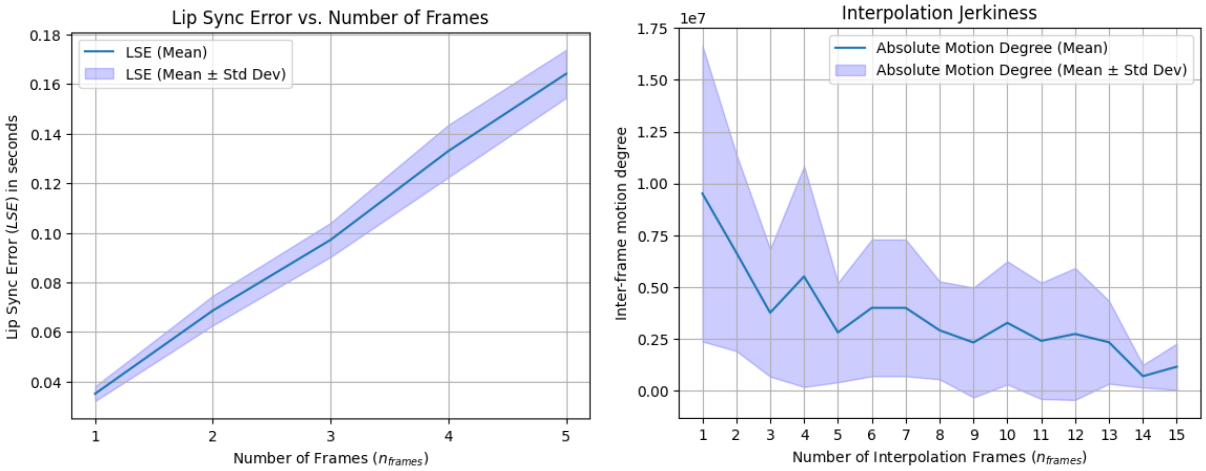
Figure 8.1: Onset reactive visualisation through noise injection and modulation.

Using a random latent walk, as a way of music visualisation, relies on heavy computations. Each time a destination latent is reached, a new random latent needs to be selected, and n_{frames} intermediary latent vectors should be calculated. Even when no new musical input is received, this process continues. This leads to small video judder, since the visualisation runs smooth during the frame generation phase and noticeably stops running during the interpolation phase. The value of n_{frames} has an impact on that judder. The bigger the value for n_{frames}, the longer an interpolation phase will take, since more intermediary latent vectors need to be calculated. At the same time, more intermediary frames will have to be rendered to screen in between source and destination latent, leading to higher lip sync error (LSE) by definition: the delay between the musical feature and its visual effect will be higher. The equations below formalize this behaviour.

$$\begin{aligned} \mathbf{LSE}_{\text{min}} &= D_i + D_g \\ \mathbf{LSE}_{\text{max}} &= D_i + n_{\text{frames}} \cdot D_g, \end{aligned} \tag{8.2}$$

with D_i the interpolation delay and D_g the generation delay.

Although a first visual reaction to the music will be visible after a delay of D_i, the peak visual reaction to a musical cue will only be completed after an LSE given in Equation 8.2.



(a) Maximum Lip Sync Error (LSE) as a function of n_{frames}.

(b) Visualisation jerkiness as a function of n_{frames}.

Figure 8.2: Hyperparameter tuning of n_{frames}.

From Figure 8.2 it should be clear that the LSE scales linearly with n_{frames} when n_{frames} is small enough. The interpolation times do not grow significantly for small n_{frames}. It should thus be clear that n_{frames} should be chosen small in order to keep the LSE under the real-time constraints specific in Section 2.2.2.

At the same time, the value of n_{frames} influences the smoothness of the generated interpolation, with a small value leading to very abrupt jumps between different latent vectors and as a consequence image content. A weigh-off needs to be made between video judder and interpolation smoothness.

8.1.1 Recurrence

The selection of random latent vectors for interpolation lacks the coherent replication of recurring images, a characteristic element of both VJing and musical compositions. The process of selecting novel latent vectors for interpolation leads to low probabilities of selecting the same latent vector twice. In the event that such repetition occurs, it does not establish a meaningful correspondence with the recurrent patterns observed in the musical composition itself.

Almost all types of music heavily rely on repetition in order to create structure in a composition, helping to unify rhythm and melody. A fissure might appear between music and visualisation

when the music repeats itself, but the visualisation does not.

8.1.2 The problem of the drunk bird, truncation, and the manifold hypothesis

Perhaps the biggest problem with the random latent walk approach is the fact that the walk does not centre around the starting latent throughout the interpolation, although one would intuitively expect it to do so, since each direction is chosen randomly.
This means that after a while, the walk drifts away to a specific location in the latent space and gets *stuck* there. With getting stuck, we mean that the images start to change only slightly, not visualizing the music properly any longer.

This effect is visualized in Figure 8.3 by measuring the motion degree over chunks of 1 second and plotting its evolution over time. Although a constant amplitude audio input was sent through the system, a clear downward trend is visible in terms of the amount of motion between frames.

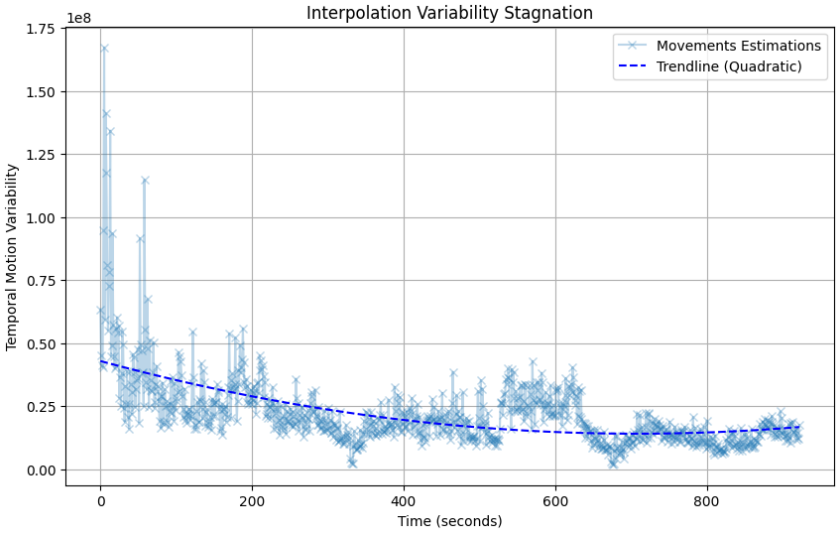


Figure 8.3: The amount of movement between frames shows a downward trend over time, although a constant amplitude input is sent through the system.

The reason for this is two-fold.
First, random walks in multidimensional spaces do not behave the same as random walks in one or two dimensions. Originally stated by Pólya in 1921 Theorem 8.1.1 states that random walks in one or two dimensions are recurrent, meaning that they are sure to pass through the same points more than once during their lifetime, while a random walk in higher dimensions is transient, with positive probability it never returns to its starting location.

Theorem 8.1.1 (Recurrence of random walks)**.** *A simple random walk in dimension d is recurrent for d = 1, 2 and transient for d >= 3 [63].*

Popov states several proofs of this theorem in his book *'Two-Dimensional Random Walk: From Path Counting to Random Interlacements'* [63]. And mathematician Shizuo Kakutani summarizes it poetically in the following statement

> "A drunk man will find his way home, but a drunk bird may get lost forever." – *Shizuo Kakutani*

The StyleGAN2 latent space is a 512-dimensional one. According to Theorem 8.1.1 it is thus clear that a drift in a certain direction will occur in each latent walk.

This, however, does not fully explain why the visual change stagnates after a while of walking randomly in the latent space. If the latent space would be uniformly distributed, the variation at the edges of the latent space should be as large as variation towards the centre of the latent space. If the images seem to stay the same after drift occurred, this can not be the case.

A possible explanation for this could be found in the manifold hypothesis, stating that real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space [64]. Applying this to the high dimensional latent spaces of GANs, it could be that near the edges of the latent space, less data is occurring than towards the center.

The truncation value of the StyleGAN2 architecture will directly influence this perceived stagnation as well. The truncation value decides the sampling space from the latent space. A low truncation value reduces the diversity of the visual content, while a truncation value of 1 results in more abstract, expanded visual space [20]. In essence, the truncation value decides on the effective dimensionality or *semantic size* of the latent space.

When you set the truncation value to 1.0, the full range of the latent space is explored during image generation: all 512 dimensions are actively used, and the generated images can exhibit a wide variety of visual features and styles. This results in more diverse and creative images, as the generator has the freedom to access a broader range of latent features.

On the other hand, when you set the truncation value to a lower value, such as 0.5 or 0.7, it restricts the exploration of the latent space. Only a subset of dimensions in the latent space is used during the generation process, effectively reducing the effective dimensionality or *semantic size* of the latent space. As a consequence, the generator focuses on a narrower set of latent features, leading to more consistent and controlled image synthesis. The generated images may look more alike and have reduced variability, but they often exhibit higher quality and more stable appearance.

It should thus be clear that a lower truncation value will lead to faster image stagnation when using the random latent space traversal algorithm sketched above.

8.2 Latent Space Interpolation

Algorithm 2 Algorithm for Latent Space Interpolation

procedure LATENT_SPACE_INTERPOLATION
  while Music is playing **do**
    Select a source latent vector $\mathbf{x}_s$ from the presaved set. ▷ Initialization
    Select a direction latent vector $\mathbf{d}$ from the presaved set.
    Normalize $\mathbf{d}$ so that it leads to a step size of 1.
    Extract the RMS and onset of the audio. ▷ Feature extraction
    Multiply $\mathbf{d}$ by the RMS extracted by the audio-handler.
    Add $\mathbf{d}$ to $\mathbf{x}_s$ to find the destination latent $\mathbf{x}_d$.
    for $i = 0$ to n_{frames} **do**
      $\mathbf{x}_i = \mathbf{x}_s \cdot \frac{i}{n_{\text{frames}}} + \mathbf{x}_d \cdot \left(1 - \frac{i}{n_{\text{frames}}}\right)$ ▷ Interpolation
    end for
    for $i = 0$ to n_{frames} **do**
      Multiply onset value with noise vector $\mathbf{n}$. ▷ Noise Injection
      Feed $\mathbf{x}_i$ through the synthesis network. ▷ Synthesis
      Render frame to the screen. ▷ Rendering
    end for
  end while
end procedure

The second algorithm proposed by Kraasch and Pasquier is the latent space interpolation algorithm [20].
Instead of choosing a random latent direction upon each interpolation phase, a set of interesting direction latent vectors is decided upon beforehand, after which a cyclic interpolation is repeated for as long as an artist chooses it to. Since The interpolation itself still works according to Equation 8.1 where now all direction latent vectors are decided upon beforehand, only a qualitative performance evaluation has been performed.

8.2.1 Discussion

Removing the random aspect of the latent space traversal algorithm, the latent space interpolation algorithm effectively solves the problem of transient walks and the subsequent stagnation of visual variation.
Moreover, due to the fact that a limited set of direction latent vectors is used, the artist can now introduce recurrence, using the same set of direction latent vectors for similar sections. A chorus

could for example have its own set of direction vectors, while the verses have another.
Since the artist can now choose the direction vectors being used, a level of control is added. There now is a possibility to try out different sets of vectors and choosing the most visually pleasing ones. If the VJ knows the music that will be played, he can even adapt the visual content of the direction vectors to the musical style. Other techniques like latent space projection and closed form factorization, discussed in Sections 9.3 and 9.2, in combination with latent space interpolation could give the freedom to an artist to visualize whatever content he wishes.

8.3 Hypersphere Interpolation

Algorithm 3 Algorithm for Hypersphere Interpolation

procedure HYPERSPHERE_INTERPOLATION
    Initialize the circular trajectory parameters: coefficient N, radius r, and centre $\mathbf{c}$.
    Calculate P using Equation 8.4 based on BPM, FPS and N. ▷ Initialization
    while Music is playing **do**
        for $i = 0$ to P **do**
            Generate a new random direction vector $\mathbf{d}$.
            Normalize $\mathbf{d}$ to ensure a consistent step size.
            Multiply $\mathbf{d}$ by the audio RMS and degree α.
            Calculate $\mathbf{x}$ according to Equation 8.3, with ϕ_j equal to $\frac{2\pi}{P}i$.
            Add $\mathbf{d}$ to $\mathbf{x}$.
            Multiply onset value with noise vector $\mathbf{n}$. ▷ Noise Injection
            Feed $\mathbf{x}$ through the synthesis network. ▷ Synthesis
            Render the frame to the screen. ▷ Rendering
        end for
    end while
end procedure

It is clear that both latent space traversal and latent space interpolation bring upon quite some overhead computation. Every interpolation step requires the calculation of n_{frames} intermediate latent codes, even when the visuals barely need to change, leading to visually noticeable judder when n_{frames} is large.

In this section, a novel algorithm is proposed, where instead latent vectors are visited on a spherical trajectory, avoiding almost all the problems stated for the previous algorithms.

To achieve this circular periodic trajectory, an n-sphere in 512 dimensions needs to be described somewhere in the GAN latent space. The n-sphere is a generalization of the normal sphere in 3 dimensions, and thus the set of points in $(n+1)$-dimensional Euclidean space that are situated at a constant distance r from a fixed point, called the centre [65].

An n-sphere in a general $(n+1)$-dimensional Euclidean space can be described in spherical coordinates according to Equation 8.3.

$$
\begin{aligned}
x_1 &= r\cos(\phi_1) \\
x_2 &= r\sin(\phi_1)\cos(\phi_2) \\
x_3 &= r\sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\
&\vdots \\
x_{n-1} &= r\sin(\phi_1)\ldots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\
x_n &= r\sin(\phi_1)\ldots\sin(\phi_{n-1})\cos(\phi_n)
\end{aligned}
\tag{8.3}
$$

Where r represents the sphere radius and ϕ_i angles ranging over $[0, 2\pi)$.

By varying all ϕ_i from 0 to 2π in steps of $\frac{2\pi}{P}$ with P the period, a circular trajectory can be described in a latent space of any dimension n with each dimension being traversed. It is clear that coordinates can be kept constant as well to keep certain features constant in a disentangled latent space.

8.3.1 Making it audio reactive

The hypersphere approach offers 3 different visual parameters to map to musical parameters: the period P, the radius r and the hypersphere centre $\mathbf{c} = [x_1, x_2, x_3, ..., x_{n-1}, x_n]$.

By interpolating on the surface of a hypersphere, a visualisation loop is now introduced by standard, that can be matched to the BPM of the music being played so that one visual loop coincides with a whole number of beats in the music being played. To achieve this the traversal period P should be related to the BPM according to Equation 8.4

$$
\mathrm{P} = \frac{\mathrm{FPS}}{\mathrm{BPM} \times 60} \times \mathrm{N}, \tag{8.4}
$$

with BPM the tempo of the music played, FPS the average amount of frames generated per second and N a chosen coefficient.

The higher the N value, the more beats will be included in one visual period and the slower the visualisation will run by default. In practice, the BPM and FPS estimations will not be completely accurate and resets will be necessary in order to avoid drift. In the rest of this chapter, drift will be neglected.

The choice of mapping the musical features to the hypersphere parameters, is left to the artist. The test setup for evaluation was designed to move the centre of the hypersphere with a factor based on the volume strength towards a direction decided upon by semantic meaning. Methods to choose this direction are explained in Chapter 9.

