Universiteit Leiden

Master Computer Science

ELECTROGAN: TOWARDS HIGH FIDELITY ELECTRONIC MUSIC GENERATION WITH STYLEGAN2

Name:	B.D. Havenaar
Student ID:	s3026531
Date:	15/08/2023
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. E.M. Bakker
2nd supervisor:	Prof. dr. M.S.K. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Generating convincing audio samples in the waveform domain is a complex and unsolved task. Various approaches have been proposed using Generative Adversarial Networks. Models developed for the speech generation domain often output insufficient sampling rates that do not meet the Nyquist criterion when generating music samples. Especially for music, having a high enough resolution is required for a good listening experience. In this paper, we show how to synthesize convincing audio samples using a StyleGAN2-based generative model. We create a large high quality 16 bit, 44.1k samples/second training dataset of approximately 540k house and techno music samples in a Short Time Fourier Transform representation. We explore various parameter settings and training tactics. We evaluate the resulting models with a Fréchet Inception Distance and show that for our application this metric is a valid alternative for a mean opinion score (MOS). For the best configurations we obtain a MOS score of 2.46, and 2.96 for studio mastered music clips, compared to 3.94 for real data.

1 Introduction

Innovations in instrument-, production- and performance technology have been a strong driver for creativity in the music industry. For instance, the invention and widespread availability of the synthesizer has led to the development of completely new genres. Also, the increase in computer performance and the uprising of Digital Audio Workstation (DAW) software have made audio production accessible for anyone with a modest workstation. As computational power continues to improve and with the continuous development of machine- and deep learning techniques, the adoption of such AI models in the music production process is inevitable. This upcoming field has seen various approaches ranging from models that produce sound effects, compose music notes, and the generation of complete songs.

The task of audio generation in general has received considerable attention through the desire to create high quality Text To Speech (TTS) models. Such models are already widely implemented through applications such as Alexa, Siri, or Amazon Polly. In comparison to speech generation, music generation is vastly more complex. High-quality speech requires about 16k samples per second and the length of individual utterances can be a fraction of a second. Due to a wider range of active frequencies, high quality music requires a resolution of approximately 44.100 samples per second at 16 bits per sample. Music covers the complete audible range of the human ear on average ranging from about 50Hz to 22.000 Hz. This gives a much broader range of frequencies that needs to be distinguished from the human ear. Following the Nyquist criterion a signal needs a sampling frequency of at least twice the maximum observable frequency. This results in that the amount of required samples necessary to create a high-quality audio signal is around 44.000 per second.

Another way of describing the complexity of generating music is through the large variety of different levels at which individual samples are correlated. On the frequency level, samples are in general correlated with nearby samples in the μs range. This is for example the case between two neighboring frequency cycles. At the same time on a much higher level samples also correlate. For example on an instrumental expression such as *vibrato* effects lead to correlating samples several ms apart. Melodies and music keys have an even wider span of tens of seconds or even minutes. The extreme difference in the magnitude of inter-sample correlations makes generating music a complex task. [1].

Apart from some recent examples, researchers often aim to create complete classical piano compositions. Such compositions contain complex melodies and chord progressions that can span several minutes of music. Due to the fact that the key frequencies of piano music are between approximately 30 and 4500hz, it often suffices to have a lower sampling frequency for piano music, although this brings some loss in higher harmonic frequencies. On the other hand piano music can have very long and extensive inter-sample correlations in long medolic pieces. This is for example the case when considering complex, almost mathematically composed classical piano pieces like Bach. In contrast to such piano compositions, most modern-day Western music genres follow the simplest com-

mon time signature of 4/4 beats and are highly repetitive. Especially in genres like *techno*, *electro*, *house*, *disco*, *drum & bass*, and other electronic music-based genres this often holds. Where piano compositions require tens of seconds of comprehensive audio to create a believable song, most electronic-based genres only require a few bars (4 beats) that can be looped into longer pieces.

The focus of this research is to address the challenge of generating convincing high-quality audio sample loops within the electronic music domain (Techno, House, Electro). StyleGAN2 generative models are trained and used to generate images of Short Time Fourier Transform representations of electronic music samples. The models are evaluated using the Fréchet Inception Distance and mean opinion scores.

2 Related Work

In this section, we give an overview of various important and state-of-the-art methods for generating audio in the symbolic and waveform domains.

2.1 Symbolic Domain

In the symbolic domain music is described in abstracted forms such as MIDI notes, musical