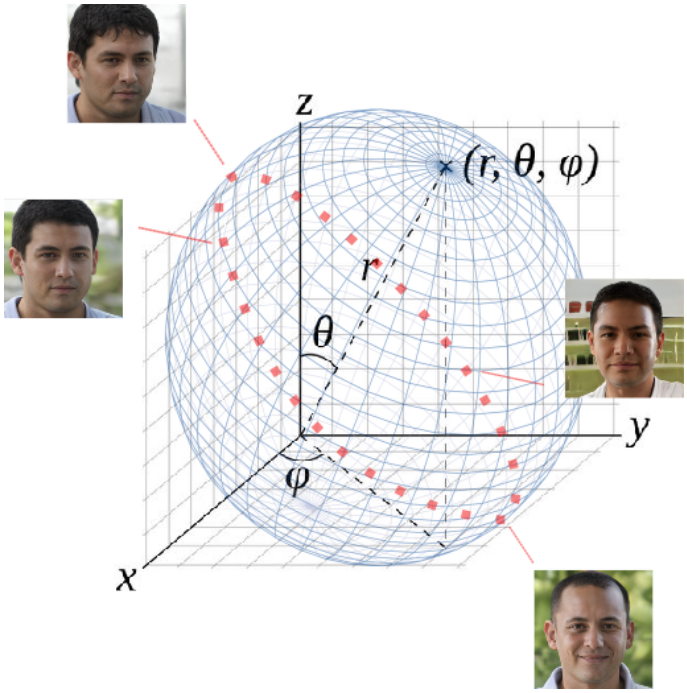


Figure 8.4: Three-dimensional visualisation of a latent space interpolation on the surface of a 2-hypersphere with radius r and centre $(\frac{1}{2},\frac{1}{2},\frac{1}{2})$.

Moving the centre of the hypersphere should happen in n_{frames} interpolation steps to avoid too abrupt changes in the visualisation. After the n_{frames} of moving in the chosen direction, the hypersphere centre should move to its original position again. This effect is represented by the degree parameter α in Algorithm 3.

It should be clear that the functionality of th n_{frames} parameter is similar in the context of the Hypersphere Interpolation algorithm, as it was in the context of the Latent Space Traversal algorithm in Section 8.1. However, since no interpolation phases are necessary when moving on the surface of an n-sphere. The risk of judder for high values of n_{frames}.
Following the same reasoning as in Equations 8.2, the formulation of the minimal and peak lip-sync error now simplify to

$$\begin{aligned} \mathbf{LSE}_{\text{min}} &= D_g \\ \mathbf{LSE}_{\text{max}} &= n_{\text{frames}} \times D_g \end{aligned} \tag{8.5}$$

with D_g again representing the generation delay.

The onset strength is injected in the noise in the same manner as described for the latent space traversal algorithm in Section 8.1.

8.3.2 Entanglement

Since the trajectory is perfectly circular, it can be expected that the images represented by latent vectors on that circular trajectory visually change at a constant rate when the sphere is traversed in a uniformly distributed latent space. However, as noted before in Section 4.2.5 the latent spaces of generative models are never completely uniform. Instead, the mapping of latent vectors to images is a rather complex and non-linear process due to latent space entanglement. This leads to inconsistent change rates of the visuals produced by traversing the circular trajectory, leading to an audio-video mismatch of heavy visual change when no musical change occurs.

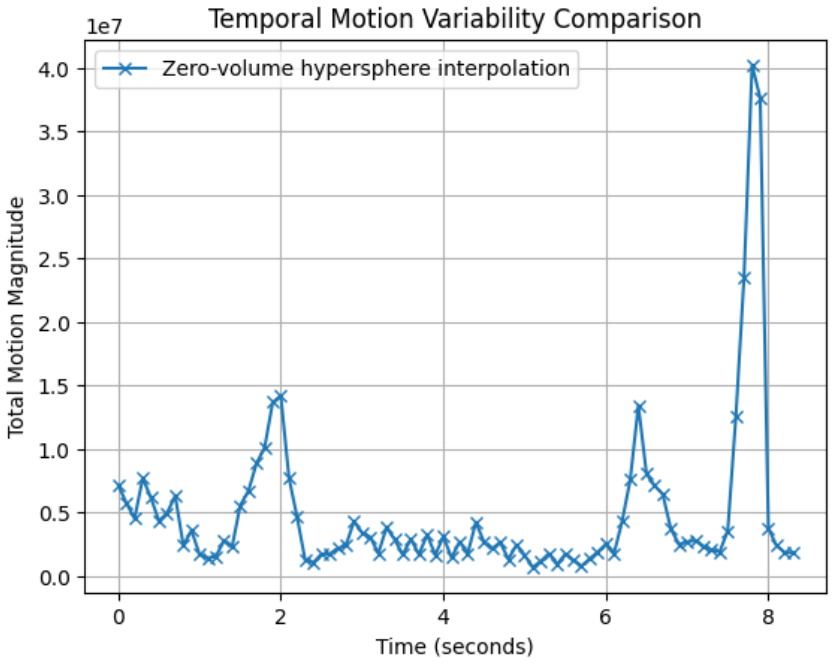


Figure 8.5: TMV of a zero-volume FFHQ 512×512 hypersphere interpolation with $r = 100$ and $c = [1, 1, ..., 1, 1]$.

This leads to a complex process of fine-tuning the radius r and hypersphere centre $\mathbf{c}$. Upon visual inspection, it was for example clear that a dense manifold of varied data lies around the $[0, 0, ..., 0, 0]$ latent in the FFHQ 512×512 latent space. If a $\mathbf{c}$ and r were chosen that created a hypersphere passing through this manifold and r is large, very fast TMV was noticeable when passing through latent space coordinates around $[0, 0, ..., 0, 0]$.

From visual inspections, it became clear that a smaller r tends to make the differences in TMV between dense and dispersed latent space regions smaller.

Entanglement is an inherent property of the latent spaces of generative models, leading to unfixable video-audio mismatches present in the latent interpolations created in this manner. A

way to mitigate this effect, is by choosing the hypersphere centre as a point in the latent space with a close to uniform distribution of images and choosing the radius in such a manner that no non-uniform regions are passed. The problem of finding these uniform regions is tackled in Chapter 9.

8.4 Quantitative Comparison

In order to get a proper idea of the performance of the different algorithms proposed in this section, quantitive measurements have been performed on both a Latent Space Traversal setup and a Hypersphere Interpolation setup. Since the Latent Space Interpolation algorithm has very similar structure to the Latent Space Traversal algorithm, their performance is supposed to be equivalent in terms of latency, frame rate, computational complexity, reactiveness, and coherence.

In order to compare both algorithms, specific test setups have been designed.
The n_{frames} value for both algorithms was chosen to be equal to 3, based on the conclusions made through Figure 8.2.
For the hypersphere interpolation algorithm, the choice was made to describe a hypersphere along the first 40 dimensions of the latent vector. This choice was made upon visual inspection of the results and the empirical finding that the hypersphere dimensionality has no impact on the frame generation time and thus latency, as visualised in Figure 8.6.

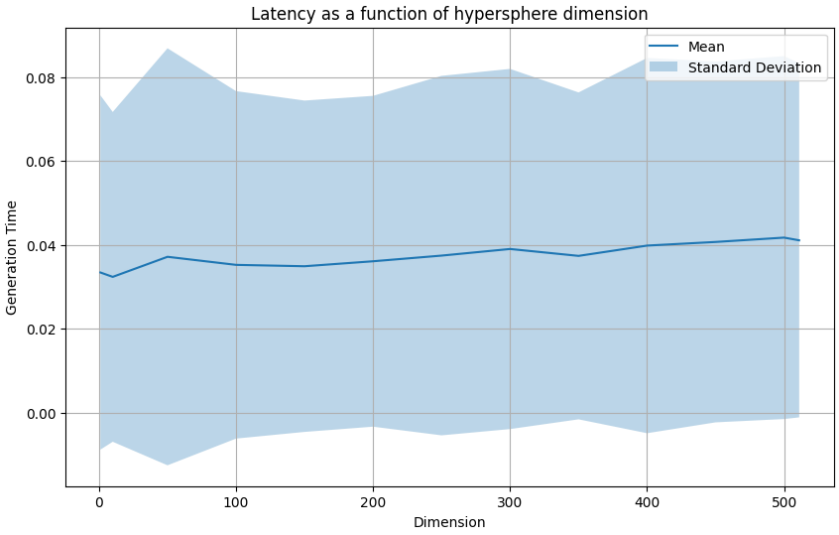


Figure 8.6: Empirical assessment of dependance latency and hypersphere dimension.

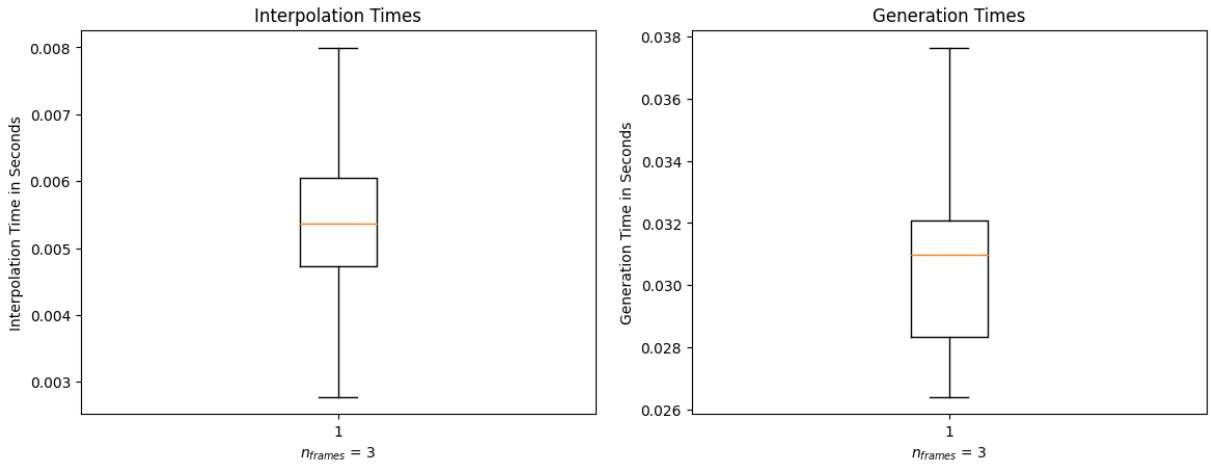
The volume was employed to move the centre of the hypersphere into a latent space direction chosen by the artist in $n_{frames} = 3$ generation steps, meaning that the full lip sync error lies at 3 generation steps.

The standard centre was chosen to lie at $\mathbf{c} = [1, 1, ..., 1, 1]$.
The radius was chosen to be constant with a value $r = 10$.

All reactivity and coherence tests were done through playing 'So U No' by 'Overmono' [1], with the volume being increased every 3 seconds, with a total length of around 67 seconds. The BPM was matched to 130, the BPM of the test track 'So U No' by Overmono, with $N = 50$.

8.4.1 Latency, Frame Rate & Computational Complexity

Using the Latent Space Traversal algorithm, leads to a maximum lip sync error of around 85 ms, but already a visual reaction to a musical cue after one interpolation cycle and one generation cycle, which on average takes 36.5 ms, since generation of one frame on average takes 31 ms and an interpolation cycle on average takes 5.5 ms. The average achievable frame rate is 32 FPS.



(a) Interpolation times.

(b) Generation Times.

Figure 8.7: Latent Space Interpolation timing measurements in seconds for $n_{frames} = 3$.

[1] Overmono. 2023. So U Kno [Recorded by Overmono]. Good Lies. XL Recordings.

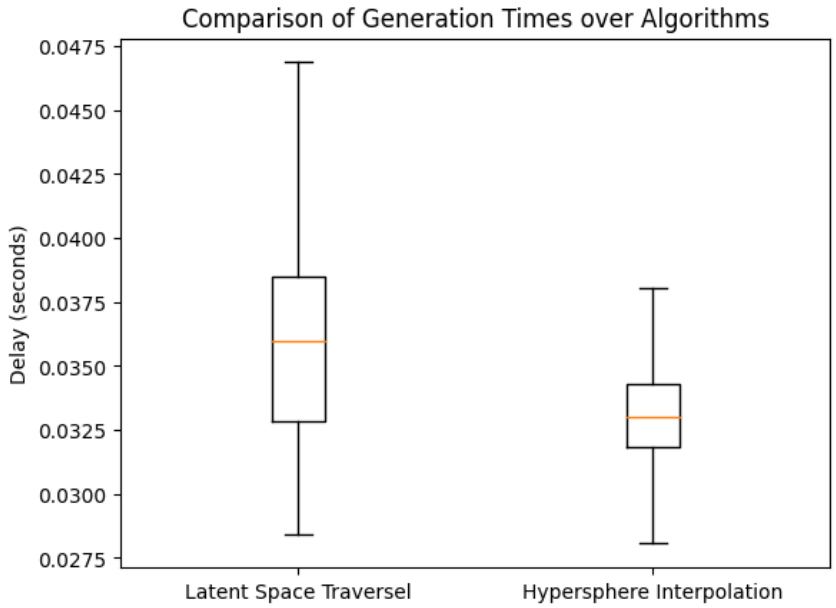


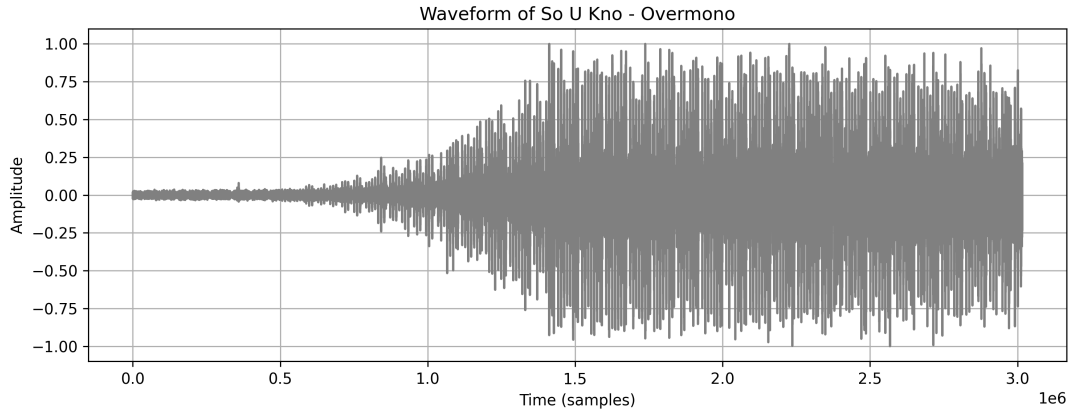
Figure 8.8: Generation times compared for the Latent Space Traversal algorithm and the Hypersphere Interpolation algorithm.

From Figure 8.8 it should be clear that the average frame rate lies higher and is more consistent, with less variability, for the Hypersphere Interpolation algorithm compared to the latent space traversal algorithm.

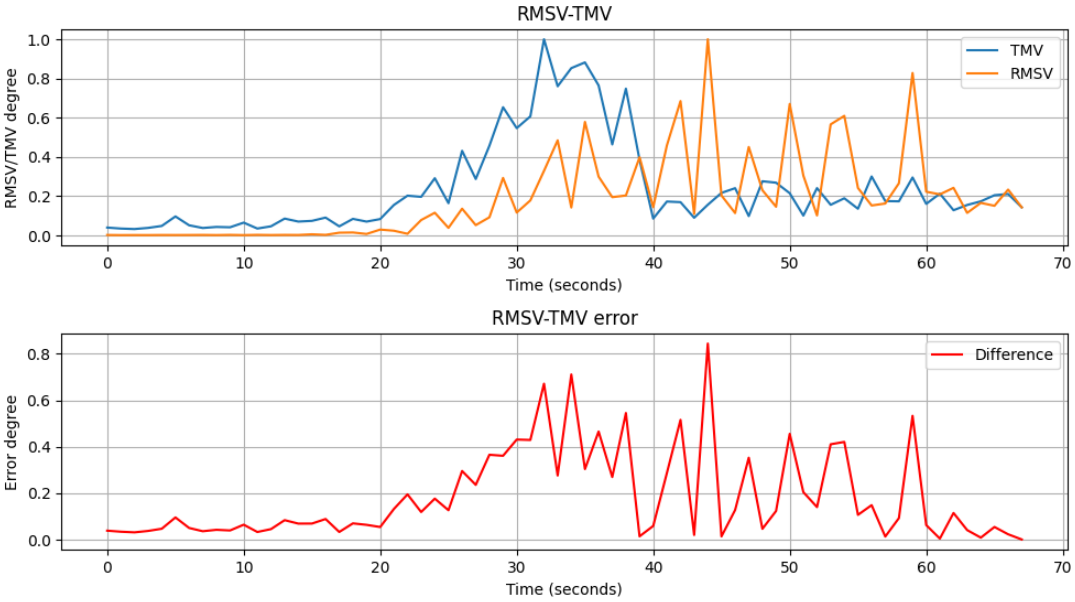
The mean generation time of the hypersphere interpolation algorithm with these parameters then takes on average around 33.64 ms, leading to a mean frame rate of around 30 FPS.

8.4.2 Reactiveness & Coherence

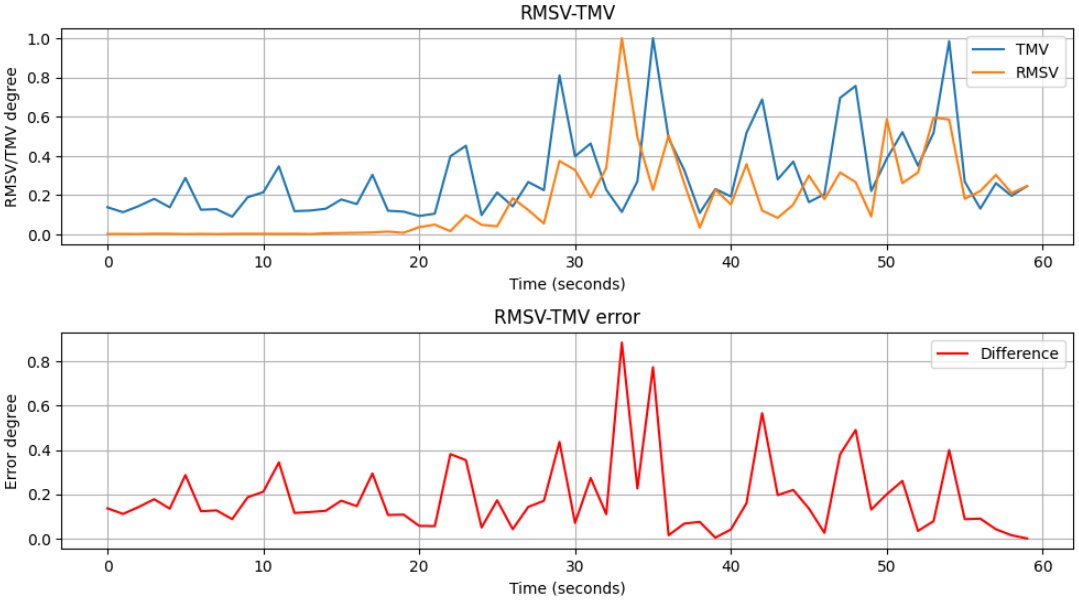
The RMSV-TMV plots and test fragment waveform for both algorithms are visualized in Figure 8.9. The mean and standard deviations of the RMSV-TMV error are given in Table 8.1.



(a) Waveform of test track 'So U Kno' by Overmono (67s).



(b) RMSV-TMV Latent Space Traversal.



(c) RMSV-TMV Hypersphere Interpolation.

Figure 8.9: Reactiveness and coherence tests of the Latent Space Traversal and Hypersphere Interpolation algorithms.

It should be clear that the TMV values for both algorithms follow the same trends as the RMSV values in the curves visualised in Figure 8.9, which indicates a decent level of coherence for both algorithms.

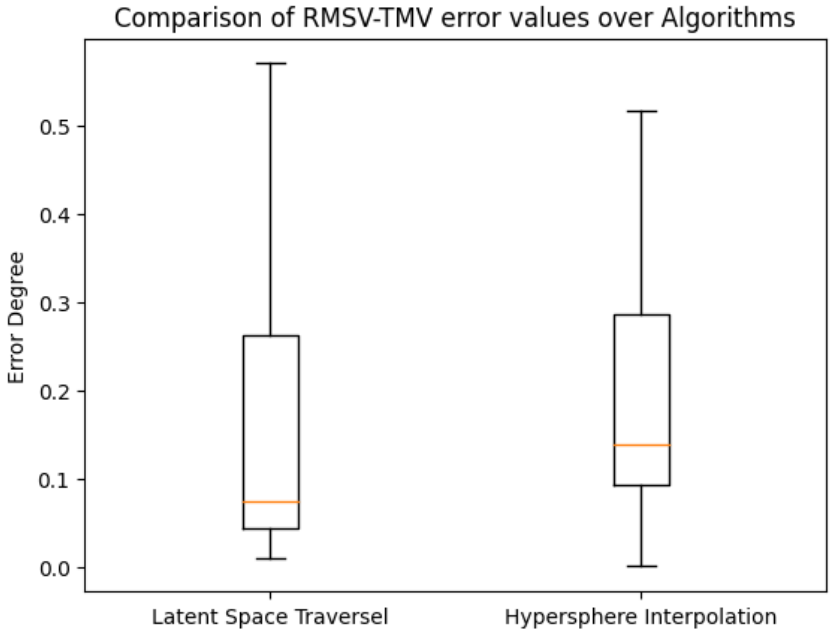


Figure 8.10: RMSV-TMV error values compared for the Latent Space Traversal algorithm and the Hypersphere Interpolation algorithm.

When comparing both algorithms, one notices that the Latent Space Traversal algorithm has a lower median RMSV-TMV error, but a higher interquartile distance, with incoherent parts in the high volume parts being masked by the almost perfectly coherent parts in the low-volume parts of the measured fragment.
Looking at the RMSV-TMV error plots thus provides more information than looking at the mean values alone.

Moreover, it should be clear that the RMSV-TMV plot of the Hypersphere Interpolation algorithm measures movement at the beginning of the recording, although no audio input is detected yet. The reason for this is the fact that the hypersphere algorithm is a looping algorithm, that interpolates on the surface of the n-sphere even when no stimulus is present.
This adds significant error to the measured RMSV-TMV error, visualised in Figure 8.10.

Although by definition, having moving visuals without an audio input does imply incoherence between the visualisation and the music, one could question how much of an impact this has on

user experience. It could for example be odd to look at stagnant visuals when silent parts occur in a musical performance.
In such cases, incoherent visualisation could still prove useful.

Both these observations indicate that although the RMSV-TMV plots and error values give a fair indication of the coherence of a visualisation system, it is not ideal for comparing the coherence of individual systems.
Although we'll keep using the RMSV-TMV error metric in the rest of this report, further research is necessary in order to perfect the metric. This is mentioned in Section 11.2

Algorithm	$\mathbb{E}$**[RMSV-TMVE]**
Latent Space Traversal	**0.187(0.21)**
Hypersphere Interpolation	0.194(0.16)

Table 8.1: Mean RMSV-TMV error of the Algorithms

8.5 Discussion

In this chapter, three different algorithms were presented that can be employed to create real-time audio reactive latent space interpolations. Table 8.2 gives a comparative overview of the objective metrics calculated for the latent space traversal algorithm in Section 8.2 and the hypersphere interpolation algorithm in Section 8.3.

Table 8.2: Performance Comparison of the Algorithms

Algorithm	T_g	$LSE_{\mathbf{min}}$	$LSE_{\mathbf{max}}$	**Frame Rate**	$\mathbb{E}$**[RMSV-TMVE]**
Latent Space Traversal	**31 ms**	36.5 ms	5.5 ms + 31 ms $\times$ n_{frames}	**32 FPS**	**0.187**
Hypersphere Interpolation	33.64 ms	**33.64 ms**	n_{frames} $\times$ 33.64 ms	30 FPS	0.194

While the latent space traversal algorithm slightly outperforms the hypersphere interpolation algorithm on all accounts except for the $\mathbf{LSE_{min}}$ value.
When it comes to frame rate, however, one should note that the reported frame rate is the theoretical optimal based on the mean generation time. As stated before, the Latent Space Traversal algorithm comes with inherent jitter because of the interpolation phase. The effective frame rate is thus never equal to this optimal. The $\mathbf{LSE_{min}}$ and $\mathbf{LSE_{max}}$ should be better representations of the apparent frame rate and the delay than the mean frame rate.
Both algorithms meet the latency constraints mentioned by Karras et al. and the ITU-R reported in Section 2.2.2 in terms of $\mathbf{LSE_{min}}$, they fail to meet the constraints set by the ATSC. Both frame rates are above the initial cinema standard of 24 FPS, but do not exceed the CFFT mentioned.

The RMSV-TMV error is low for both algorithms, implying both visualise the music in a coherent and reactive fashion. However, the Latent Space Traversal algorithm seems to slightly outperform the Hypersphere Interpolation algorithm. A reason for this could be the looping nature of the Hypersphere Interpolation algorithm that leads to omni-present motion, even when no audio input is observed.
The hypersphere algorithm takes away the problem of transient walks, making the system stable over long timeframes and thus making prolonged VJ sets possible.
On top of that, the hypersphere algorithm is recurrent, allowing the artist to represent repeating patterns in the music by matching the tempo with the interpolation period.
This chapter offers the first quantitive overview of current state-of-the-art audio reactive latent interpolation algorithms and mentions some key problems that could occur. It moreover proposes a novel algorithm that delivers promising results in terms of lip-sync error and RMSV-TMV error at the cost of a slightly lower frame rate. On top of that, the algorithm is more stable over the long-term, allowing for prolonged visual performances.

9 Creating visual narrative through Semantic Control

As stated in Section 2.2.1, a modern VJing system should allow the artist extensive control to influence the generation process in real-time. The reason for this is two-fold.
First, as described in earlier sections, none of the proposed approaches are completely error-prone: the latent space will for example always be partly entangled and onset detection is never completely accurate. A human in the loop could track the performance of the system and interact with it to mask these errors to the public.
On the other hand, a VJ set should actively react to the energy of an audience and the atmosphere the DJ is creating. Although the system described here succeeds in reacting to musical cues, it fails to recognize the needs of the audience itself. Moreover, the visualisations generated during live performance should have a form of semantic value: the image sequences need to tell something meaningful. The artist wants to tell a story.
Using the system as is, will not lead to a visual narrative. To tell a story, the artist needs to make active choices to steer the visualisation into semantically meaningful directions.

In this chapter, four different methods are proposed that make it possible for the VJ to have more control over the performance and semantic value of the set, while keeping system performance optimal.

9.1 Collecting the Data

It could seem trivial, but the first step in creating a visual narrative for an artist is by collecting the right data to reflect the message they want to tell.
Kraasch and Pasquier mention three different methods to collect the right data for training the synthesis model [20].

The first option they discuss is using already curated datasets and pretrained models. They emphasize the importance of taking copyright issues in mind and stress this approach could lead to less novel imagery.
Secondly, they propose to use public imagery created by artists under open licence. Although more labour-intensive since imagery needs to be manually selected, this approach could lead to novel creative visualisations.
Finally, they mention creating every image of the dataset yourself as an artist. Taking away any copyright concerns, at the cost of a labour- and creatively intensive process.

In this research project, an extra approach is proposed, which uses artificially generated imagery by making use of for example Stable Diffusion [66] or Midjourney [4]. This could provide for an effective closed-loop in future software, where the artist could create audio reactive visualisations solely based on text prompts. A setup like this was used at the Ghent Algorave performance, described in Appendix D.2.2.

9.2 Finding Meaningful Directions of Change

As explained in Section 5.1, a latent vector can be manipulated by adding a direction vector to its encoding. Both the latent space traversal algorithm and the hypersphere interpolation algorithm presented in Sections 8.1 and 8.3.1 are based on this property of the latent space, by gradually adding a direction vector on musical cues.
Choosing a meaningful latent direction to move in, however, is not a straightforward operation when working with a StyleGAN2 model. Since the StyleGAN2 latent space is not as disentangled, just changing one factor of the latent code should in general not lead to meaningful change of one factor of variation in the image.
There is no straightforward linear relation to be found between the latent code variation and the semantically meaningful variation of the generated image.
On top of this, the StyleGAN2 latent space has 512 dimensions, which makes finding a meaningful direction vector like searching for a needle in a haystack.
To solve this issue, several approaches have been proposed to automatize this process. The GANSpace algorithm [49] used by Kraasch and Pasquier in the Autolume project [20] is one of these approaches. By performing PCA on a set of sampled data, principal components of the StyleGAN2 $\mathcal{W}$-space can be discovered and added to vectors to create semantically meaningful change. Due to the layered structure of the StyleGAN2 architecture, it is possible to only add change to the $\mathcal{W}$ style vectors injected in certain layers, adding even more control of the generated image.

In this project, an alternative algorithm to find these meaningful directions of change is used: The Semantic Factorization (SeFA) algorithm proposed by Shen and Zhou [67], since it is reported to

outperform the GANSpace algorithm in terms of FID score and user surveys. It is an unsupervised approach, discovering meaningful factors of variation only looking at the pre-trained weights of a model generator.

Shen and Zhou state that the generator in GANs can be viewed as a multistep function that gradually projects the latent space to the image space. They proceed to state that each individual step could be formalized as an affine transformation, so that the output after the first projection step y' is as formulated in Equation 9.1.

$$\begin{aligned} \mathbf{y}' \triangleq G_1(\mathbf{z}') &= G_1(\mathbf{z} + \alpha\mathbf{n}) \\ &= \mathbf{A}\mathbf{z} + \mathbf{b} + \alpha\mathbf{A}\mathbf{n} = \mathbf{y} + \alpha\mathbf{A}\mathbf{n} \end{aligned} \tag{9.1}$$

They conclude that the weight parameter $\mathbf{A}$ should contain the essential knowledge of the image variation, and it should thus be possible to find the important latent directions through decomposition of $\mathbf{A}$. Consequently, they stated that the problem of semantic factorization is equivalent to the optimization problem formulated in the following equation

$$\mathbf{n}^* = \underset{\left\{\mathbf{n}\in\mathbb{R}^d : \mathbf{n}^T\mathbf{n}=1\right\}}{\arg\max} \|\mathbf{A}\mathbf{n}\|_2^2, \tag{9.2}$$

where $\|\cdot\|_2$ denotes the l_2 norm. or

$$\mathbf{N}^* = \underset{\left\{\mathbf{N}\in\mathbb{R}^{d\times k} : \mathbf{n}_i^T\mathbf{n}_i=1 \forall i=1,\cdots,k\right\}}{\arg\max} \sum_{i=1}^{k} \|\mathbf{A}\mathbf{n}_i\|_2^2, \tag{9.3}$$

where $\mathbf{N} = [\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_k]$ correspond to the top- k semantics.

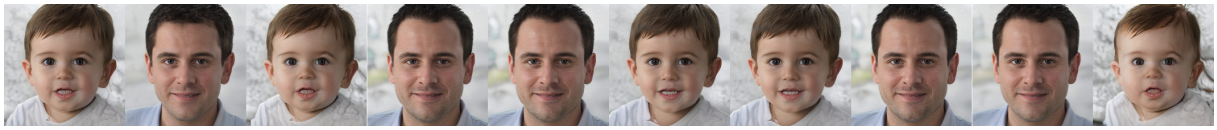
To solve the optimization Lagrange multipliers are introduced leading to

$$\begin{aligned} \mathbf{N}^* &= \underset{\mathbf{N}\in\mathbb{R}^{d\times k}}{\arg\max} \sum_{i=1}^{k} \|\mathbf{A}\mathbf{n}_i\|_2^2 - \sum_{i=1}^{k} \lambda_i \left(\mathbf{n}_i^T\mathbf{n}_i - 1\right) \\ &= \underset{\mathbf{N}\in\mathbb{R}^{d\times k}}{\arg\max} \sum_{i=1}^{k} \left(\mathbf{n}_i^T\mathbf{A}^T\mathbf{A}\mathbf{n}_i - \lambda_i\mathbf{n}_i^T\mathbf{n}_i + \lambda_i\right) \end{aligned} \tag{9.4}$$

By taking the partial derivative on each $\mathbf{n}_i$, we have

$$2\mathbf{A}^T\mathbf{A}\mathbf{n}_i - 2\lambda_i\mathbf{n}_i = 0 \tag{9.5}$$

All possible solutions to Equation 9.5 should be the eigenvectors of the matrix $\mathbf{A}^T\mathbf{A}$. To get the maximum objective value and make the columns of $\mathbf{N}$ are chosen as the eigenvectors of $\mathbf{A}^T\mathbf{A}$ associated with the k largest eigenvalues. The SeFa algorithm returns a list of 512 eigenvectors that could be viewed as directions of change, sorted from most impactful to least impactful. This way, the artist gets the possibility to choose from 512 pre-calculated direction vectors that are guaranteed to be impactful and semantically meaningful. A couple of eigenvectors and their effects on the FFHQ 512×512 dataset are visualised in Figure 9.1. Since these vectors are pre-calculated, they add no overhead on computation time.



(a) Eigenvector 1: age.

(b) Eigenvector 5: eyecolor.

(c) Eigenvector 2: gender.

(d) Eigenvector 4: glasses.

Figure 9.1: 10 equidistant frames of 4 different visualisations where volume was mapped on different eigenvectors/directions in the FFHQ 512 × 512 latent space.

9.3 Latent Space Projection

As mentioned in Section 8.2 the artist has the option to pre-save certain latent codes and interpolate in between those. Again, the question then rises how the artist can find these meaningful latent vectors in the multidimensional latent space.
A possible solution lies in the method of latent space projection, where an existing real image gets projected into the latent space of the model.
When using StyleGAN2 models, no straight-forward method exists to invert the inference process and create a latent code from an image. To achieve the goal of projecting in image in the StyleGAN latent space, GAN inversion is necessary.
Creswell and Bharath proposed an optimization problem that realises this inversion process, described by Equation 9.6.

$$z^* = \min_z -\mathbb{E}_x \log[G(z)] \tag{9.6}$$

They go on to state that, provided that the computational graph for $G(z)$ is known, z^* can be calculated via gradient descent, taking the gradient of G with respect to z.

This process is described in detail in Algorithm 4

Algorithm 4 Algorithm for inferring $z^* \in \mathcal{Z}$, the latent representation for an image $x \in \mathbb{R}^{m \times n}$

procedure INFER(x)
  $z^* \sim P_{\mathcal{Z}}(Z)$ ▷ Initialize z by sampling the prior distribution
  while NOT converged **do**
    $L \leftarrow -(x \log G(z^*) + (1 - x) \log(1 - G(z^*)))$ ▷ Calculate the error
    $z^* \leftarrow z^* - \alpha \nabla L$ ▷ Apply gradient descent
  end while
  return z^*
end procedure

Gatys, Ecker, and Bethge observed that the learned filters of the VGG image classification model [53] are good feature extractors and proposed to use the covariance statistics of the extracted features to measure the high-level similarity between images perceptually [69]. These statics were formalized as the perceptual loss and prove to be good loss functions to perform gradient descent on during the inversion algorithm proposed in Algorithm 4.

The approach sketched above is the approach used in the StyleGAN2-ADA Pytorch implementation by Karras et al. employed in this project [38]. Some StyleGAN2 projections and their perceptual loss are visualised in Figure 9.2. This projection algorithm has some substantial computation overhead and should thus happen beforehand.

(a) PL = 0.110 (b) PL = 0.112 (c) PL = 0.001

Figure 9.2: Sample of images projected into the 128 × 128 StyleGAN latent space

9.4 MIDI-control

As mentioned in Section 2.2.1 the VJ should have the option to intervene in the visualisation process and change important parameters on the fly. In this research setup, the Behringer BCF2000 MIDI-controller was used, visualised in Figure 9.3.
Using the hypersphere algorithm linked to a StyleGAN2 synthesis network, the sliders were mapped to the different system parameters like centre, period, and radius. On top of that, the strength of the injected noise and degree of movement in the eigenvector direction can be adapted as well. Moreover, the artist has the option to save up to 10 pretrained models and could switch between them in real-time. When an interesting set of parameters is found, the VJ can save them to one out of 6 slots and recall them whenever necessary. A reset button frees all used slots again. Finally, the used eigenvector and resolution can be changed as well.

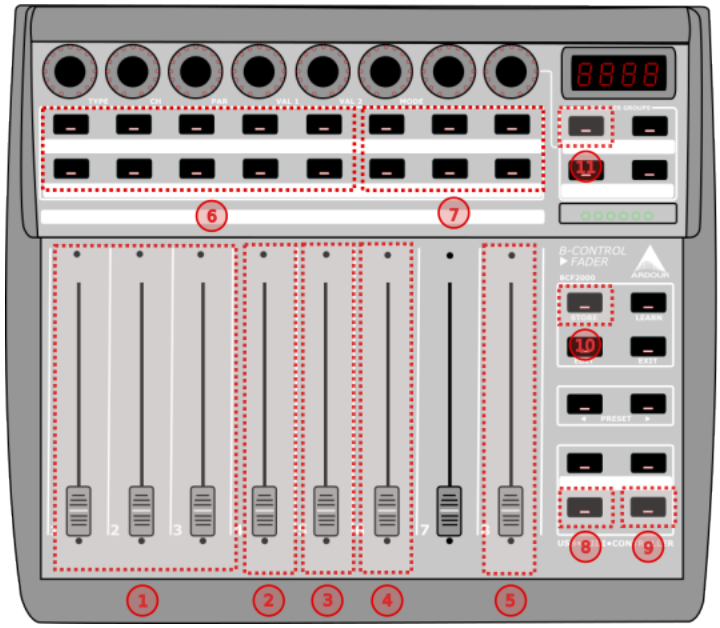


Figure 9.3: The physical mapping between the MIDI-controller and the system parameters.

10 β-VAE models as Synthesis Networks

As stated in Section 5.3.2, Kraasch and Pasquier only allowed users and artists to make interpolations based on a pretrained vanilla StyleGAN2 model [20]. For training, however, they used the StyleGAN2(-ADA) training scheme, in order to allow the visual artist to use smaller datasets. The same reasoning is followed in our research, in order to improve user-friendliness for VJs and artists it is desired to reduce training times and dataset sizes to their minimum, making the use of generative models for VJing more versatile and attractive.
In this chapter, the potential of a β-VAE as a synthesis network is researched.
From Section 4.2 it should be clear that both GAN and VAE models achieved state-of-the-art generation results when first released and are both considered to be high-quality generative architectures.
The choice of using VAEs is motivated by several of their attractive properties, but come with some disadvantages as well. Based on theoretical assumptions, a few researched questions are formulated that will be answered throughout this research.

10.1 Why VAEs? The hypotheses.

10.1.1 Lightweight Training Procedure

First of all, training a VAE model from scratch is known to be in general easier compared to the training of a GAN model. The reason for this lies in the innate unsupervised nature of training a VAE model as described in Section 4.2.3 compared to the dual model supervised training for GAN models described in Section 4.4, requiring precise synchronization. Apart from shorter and more stable training, this leads to the possibility of using smaller datasets to train a VAE from scratch compared to a GAN. Can we prove that using VAEs could potentially minimize the time and effort an artist needs to spend on training a model, while at the same time allowing him/her

to create models from smaller datasets?

10.1.2 Computation Time and Memory Consumption

VAE models are generally more lightweight than their StyleGAN counterparts, due to their simpler architecture. VAEs should be more efficient in both a computational sense and when it comes to CPU and GPU memory usage.
By design, inference from a VAE involves the sampling of the latent means and standard deviations as described in Section 4.2 and feeding those through the decoder, a neural network. Inference from a StyleGAN2 model involves sampling a random noise latent vector and feeding it through the generator network, which generally has a more complex architecture than the decoder network of a VAE. Our assumption is that by using a VAE model as the synthesis network, the frame rate (FPS) of live visualisation could be significantly raised and less computational resources in terms of both GPU and CPU memory should be needed.
This could open the door to better real-time signal processing and musical feature extraction, running multiple models at once for live mixing or more advanced projection mapping.

10.1.3 Latent Space Topology

The encoder module of a β-VAE maps an image to its latent code. This latent code consists of one latent vector of 128 dimensions: both representing the mean μ of the multivariate Gaussian latent distribution and the log variance $\log(\sigma^2)$ of the latent distribution.
It is thus important to note that the latent space of VAE is a probabilistic one. Images are not hard coded in the space itself, but probabilistic parameters are represented in the latent space. By sampling those parameters, an image can be generated. This inherently adds non-determinism to the interpolation process. Sampling the same μ and σ latent vectors, could lead to different generated images every time.
At the same time, the fact that the latent space is probabilistic, makes its topology more smooth. Images generated from relatively centred latent variables, result in images strongly represented in the training set, while images represented by latent vectors towards the edges represent more novel and surprising data. The assumption thus is that using a β-VAE synthesis network could lead to smoother interpolations than using StyleGAN2 and finer control over the novelty of the images generated.

10.1.4 Latent Space Entanglement

As described in Section 4.2.5 the concept of latent space entanglement has a profound impact on the coherence of latent interpolations. In a heavily entangled latent space, there is no linear relation between the length of a trajectory in the latent space and the variation in visual output. This is unwanted, since the trajectory in the latent space forms the mapping between musical features and visual ones in our approach to music visualisation.

VAEs and especially β-VAEs are known to have a less entangled latent space than styleGAN2 models. Our assumption is thus that musical cues could be mapped more accurately to the VAE latent space, than to the StyleGAN2 latent space, leading to better translation from music to visuals. On top of that, it should be easier to find meaningful directions in the β-VAE latent space than in the StyleGAN2 latent space, since in β-VAE each dimension in the latent space represents another factor of variation. This property could make it more intuitive for an artist to create meaningful visualisations.

10.1.5 Reconstruction Ability

In the training objective of VAEs, a strong emphasis is put on the reconstruction loss, in the ELBO Loss described in Equation 4.8. This leads to VAEs representing the training data very well while being worse at generating novel data after training.
A StyleGAN2 model should thus outperform a VAE model when it comes to generating images that differ from the training set and will generate a wider diversity of visualisations. However, increasing the value of β when training a β-VAE, diminishes that emphasis and puts more weight on the regularization parameter, leading to worse representation of the training data.

10.1.6 Visual Quality

All possible advantages of using VAEs aside, one needs to acknowledge that the visual quality of VAEs is in general worse than the visual quality of StyleGAN2. A common artefact in VAEs is a blur towards the edges of an image, an artefact being completely absent in images generated by StyleGAN2 models. StyleGAN2 models are in general also better at generating higher resolution images, while VAEs often struggle with this. The reason for this is that VAEs prioritize reconstruction over the creation of high-fidelity images.
By the same cause, the images generated by a VAE are in general less diverse than the ones generated by a StyleGAN2 model, with less attention to detail.
Our assumption is that a StyleGAN2 based system will outperform a VAE based system when it comes to overall visual quality and diversity. On the other hand, however, a VAE based system

Table 10.1: Training parameters and timing StyleGAN2-ADA and β-VAE.

Data set	Model	FID	#Epochs	kimg	Batch size	Learning rate	β	Training Time
CelebA [70]	β-VAE	116.33	130	-	64	1×10^{-4}	1×10^{-3}	9h 55m 25s
	StyleGAN2-ADA	**17.76**	-	600	16	1×10^{-8}	-	9h 48m 45s

should be more computationally efficient, leading to new applications in situations where higher frame rates are necessary.
Moreover, the aesthetic created by using a VAE synthesis network, although of lower objective quality, could work in the context of music visualisation and offer an extra tool to the generative VJ.

10.2 Quantitative Performance Evaluation

10.2.1 Training

In order to evaluate the performance of the VAE based system, both a β-VAE and StyleGAN2 model were trained on the same dataset. The used dataset is the CelebA dataset [70], containing 202,599 images of celebrity faces cropped to a resolution of 128×128. Both the StyleGAN2 model and the VAE model were trained to generate 128×128 images, since VAE models are notoriously bad at generating higher resolutions.
The exact CNN architecture of the encoder is visualised in Table B.1, the StyleGAN2 model was trained using the official StyleGAN2-ADA PyTorch Implementation [71] without changing anything to the vanilla generator and discriminator architectures presented there.
The StyleGAN2 model was trained from scratch, without resuming from a pretrained checkpoint.

The training parameters and training times for both models are visualised in Table 10.1, FID values are visualised in Figure 10.1. FID scores were calculated through the `pytorch-fid` GitHub repository [72]. Although the VAE model could potentially get better with a few extra training epochs, Figure 10.1 indicates that the FID score reachable with a β-VAE model is still a lot higher than the achievable score for GANs. In order to still be able to compare both, the two models used in the rest of this section were trained for about the same time with the β-VAE model trained for 130 epochs in 9 hours, 55 minutes and 25 seconds and the StyleGAN2 model for 600 kimg and 9 hours 48 minutes and 45 seconds. Both models were trained on the Nvidia Geforce RTX 3070 Ti Laptop GPU.

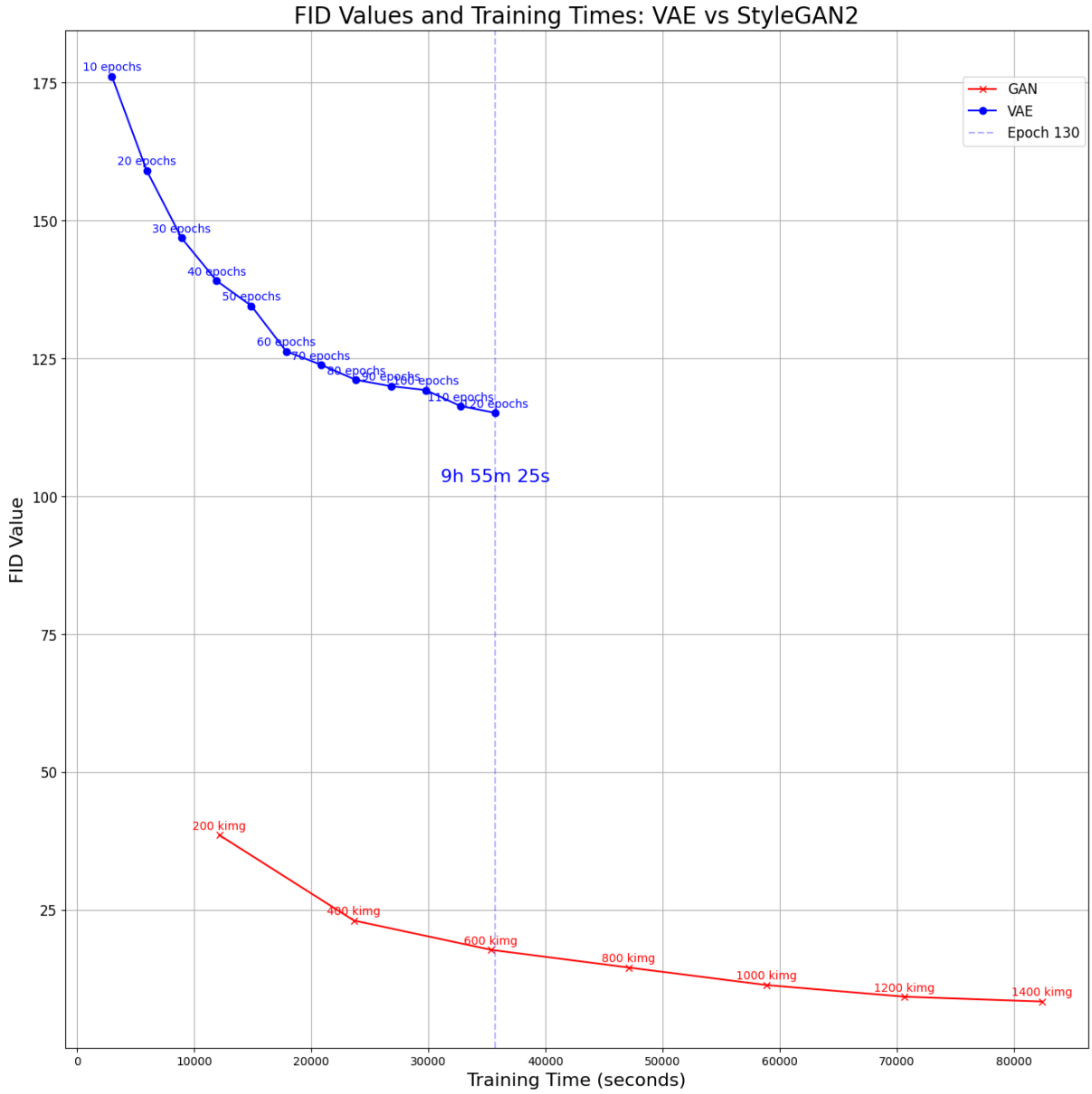
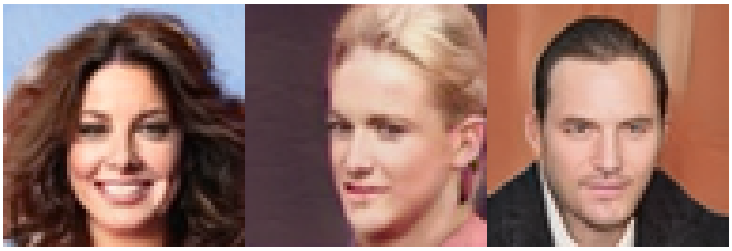


Figure 10.1: VAE and GAN FID scores as a function of time (seconds).

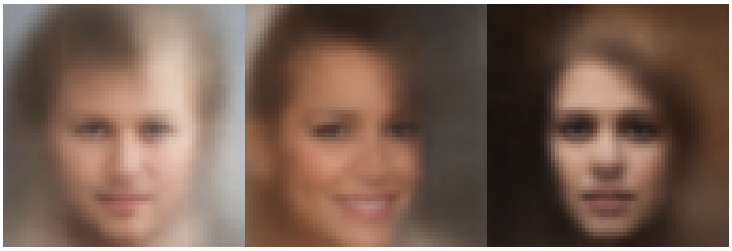
10.2.2 Visual Quality

From Table 10.1 and Figure 10.1 it should be clear that the StyleGAN2 model outperforms the β-VAE model when it comes to FID scores, which lies within expectation. As stated before, VAEs are known to lead to less qualitative and diverse results than GAN-based models.

From visual inspection, it is clear that the images generated by the VAE are less detailed and more blurry. The blur around the edges, typical for VAE based models, is especially visible.



(a) 3 StyleGAN2 128 $\times$ 128 images.



(b) 3 β-VAE 128 $\times$ 128 images.

Figure 10.2: Inferences from StyleGAN2 and β-VAE compared.

10.2.3 Entanglement

The entanglement of the StyleGAN2 $\mathcal{Z}$-latent space and $\mathcal{W}$-latent space were calculated, as well as the entanglement of the β-VAE latent space for different values of β in terms of the PPL. The PPL for the StyleGAN2 latent spaces was calculated making use of the `perceptual_path_length.py` contained in the StyleGAN2-ada-Pytorch repository and was adapted to function with VAE models as well. The results are visualised in Table 10.2.

Model	**GAN**		**VAE**		
	$\mathcal{Z}$	$\mathcal{W}$	$\beta = 0.001$	$\beta = 10$	$\beta = 100$
PPL	18.95	14.32	4.29	4.06	**3.82**

Table 10.2: Comparison of PPL Scores for GAN and VAE

The PPL measurements lie within the line of expectation. β-VAE models are overall less entangled than StyleGAN2 models, with the $\mathcal{Z}$-latent space being more entangled than the $\mathcal{W}$-latent space. The β-parameter has a clear impact on the entanglement as well, with higher values leading to lower PPL scores and vice versa.

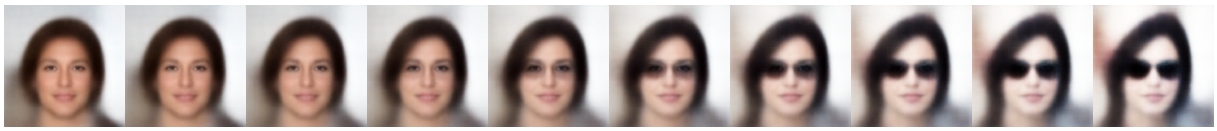
Visual inspection was performed on the interpolations as well. It became clear that some dimensions of the VAE latent vector encoded isolated features, through adapting one dimension at a time and performing linear interpolation between the end points. Figure 10.3 shows how gradually increasing the value of the 7th dimension of the latent vector leads to a shift in azimuth from left to right.

(a) Degree +10: azimuth shifting to the right.

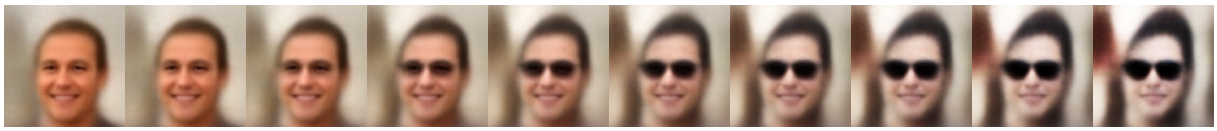
(b) Degree -10: azimuth shifting to the left.

Figure 10.3: Effect of adding a factor of -10/+10 to the 7th dimension of the $\beta = 100$-VAE latent space in 10 interpolation steps.

Remarkably, no matter what latent code is adapted in this way, the semantic effect is always an azimuth shift. Showing that the ($\beta = 100$)-VAE learned to encode azimuth in the 7th dimension of the latent code.



(a) Seed 1, Dimension 42, Degree +10: glasses added.



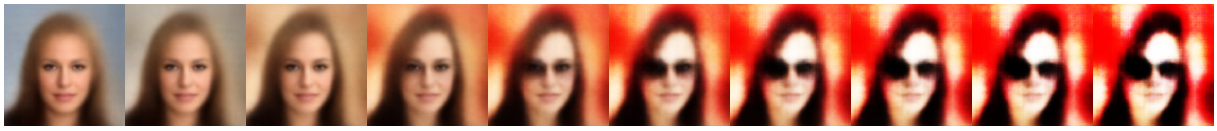
(b) Seed 2, Dimension 42, Degree +10: glasses added.

Figure 10.4: Two different latent vectors altered in the same latent dimension 42 lead to similar manipulations semantically.

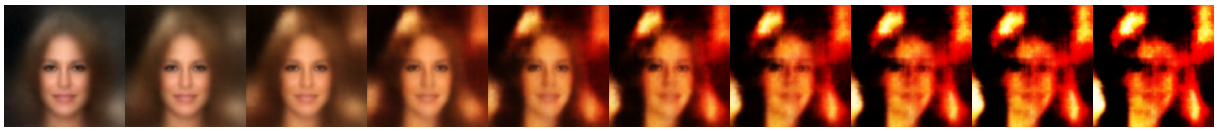
Other examples of semantics encoded in a single latent dimension are visualised in Figure 10.15

Figure 10.5a shows the effect of changing multiple of these dimensions at once, combining two semantic manipulations, and showing the disentangled nature of a β-VAE trained with high β. When comparing the result with the interpolation generated by the VAE trained with low β in Figure 10.5b, one can see that the result is not as good. The features are more entangled when trained with low β.

The fact that the β-VAE latent space is disentangled not only makes interpolations more smooth, it allows the artist intuitive control over image semantics. Finding meaningful directions in the StyleGAN2 latent space appeared not to be self-evident, as described in Section 9.2. In β-VAEs however, the algorithm for finding interesting directions in the latent space is given by design. By just scaling one latent dimension of a latent code, a semantically meaningful reaction to the music



(a) $\beta = 100$, Dimension 42 & 92, Degree +25 & +10: glasses added and background colour changed.



(b) $\beta = 0.001$, Dimension 42 & 92, Degree +25 & +10: glasses added and background colour changed.

Figure 10.5: Manipulating multiple latent dimensions at once leads to more disentangled results for $\beta = 100$ & for $\beta = 0.001$.

can already be delivered. By manipulating multiple dimensions at once, the semantic effects of these manipulations are combined as well. Finding semantic directions and manipulations is thus easier in β-VAE models than StyleGAN2 models, where specific algorithms are needed to find these semantically meaningful directions of change.

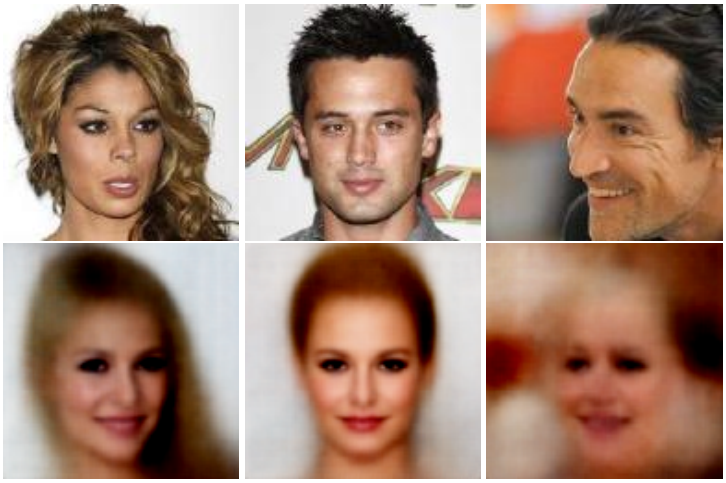
10.2.4 Projection

In much the same way as it gives an intuitive method to find semantically relevant directions in the latent space, the β-VAE architecture allows for image projection in that latent space by design. The reason for this being its encoder-decoder architecture, where the training procedure involves training both modules at the same time, as explained in Section 4.2. Instead of throwing away the encoder part of the pipeline after training, it could be used to project images into the VAE latent space without the need for any other algorithms.
Moreover, the method is incredibly fast, since projection now only involves a forward pass through a pretrained encoder and decoder. With inference times remarkably low for β-VAE models, as will be explained in Section 10.2.5.

Some results of employing this method are visualised in Figure 10.6. The perceptual loss as presented by Zhang et al. was used to score the success of the projection [52], through the `lpips` Python package [1].

From both visual inspection and a look at the perceptual losses, it should become clear that altough the projection process is easier and faster, the resulting images are less close in resemblance to the original images than the projected images presented in Figure 9.2.

[1] GitHub: `https://github.com/richzhang/PerceptualSimilarity#c-about-the-metric`



(a) PL = 0.420 (b) PL = 0.384 (c) PL = 0.540

Figure 10.6: Sample of images projected into the 128 × 128 β-VAE latent space

10.2.5 Latency, Frame Rate & Computational Complexity

When it comes to latency, a VAE-based system clearly outperforms GAN-based systems. Figure 10.7 shows how the mean generation time for the 128 × 128 StyleGAN2 model is a factor of 10 bigger than the mean generation time for the β-VAE model. For clarity, Figure 10.8 shows the generation times on separate scales. The StyleGAN2 model has a frame generation time of 13.0 ms, while the β-VAE model generation process only takes 1.26 ms on average. Both models employ the hypersphere algorithm from Section 8.3.1.

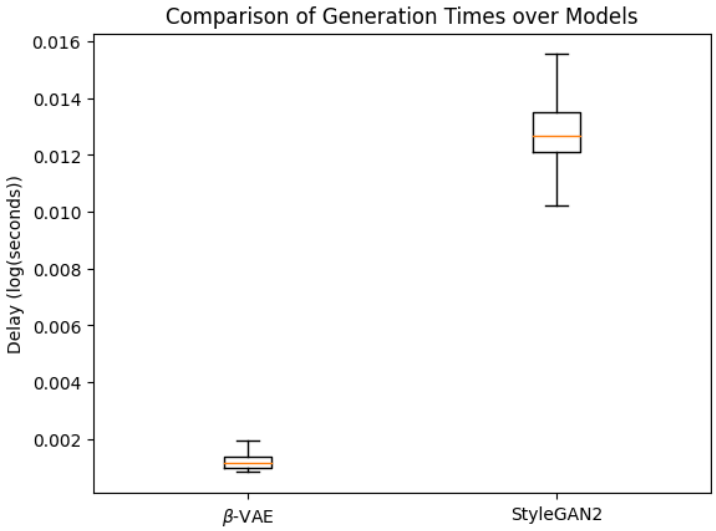
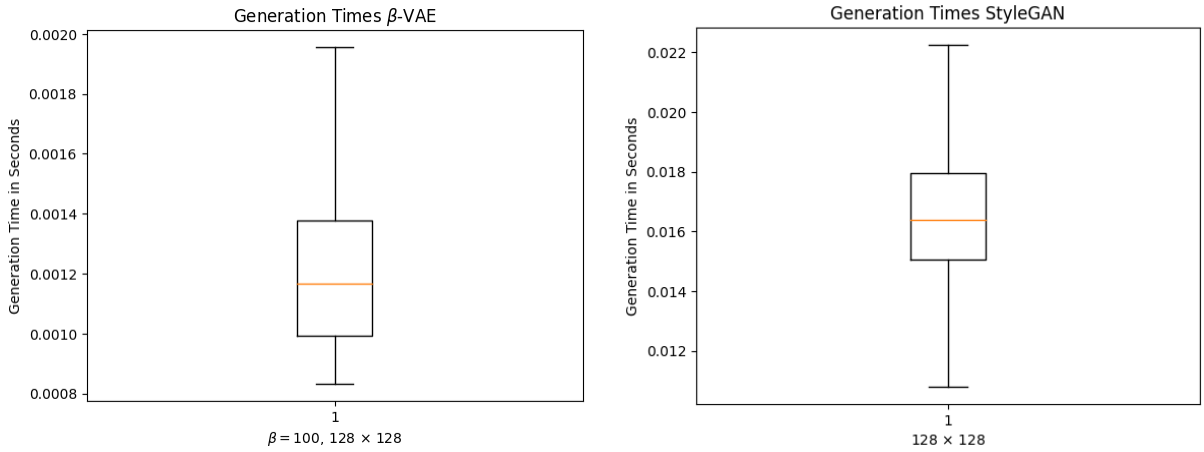


Figure 10.7: Comparison of the 128 × 128 β-VAE and StyleGAN2 generation times.

This allows the VAE-based system to run at average speeds up to 794 FPS, far above the



(a) β-VAE. (b) StyleGAN2.

Figure 10.8: Comparison of the 128 × 128 StyleGAN2 and β-VAE generation times.

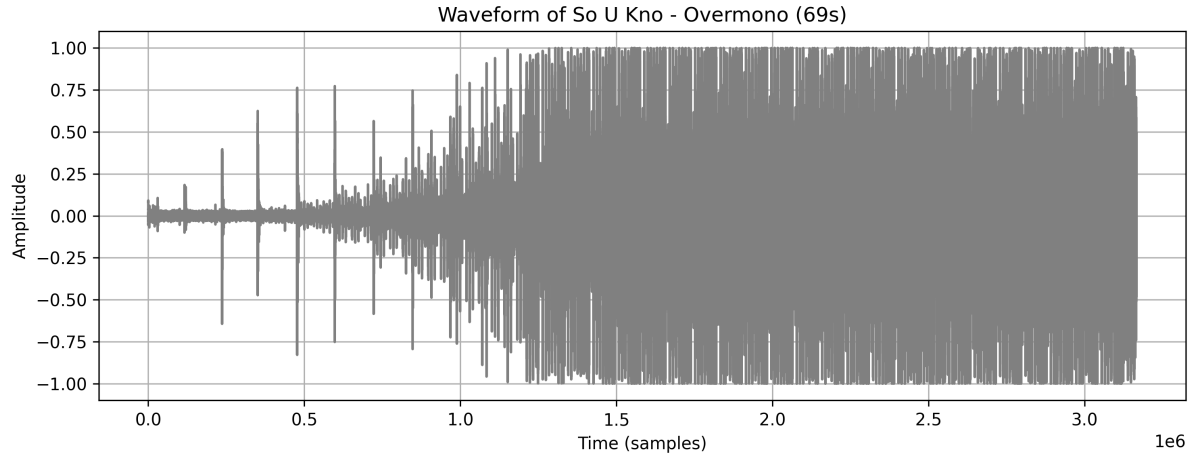
lip-sync-error and CFFT constraints specified in Section 2.2.2. This gives the artist the freedom to run the systems at any frame rate they deem necessary. The frame rate of the 128 × 128 StyleGAN2 model, however, reaches an average of 77 FPS.

10.2.6 Memory Usage

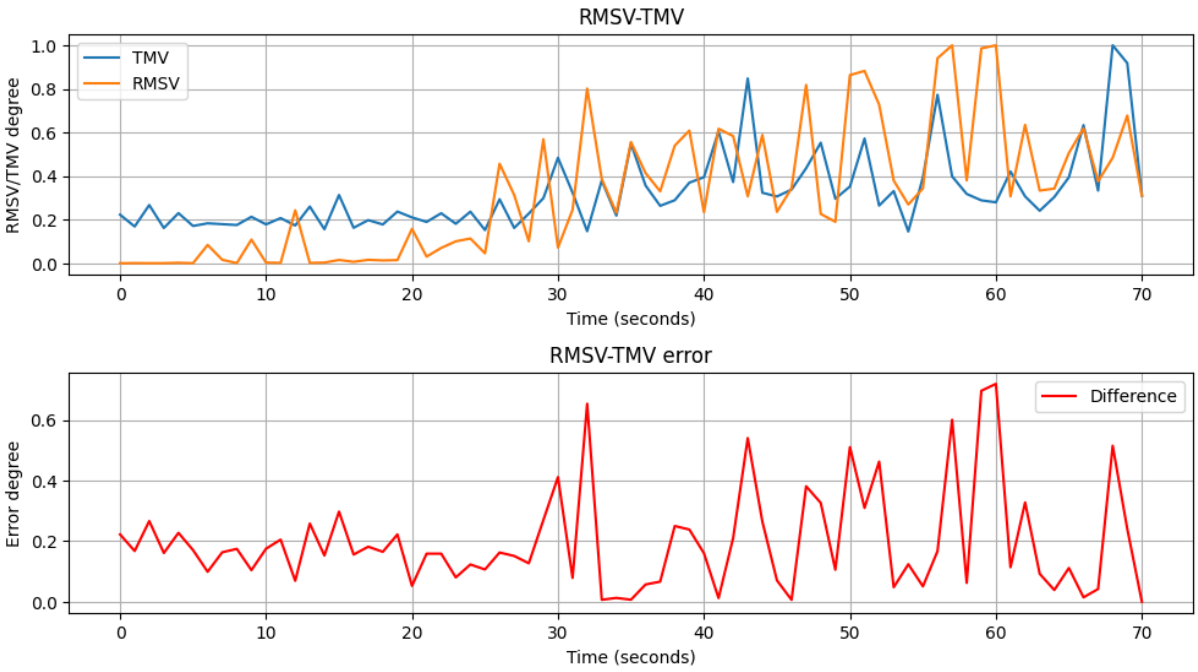
When it comes to GPU memory usage, empirical measurements led to unexpected results. When using the 128 × 128 StyleGAN2 model, the GPU utilization lies at 91.35 % on average. When on the other hand, a VAE model is used, that utilization lies at 95.28 %.
These results do not lie in line with the theoretical prognosis made in the introduction of this Chapter, nor does it explain the very fast inference times measured in the previous section.
A possible reason could be that the code as currently implemented always runs at maximum frame rates, only slowing down by inserting `sleep()` statements when a certain slower frame rate is desired.
A deeper look into the dynamics of the GPU processes is necessary to fully explain this observation.

10.2.7 Reactiveness & Coherence

In the more disentangled latent space of β-VAE, one would expect more coherent visualisations. This was tested by running both the VAE model and the GAN model at an average frame rate of 58 FPS and calculating the RMSV-TMV error. Both models used the hypersphere interpolation algorithm presented in Section 8.3.1, with the BPM matched to 130 BPM, the BPM of the test song 'So U No' by Overmono and the c parameter equal to 50.



(a) Waveform of test track 'So U Kno' by Overmono (69s).



(b) RMSV-TMV plots.

Figure 10.9: Reactiveness and coherence test of the hypersphere interpolation algorithm in the β-VAE latent space.

The RMSV-TMV error, visualised in Figure 10.10 measures on average 0.23 for the 128 $\times$ 128 β-VAE, while the mean for the 128 $\times$ 128 StyleGAN2 lies at 0.17.

From comparing the RMSV-TMV plots and error values, it should be clear that the StyleGAN2 model still outperforms the β-VAE based model when it comes to coherence. However, both perform quite well, with errors staying below 0.1.

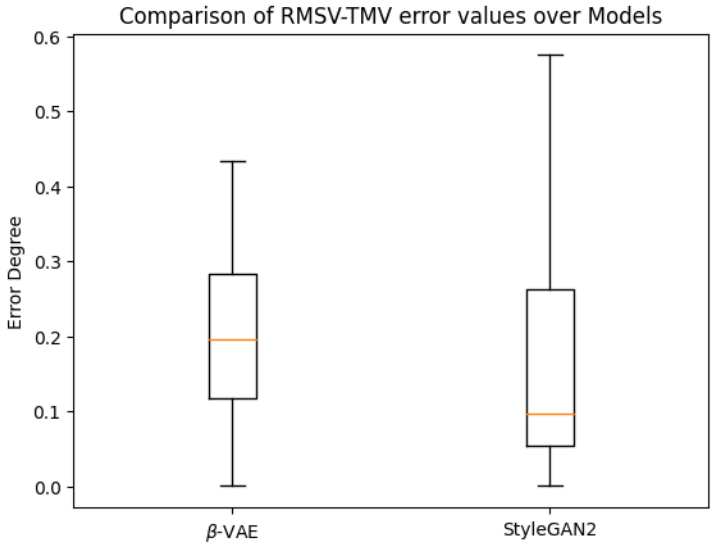


Figure 10.10: Comparison of the 128 × 128 β-VAE and StyleGAN2 RMSV-TMV error.

Model	$\mathbb{E}$**[RMSV-TMVE]**
β-VAE	**0.23 (0.16)**
StyleGAN2	0.17 (0.18)

Table 10.3: Mean RMSV-TMV error of the Models

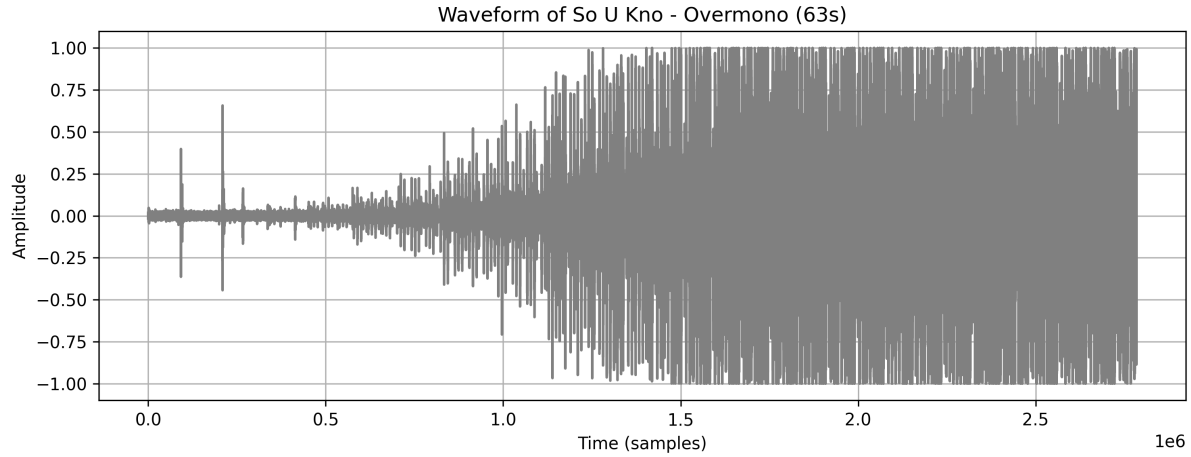
10.3 Discussion

In this chapter, the potential of β-VAE models was researched as synthesis models for the real-time audiovisual system. A comparative overview between the performance of StyleGAN2 and β-VAE is given in Table 10.4.

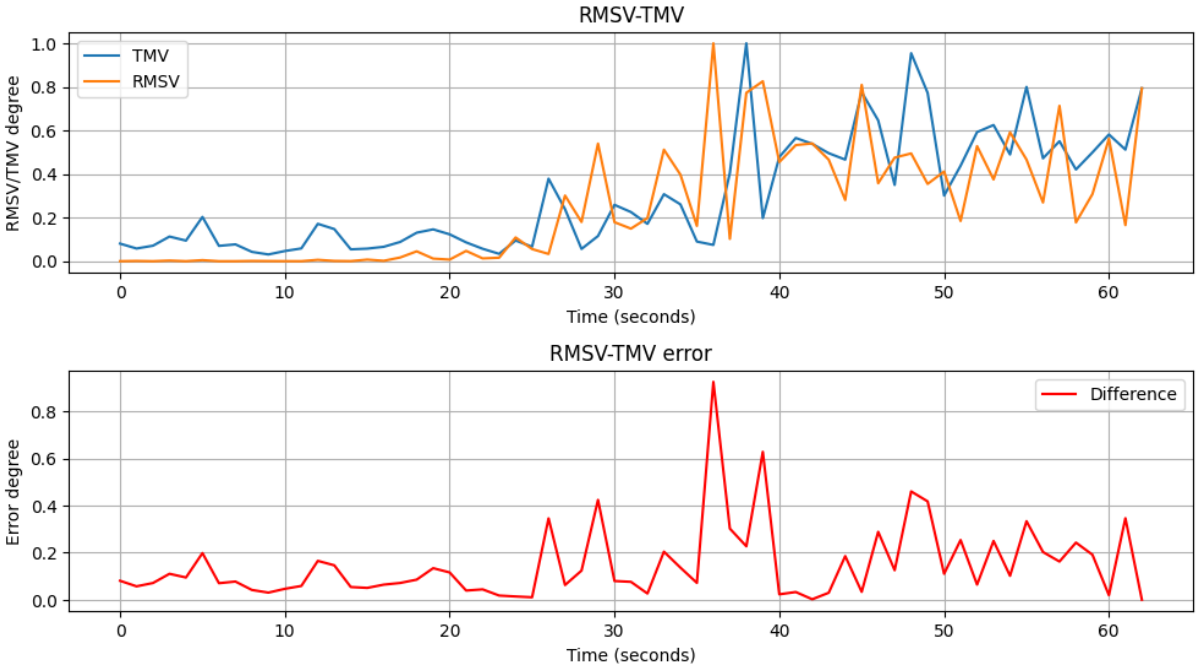
Model	**FID**	**PPL$_{\mathbf{min}}$**	**Frame Rate**	**GPU Util.**	**RMSV-TMV E**
StyleGAN2	**17.76**	14.32	77 FPS	**91.35** %	**0.17**
β-VAE	116.33	**3.82**	**794 FPS**	95.28 %	0.23

Table 10.4: Performance Comparison of the Models.

When comparing the performance results of both models, it is clear that the VAE model is promising as a synthesis network and worth further research in the context of real-time audio visualisation.
In terms of training time, a β-VAE model seems to need less time to be trained to its achievable performance in terms of FID than a StyleGAN2 model, but that FID score lies significantly



(a) Waveform of test track 'So U Kno' by Overmono (69s).



(b) RMSV-TMV plots.

Figure 10.11: Reactiveness and coherence test of the hypersphere interpolation algorithm in the 128×128 StyleGAN2 latent space.

higher for the β-VAE model than for the StyleGAN2 model.
This lies in line with visual inspection, upon which it quickly becomes clear that the images generated by the VAE model are of lower visual quality and less diverse, with blurring effects as the most prominent artefacts. Moreover, β-VAE models fail to perform at high resolutions.
On the other hand, lower PPL values are measured for β-VAE models, with the lowest scores reported for high β values.

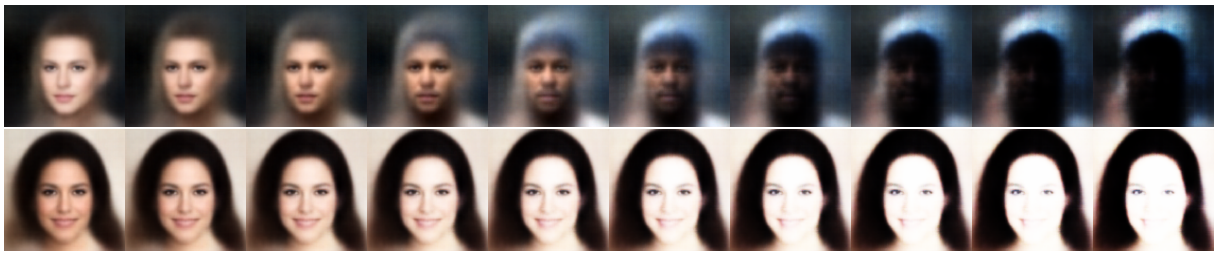


Figure 10.12: Dimension 12, Degree= +/-10: skincolor.

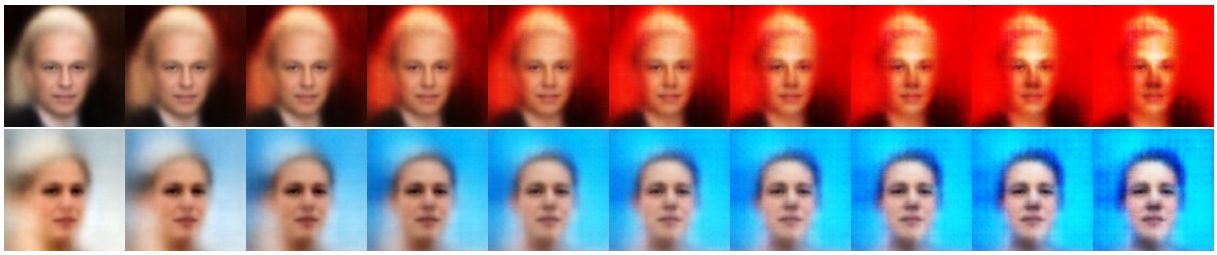


Figure 10.13: Dimension 92, Degree= +/-10: background color.

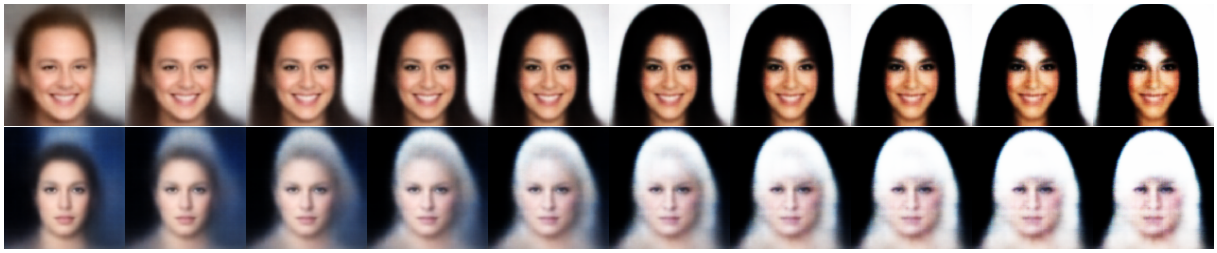


Figure 10.14: Dimension 81, Degree=+/-10: hair colour.

Figure 10.15: Semantically meaningful features encoded in a single dimension of the ($\beta = 100$)-VAE latent space

This tells that the VAE latent space is less entangled than the StyleGAN2 latent space, which is confirmed through visual inspection of the latent space interpolations. The disentangled nature of β-VAEs makes for easier control of semantically relevant features, since by design the β-VAE model forces single factors of variation upon single latent space dimensions.
Simple cycles through all dimensions before a VJing set, allow the artist to make the music change latent vectors in the latent space directions of most interesting visual change. Moreover, if the model is trained with a high β, these semantic manipulations become additive, making combinations of semantic change possible by changing multiple dimensions at the same time.
On top of that, the β-VAE inference process is faster by more than a factor 10 compared to the StyleGAN2 inference process, making frame rates of up to 794 FPS possible.
This creates the opportunity for more extensive signal processing, running visualisations over communication networks, running live visualisations on less powerful GPUs and even running multiple models at the same time, allowing for smooth transitions between models trained on different datasets.
This fast inference time, also comes in handy when it comes to image projection. Whereas StyleGAN2 architectures need dedicated architectures and long projection times to map an image into its latent space, VAE based models can project imagery into the latent space by design, through their encoder-decoder architecture.
However, these fast projection times come at the cost of lower precision and quality of the image projections themselves.
Although inference times are remarkably low, the GPU utilization seems to increase instead of decrease when trading a StyleGAN2 synthesis network for a β-VAE one.
The reasons for this should be researched further.
When it comes to coherence, the RMSV-TMV error measures slightly better coherence between the music and the visualisations when a StyleGAN2 synthesis model is used, which could have reasons that need to be researched further.
On a more subjective level, using β-VAE models for audio visualisation delivers a unique aesthetic, even though resolutions are low. The artist could make the choice to use this pixelated effect as a visual style, with the gain of having more intuitive live control over the semantic changes and the ability to operate at much higher frame rates, lower the lip sync error and increasing the reactiveness of the system.

11 Conclusion, Future Work & Impact

11.1 Conclusion

To our current knowledge, this project delivered the first in-depth performance evaluation of current state-of-the-art techniques in real-time audio visualisation by analysing the techniques used in the Autolume [20] system architecture.
This was done by isolating three modules of the architecture: *The Audio-Handler*, *The Mapping Module* and the *The Synthesis Module*, reverse-engineering their functionality and quantitively evaluating its performance.

In order to do this, a new metric was designed to measure the reactiveness and coherence of the visualisation system: the RMSV-TMV error, described in Section 6.2.4. This metric could prove to be a useful way to measure the quality of generative audiovisual systems in further research.

The Audio Module implemented real-time musical feature extraction. More specifically, the audio module extracted onset and RMS values of audio chunks in real-time and relayed those values to the Mapping Module. The quality of the onset detection was evaluated visually and was reported to be imperfect, leading to the parallel implementation of OSC musical feature communication.

The Mapping Module was analysed through reverse-engineering two of the algorithms implemented by Kraasch and Pasquier: The Latent Space Traversal and the Latent Space Interpolation algorithms in Sections 8.1 and 8.2, respectively. Since the techniques used by both are very similar, only the Latent Space Traversal algorithm was thoroughly evaluated for performance. Next to measuring frame rate and coherence, visual inspection of the system lead to the conclusion that the random nature of the Latent Space Traversal algorithm lead to unstable long-term image generation and eventually stagnation. To our knowledge, this stability flaw has not been reported in literature before.

Next to the two existing algorithms, a new algorithm was proposed and implemented: the Hypersphere Interpolation algorithm in Section 8.3. This algorithm takes away the inherent randomness of the Latent Space Traversal algorithm and thus its instability over the long-term, while keeping latency and frame rate in line with the Latent Space Traversal algorithm. The new algorithm is even more performant when it comes to coherence and reactiveness, while allowing the artist to introduce a notion of recurrence in their performance.

The Synthesis Module implemented by [20] only allowed for pretrained StyleGAN2 architectures to be used for inference and frame generation.
In this research project, the potential of β-VAEs as synthesis networks for real-time audio visualisations was analysed. By first formulating hypotheses based on theoretical assumption, questions were formulated that were answered through empirical measurement.
At the cost of visual quality and resolution, β-based systems deliver promising results when it comes to frame rate and latent space entanglement. Having the possibility to run at any chosen frame rate and offering intuitive control over semantic features through its disentangled latent space.
Offering the option to work with β-VAE models extends the creative toolkit of the artist.

A final advantage of β-VAE model is that they offer projection methods and semantics discovery by design, whereas for StyleGAN2 models, both of these mechanisms had to be implemented separately. The closed-form factorization approach to semantics discovery has been implemented for the first time in the context of real-time audio visualisation and gradient descent with perceptual loss have been used for image projection purposes. MIDI-control was added to the created software in order to give the artist the possibility to interact with the system in real-time, in an intuitive and user-friendly way.

11.2 Future Work

The work presented in this research project answered some questions, but made as many rise up. It can be improved upon and expanded in multiple directions, of which some were selected and presented in this section, organised per chapter.

The Metrics

- *RMSV-TMV*: The RMSV-TMV error metric designed in this project to measure the coherence degree of an audio visualisation system is imperfect. Although it gives a broad idea on how coherent the music and visuals are, comparing two error values with one another to decide upon which system is more coherent isn't as accurate as wanted.
  The fact that the values are normalized is part of the problem.

Even though the RMSV normalized peak lies at the same time instant as the TMV normalized peak, it does not mean that the degrees of change lie in the same order of magnitude. The RMSV-TMV only tells if the RMSV and TMV peaks recorded over a certain period of time lie at the same place, not if the two are in the same order of magnitude. A system could for example have high degrees of visual change upon a small musical change, if that small musical change is the maximum musical change recorded over the measured time frame.
Moreover, as mentioned in Section 8.4, although the RMSV-TMV plots give a fair indication of (in)coherence between the music and its visualisation, this does not necessarily reflect how an audience perceives this (in)coherence. More research is necessary in order to quantify how close the coherence measurements through the RMSV-TMV metric lie to human judgement. Implementing Dynamic Time Warping (DWT), cross-correlation between RMSV and TMV values and perceptual models all seem reasonable approaches for future fine-tuning of the metric.

- *User Survey*: As mentioned before, the quality of an audio visualisation system is inherently subjective. In order to still formalize some concepts considering aesthetics and user experience, user surveys could prove to be helpful in future research.

Feature Extraction

- *BPM Extraction*: In the project as is, real-time BPM extraction is not yet implemented. The BPM value should be hardcoded by the artist before a performance, which of course is not practical when performances are of longer durations and BPM changes occur during the performance. Using the `madmom.features.tempo` [1] method, included in the MadMom [61] Python package, should make implementing real-time BPM tracking feasible.

- *Harmonic/Percussive Split*: Brouwer split the audio track he used for offline audio visualisation in four separate streams, based on the frequency content they contained. This way he had the possibility to map different system different parameters to different parts of the frequency spectrum.
By making use of the work by Driedger, Müller, and Disch on harmonic-percussive splits [73], he for example made the percussive onsets drive the noise injection, while the vocal RMS scaled the direction latent vector.
Kraasch and Pasquier dropped this feature in their work because of the computational overhead it added to the system. In future work, it seems useful to implement this feature again, especially seen the minimal computational overhead β-VAE inference incurs.

- *Feature Semantics*: The features extracted from the audio signal in this work are low-level ones. They represent properties of the audio signal itself, but only give minimal information

[1] Documentation: `https://madmom.readthedocs.io/en/v0.16/modules/features/tempo.html`

on the musical semantics. This could be improved upon in feature work by making use of the current findings in music embeddings and latent music representations.
By matching musical semantics to image semantics, more coherent visualisations can be created.

The Algorithm

- *Chroma-based Interpolation*: One of the algorithms Kraasch and Pasquier described in the Autolume paper [20] was not discussed and evaluated on performance in this research project: the Chroma-based Interpolation algorithm.
  It could be worth also quantifying the performance of the Chroma-based Interpolation algorithm with the metrics mentioned here.

- *$\mathcal{W}$-latent space hypersphere*: A crucial error slipped into the code of the hypersphere algorithm: the hypersphere is created in the $\mathcal{Z}$ latent space, while inference happens from the $\mathcal{W}$ latent space. This means that the trajectory described by moving on the hypersphere will not result in a perfect circle, since the mapping from the $\mathcal{Z}$ latent vector to the $\mathcal{W}$ latent vector will deform the hypersphere. This is a bug that should be fixed in order to assure uniform visualisations in local uniform manifolds.

- *Long-term structure*: Although an artist can actively switch system parameters between different sections of the music, this process is not automatized by default.
  Analysis of long-term structure and adapting the system parameters to that, by for example using Laplacian Segmentation [45] could be a subject for future research.

Semantic Control

- *Text-to-Image Control*: An initial goal of this project was to implement text-based control over the direction vectors being used at each time. For that, existing combinations of the OpenAI's text-to-image model CLIP [74] and StyleGAN based models exist. Examples from literature are CLIP2StyleGAN [75] by Abdal et al., allowing to find semantically meaningful directions of change by the means of text, and StyleCLIP [76] allowing for text-driven manipulation of images. Due to timing constraints, this text-to-image feature was not implemented, but could be a subject for further research.

- *Live Coding Interface*: The current setup only offers live control via a MIDI-control, not through a graphical user interface (GUI) or command-line interface (CLI). Seen the close ties in goals and characteristics of this project with the goals and characteristics of the Live Coding community, it could be interesting to make all control based on a CLI, which would allow for seamless integration of text-to-image control as well.

- *Network Bending & Model Rewriting*: The techniques of Network Bending [46] and Model Rewriting were not discussed here. The reason for this that it did not seem like an optimizable part of the Autolume pipeline. It could however be interesting to measure some performance metrics when using these two control techniques as well.
- *Performance of closed-loop approach*: In Section 9.1 a closed-loop approach was proposed, where artificially generated imagery could be used to train the synthesis network.
  The performance of a system like this was not measured in-depth. In future work this could be quantified and possible artefacts inherent to this approach could be reported.

The Model

- *Higher Resolution VAE*: The β-VAE experiments in this project were all performed on low-resolution VAE models, based on the knowledge from literature that VAEs underperform when handling higher resolutions. This was not empirically tested, however, and it could be worth experimenting with higher resolution VAE-based systems.
- *GPU Utilization*: As mentioned in Section 10.2.6, the reasons for the increased GPU Utilization are still unclear and should be further researched.
- *GAN Compression*: Kraasch and Pasquier compressed the StyleGAN2 models they employed for synthesis in order to increase the achievable frame rate [20]. The same approach could be implemented on the algorithms and models proposed in this research project to further increase frame rate and decrease lip sync error.

11.3 Broader Impact

This research project tried to quantify and improve upon the current state-of-the-art in generative model based real-time audio visualisation, and is believed to have contributed to this specific field. By measuring performance quantitively for the first time, this work lays down a clear foundation of performance benchmarks, paving the way for future studies.
Generative model based VJing tools could provide the current VJing scene with a unique new visual look and creative process. Moreover, the techniques discussed in this project were specifically designed to be user-friendly, lowering the barrier to get into VJing as an artist.
The system could prove useful in settings where a live VJ is no feasible solution. Small venues, or even home parties that had no access to live audio visualisation before, could use the system as proposed in this research project to elevate the musical experience of the event, without outrageous costs.

It should however be noted, that the current rise in generative AI is of major concern to many artists and creatives working in the field. As a researcher, I think these concerns are valid and should be taken into account when developing a system based on generative technology. As discussed in Section 9.1 the data where the synthesis model is trained on, should be gathered and employed with permission of the artists who created it.
The topic of job displacement for current VJs is presently not an imminent worry, since the intrinsic value of human intervention remains indisputable. Throughout this discourse, the symbiotic relationship between human artistry and the system's outputs has been emphasized. The artistic intuition and responsiveness to audience energy inherent to a VJing performance cannot be replicated autonomously by the systems described in this study. However, researchers must exercise caution in the progression of these technologies and their integration into real-world scenarios.

On a final note, the impact of the created system seems to stretch beyond just the artistic one. As is common knowledge within the field, the inner workings of deep generative models are still far from completely understood, with neural networks often still working as black-boxes.
I believe that by welcoming people from different backgrounds to experiment with the latent spaces of generative models, novel insights and alternative perspectives can be cultivated.
Opening up the field to a new group of creative thinkers, could open the door for new ideas.

Bibliography

[1] *IBM - Artificial Intelligence*. https://www.ibm.com/topics/artificial-intelligence. Accessed: July 2, 2023. 2023.

[2] *Artificial intelligence (AI) market size worldwide in 2021 with a forecast until 2030*. https://www.statista.com/statistics/1365145/artificial-intelligence-market-size/%23:%7E:text=According%20to%20Next%20Move%20Strategy%2C%20a%20vast%20number%20of%20industries.. Accessed: July 2, 2023. 2023.

[3] *Dall-E*. https://openai.com/dall-e-2. Accessed: July 2 2023. 2023.

[4] *MidJourney*. https://www.midjourney.com/. Accessed: July 2 2023. 2023.

[5] *ChatGPT-3*. https://chat.openai.com/. Accessed: July 2 2023. 2023.

[6] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: (Dec. 2019). URL: http://arxiv.org/abs/1912.04958.

[7] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *International conference on learning representations*. 2016.

[8] Wikipedia. *Musical notation — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Musical%20notation&oldid=1161873581. [Online; accessed 03-July-2023]. 2023.

[9] *Musical notation - Integral Serialism, Microtonal Music, and Space Time Notation — britannica.com*. https://www.britannica.com/art/musical-notation/20th-century-notation. [Accessed 03-Jul-2023].

[10] Richard L. Crocker. "Pythagorean Mathematics and Music". In: *The Journal of Aesthetics and Art Criticism* 22.2 (1963). Publisher: [Wiley, American Society for Aesthetics], pp. 189–198. ISSN: 0021-8529. DOI: 10.2307/427754. URL: https://www.jstor.org/stable/427754 (visited on 07/04/2023).

[11] Niels Hutchison. *Giuseppe Arcimboldo: the musical grey scale*. URL: http://www.colourmusic.info/arcim.htm.

[12] *A Brief History of Visual Music*. [Accessed 03-Jul-2023]. Sept. 2019. URL: https://overprocessedthinking.com/brief-history-of-visual-music/.

[13] Maarten Franssen. “The ocular harpsichord of louis-bertrand castel”. In: *Tractrix* 3.1991 (1991), pp. 15–77.

[14] Jenny Hans. *Cymatics: A Study of Wave Phenomena and Vibration*. 2001.

[15] William Moritz. “The dream of color music, and machines that made it possible”. In: *Animation World Magazine* 2.1 (1997).

[16] Pablo Perez Zarate. “Expanding Minds With Visual Music”. In: *Proceddings Of Understanding Visual Music 2015 Symposium*. 2015, p. 74.

[17] Annet Dekker. “Synaesthetic performance in the club scene”. In: *3rd Conference on Computational Semiotics for Games and New Media*. 2003, p. 24.

[18] Robin Oppenheimer. *Maximal art: The origins and aesthetics of West Coast Light Shows*. Apr. 2009. URL: https://rhizome.org/editorial/2009/apr/15/maximal-art-the-origins-and-aesthetics-of-west-coa/.

[19] David Fodel. “Live Cinema: Context and ” Liveness ””. In: May 2013.

[20] Jonas Kraasch and Philippe Pasquier. “Autolume-Live: Turning GANs into a Live VJing tool”. In: *Proceedings of the 10th Conference on Computation, Communication, Aesthetics & X* (2022). Publisher: University of Porto, pp. 168–185. DOI: 10.24840/xcoax_2022_45.

[21] James Davis, Yen-Hao Hsieh, and Han-Chung Lee. “Humans perceive flicker artifacts at 500 Hz”. In: *Scientific Reports* 5 (2015), p. 7861. DOI: 10.1038/srep07861.

[22] Paul Read and Mark-Paul Meyer. *Restoration of motion picture film*. Elsevier, 2000.

[23] Doris Baltruschat, Jennifer Grek Martin, and Ernest Mathijs. “The reception of The Hobbit in Canada: Does fantasy need technology?” In: *technology* (2016).

[24] “Rec. ITU-R BT.1359-1 1 RECOMMENDATION ITU-R BT.1359-1 RELATIVE TIMING OF SOUND AND VISION FOR BROADCASTING”. In: 1998. URL: https://api.semanticscholar.org/CorpusID:155097085.

[25] Advanced Television Systems Committee. *ATSC Implementation Subcommittee Finding: Relative Timing of Sound and Vision for Broadcast Operations*. Technical Report IS-191. 1750 K Street, N.W., Suite 1200, Washington, D.C. 20006, June 2003.

[26] Hans Brouwer. *Audio-reactive Latent Interpolations with StyleGAN*. Tech. rep. 2020. URL: http://arxiv.org/abs/1912.04958..

[27] *IBM - Machine Learning*. https://www.ibm.com/topics/machine-learning. [Accessed: June 26, 2023]. 2023.

[28] Martin Jaggi and Nicolas Flamarion. *Machine Learning*. Ecole Polytechnique Fédérale de Lausanne, 2021.

[29] dr. ir. De Boom Cedric and dr. ir. Verbelen Tim. *Deep Generative Models*. Ghent University, 2022.

[30] Tomas Brants and Ide Van den Borre. "Prognostic Outcome Prediction in Head and Neck Cancer using Deep Learning on Pre-Treatment PET/CT". MA thesis. Ghent, Belgium: Ghent University, 2022.

[31] G. Cybenko. "Approximation by superpositions of a sigmoidal function". en. In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274 (visited on 07/05/2023).

[32] Abhinav Saurabh, Adyasha Sahu, and Chellavelu Vijayakumaran. "Multiple diseases detection using deep learning". In: (Jan. 2020), pp. 3546–3555.

[33] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes*. arXiv:1312.6114 [cs, stat]. Dec. 2022. URL: http://arxiv.org/abs/1312.6114 (visited on 07/05/2023).

[34] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: (June 2014). URL: http://arxiv.org/abs/1406.2661.

[35] John von Neumann and Oskar Morgenstern. *Theory of games and economic behavior. With an introduction by Harold Kuhn and an afterword by Ariel Rubinstein.* English. 4th print of the 2004 sixtieth-anniversary ed. Princeton, N J: Princeton University Press, 2007. ISBN: 978-0-691-13061-3.

[36] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: (Dec. 2018). URL: http://arxiv.org/abs/1812.04948.

[37] Tero Karras et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2017. DOI: 10.48550/ARXIV.1710.10196. URL: https://arxiv.org/abs/1710.10196.

[38] Tero Karras et al. *Training Generative Adversarial Networks with Limited Data*. en. arXiv:2006.06676 [cs, stat]. Oct. 2020. URL: http://arxiv.org/abs/2006.06676 (visited on 03/02/2023).

[39] Mikołaj Bińkowski et al. *Demystifying MMD GANs*. arXiv:1801.01401 [cs, stat]. Jan. 2021. URL: http://arxiv.org/abs/1801.01401 (visited on 07/19/2023).

[40] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper%5C%5Ffiles/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[41] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: (Nov. 2015). URL: http://arxiv.org/abs/1511.06434.

[42] Yujun Shen et al. "Interpreting the Latent Space of GANs for Semantic Face Editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[43] Wikipedia. *Chroma feature — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Chroma%20feature&oldid=1102902598. [Online; accessed 20-July-2023]. 2023.

[44] Ali Mottaghi et al. "OBTAIN: Real-Time Beat Tracking in Audio Signals". In: *International Journal of Signal Processing Systems* 5 (Apr. 2017). DOI: 10.18178/ijsps.5.4.123-129.

[45] Brian McFee and Dan Ellis. "Analyzing Song Structure with Spectral Clustering." In: *ISMIR*. Citeseer. 2014, pp. 405–410.

[46] Terence Broad, Frederic Fol Leymarie, and Mick Grierson. "Network Bending: Expressive Manipulation of Deep Generative Models". en. In: *Artificial Intelligence in Music, Sound, Art and Design*. Ed. by Juan Romero, Tiago Martins, and Nereida Rodrıguez-Fernández. Vol. 12693. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 20–36. ISBN: 978-3-030-72913-4 978-3-030-72914-1. DOI: 10.1007/978-3-030-72914-1_2. URL: http://link.springer.com/10.1007/978-3-030-72914-1%5C%5F2 (visited on 07/26/2023).

[47] Pierre Soille. "Erosion and Dilation". In: *Morphological Image Analysis: Principles and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–103. ISBN: 978-3-662-05088-0. DOI: 10.1007/978-3-662-05088-0_3. URL: https://doi.org/10.1007/978-3-662-05088-0%5C%5F3.

[48] Yuchen Liu et al. *Content-Aware GAN Compression*. arXiv:2104.02244 [cs]. Apr. 2021. URL: http://arxiv.org/abs/2104.02244 (visited on 05/29/2023).

[49] Erik Härkönen et al. "GANSpace: Discovering Interpretable GAN Controls". In: (Apr. 2020). URL: http://arxiv.org/abs/2004.02546.

[50] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. arXiv:1706.08500 [cs, stat]. Jan. 2018. URL: http://arxiv.org/abs/1706.08500 (visited on 08/04/2023).

[51] Christian Szegedy et al. *Going Deeper with Convolutions*. arXiv:1409.4842 [cs]. Sept. 2014. URL: http://arxiv.org/abs/1409.4842 (visited on 08/04/2023).

[52] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[53] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[54] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". en. In: *Image Analysis*. Ed. by Gerhard Goos et al. Vol. 2749. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. ISBN: 978-3-540-40601-3 978-3-540-45103-7. DOI: 10.1007/3-540-45103-X_50. URL: http://link.springer.com/10.1007/3-540-45103-X%5C%5F50 (visited on 08/04/2023).

[55] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[56] Xiph.Org Foundation. *Xiph.Org Test Media*. Accessed on: August 4 2023. Xiph.Org Foundation, 2023. URL: https://media.xiph.org/video/derf/.

[57] The PyAudio Authors. *PyAudio: Python bindings for PortAudio*. https://people.csail.mit.edu/hubert/pyaudio/. Accessed on July 30, 2023. 2022.

[58] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015.

[59] J.P. Bello et al. "A tutorial on onset detection in music signals". en. In: *IEEE Transactions on Speech and Audio Processing* 13.5 (Sept. 2005), pp. 1035–1047. ISSN: 1063-6676. DOI: 10.1109/TSA.2005.851998. URL: http://ieeexplore.ieee.org/document/1495485/ (visited on 07/31/2023).

[60] Sebastian Böck et al. "Online real-time onset detection with recurrent neural networks". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12), York, UK*. sn. 2012, pp. 17–21.

[61] Sebastian Böck et al. "madmom: a new Python Audio and Music Signal Processing Library". In: *Proceedings of the 24th ACM International Conference on Multimedia*. Amsterdam, The Netherlands, Oct. 2016, pp. 1174–1178. DOI: 10.1145/2964284.2973795.

[62] Georg Pólya. "Über eine Aufgabe der Wahrscheinlichkeitsrechnung betreffend die Irrfahrt im Straßennetz". In: *Mathematische Annalen* 84.1 (Mar. 1921), pp. 149–160. ISSN: 1432-1807. DOI: 10.1007/BF01458701. URL: https://doi.org/10.1007/BF01458701.

[63] Serguei Popov. *Two-Dimensional Random Walk: From Path Counting to Random Interlacements*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2021. DOI: 10.1017/9781108680134.

[64] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. *Testing the Manifold Hypothesis*. 2013. arXiv: 1310.0425 [math.ST].

[65] Wikipedia contributors. *N-sphere*. Accessed: 16 May 2023. 2023. URL: https://en.wikipedia.org/wiki/N-sphere.

[66] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[67] Yujun Shen and Bolei Zhou. "Closed-Form Factorization of Latent Semantics in GANs". In: (July 2020). URL: http://arxiv.org/abs/2007.06600.

[68] Antonia Creswell and Anil Anthony Bharath. *Inverting The Generator Of A Generative Adversarial Network*. arXiv:1611.05644 [cs]. Nov. 2016. URL: http://arxiv.org/abs/1611.05644 (visited on 05/30/2023).

[69] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. *A Neural Algorithm of Artistic Style*. arXiv:1508.06576 [cs, q-bio]. Sept. 2015. URL: http://arxiv.org/abs/1508.06576 (visited on 08/10/2023).

[70] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[71] Tero Karras et al. "Training Generative Adversarial Networks with Limited Data". In: *Proc. NeurIPS*. 2020.

[72] Maximilian Seitzer. *pytorch-fid: FID Score for PyTorch*. https://github.com/mseitzer/pytorch-fid. Version 0.3.0. Aug. 2020.

[73] Jonathan Driedger, Meinard Müller, and Sascha Disch. *Extending Harmonic-Percussive Separation of Audio Signals*. Jan. 2014.

[74] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. arXiv:2103.00020 [cs]. Feb. 2021. URL: http://arxiv.org/abs/2103.00020 (visited on 05/30/2023).

[75] Rameen Abdal et al. *CLIP2StyleGAN: Unsupervised Extraction of StyleGAN Edit Directions*. arXiv:2112.05219 [cs]. Dec. 2021. URL: http://arxiv.org/abs/2112.05219 (visited on 05/30/2023).

[76] Or Patashnik et al. "StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery". en. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 2065–2074. ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.00209. URL: https://ieeexplore.ieee.org/document/9710854/ (visited on 05/29/2023).

[77] NVidia Corporation. *FFHQ dataset*. https://github.com/NVlabs/ffhq-dataset. GitHub repository. 2014.

A Theorems

A.1 Theorems in Probability Theory

Theorem A.1.1 (Chain Rule). *[29] If $A_1, A_2, \ldots, A_n$ are events with $p(A_i) > 0$, then:*

$$p(A_1, A_2, \ldots, A_n) = p(A_1) \cdot (A_2|A_1) \ldots (A_n|A_1, A_2, \ldots A_{n-1})$$

Theorem A.1.2 (Bayes' Rule). *[29] If A and B are events with $p(A) > 0$ and $p(B) > 0$, then:*

$$\begin{aligned} p(A|B) &= \frac{p(A,B)}{p(B)} \\ &= \frac{p(B|A) \cdot p(A)}{p(B)} \end{aligned}$$

Theorem A.1.3 (Statistical Independence). *[29] If A and B are independent events, then:*

$$p(A|B) = p(A) and p(B|A) = p(B)$$

Consequence via chain rule:

$$p(A,B) = p(A) \cdot p(B)$$

Theorem A.1.4 (Jensen's Inequality). *For a concave function f:*

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$$

Theorem A.1.5 (Monte Carlo Estimation). *The Monte Carlo simulation of an expected value $\hat{\mu}$ is given by*

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

B Overview of used model Architectures

B.1 CelebA 128×128 β-VAE encoder architecture

Table B.1: Description of the CNN encoder architecture of the trained β-VAE used in Chapter 10, the decoder is the inverse of the encoder.

Operation layer	**Number of Filters**	**Kernel Size**	**Output Size**
Input images	-	-	$3 \times 128 \times 128$
Convolution	128	7×7	
BatchNorm2D	-	-	$128 \times 42 \times 42$
LeakyReLU	-	-	
Convolution	256	7×7	
BatchNorm2D	-	-	$256 \times 14 \times 14$
LeakyReLU	-	-	
Convolution	512	7×7	
BatchNorm2D	-	-	$512 \times 4 \times 4$
LeakyReLU	-	-	
	1		
Convolution	1024	7×7	
BatchNorm2D	-	-	$1024 \times 1 \times 1$
LeakyReLU	-	-	
	1		
Flatten	-	-	4096
Fully Connected (Mean)	-	-	128
Fully Connected (Variance)			

C The Datasets

The audio visualisation systems in this project were all based on models trained on different datasets of human faces. The reason for this being that the human brain is particularly strong at distinguishing fine details in human faces. It even has specialized regions for human face recognition, such as the occipital face area (OFA) and the fusiform face area (FFA). The aim of using of visualisations of human faces was thus to have a more precise impression of details upon visual inspection of the results. In this project, the FFHQ and CelebA datasets were used at resolutions of 512×512 and 128×128, respectively [77, 70].

C.1 FFHQ

Flickr-Faces-HQ (FFHQ) is a high-quality image dataset of human faces, originally created as a benchmark for generative adversarial networks (GAN) and proposed in the work by Goodfellow et al. [34].

The dataset consists of 70,000 high-quality PNG images at 1024×1024 resolution and contains considerable variation in terms of age, ethnicity, and image background. It also has good coverage of accessories such as eyeglasses, sunglasses, hats, etc. The images were crawled from Flickr, thus inheriting all the biases of that website, and were automatically aligned and cropped. Only images under permissive licences were collected. Various automatic filters were used to prune the set, and finally Amazon Mechanical Turk was used to remove the occasional statues, paintings, or photos of photos. In this project, the images were resized to a resolution of 512×512.

Figure C.1: Sample of the FFHQ 1024 × 1024 dataset [77].

C.2 CelebA

The CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations. For this project, the images were rescaled to a resolution of 128 × 128.

Figure C.2: Sample of the CelebA dataset [70].

D The Software & Performances

D.1 Public Release

The software created in this project for testing purposes will be cleaned up and made publicly available under the terms of the GNU General Public License as published by the Free Software Foundation [1] by Fall 2023.

D.2 Live Performances with the created system

D.2.1 Jokerweek 2023 @Duivelsteen, Ghent

At the projectweek of the Architectural Engineering students of Ghent University, a VJ set was performed with the system architecture presented in this research paper in combination with prerecorded clips through the VJing software Resolume.
Projection mapping was performed to visualise a full image on 2 cathode ray television systems and a projector.

D.2.2 Algorave @De Roes, Ghent

At the 2023 Ghent Algorave, an audiovisual live coded set was performed together with Miel Peeters and Theo Depraetere as the collective 'Break;'. Music was Live Coded through the sequencing software Orca [2]. OSC-signals were sent over a local network and used to steer

[1] https://www.gnu.org/licenses/
[2] Orca Documentation: `https://100r.co/site/orca.html`

Figure D.1: Jokerweek 2023 @Duivelsteen, Ghent

the *Hypersphere Interpolation* algorithm in the StyleGAN2 and β-VAE latent spaces. The visualisations created this way were streamed over a local network and post-processed in the Live Coding environment Hydra [3]. By making use of the latent space projection methods described in Section 9.3, the faces of all three artists were projected in the StyleGAN2 latent space. Their latent codes were presaved and mapped to a MIDI-controller button. The performance was ended with an interpolation in between the three faces, increasing in speed up until the end of the set.

D.2.3 Interactive Installation 'Latent Spaces' @Odus, Dentergem

Latent Spaces *"A collaborative dance between machine and individual. A generative artificial model was trained to generate non-existing data in real-time based on a musical input. While human improvisation drives musical creation, the machine dreams images based on that human input. This way a creative conversation is set up between man and machine. Start talking. Explore the latent space." — Arthur Deleu*

For arts and music festival Odus organised in Dentergem by Odus Collective [4], an interactive system was designed. Visitors were invited to improvise on a background soundscape, by playing on a MIDI-synthesizer. The improvisation was used as an input to steer the *Hypersphere Interpolation* algorithm explained in Section 8.3. This way, visitors not only had the opportunity to improvise music, but at the same time improvise in the visual realm through latent space exploration.

[3] Hydra Documentation: `https://hydra.ojack.xyz/`
[4] Official Odus facebook page: `https://www.facebook.com/oduscollective/`

The imagery was projected on 3 cathode ray television systems. The soundscape was composed by musician Blixa Declerck.

Figure D.2: Algorave @De Roes, Ghent

Figure D.3: 'Latent Spaces' @Odus, Dentergem.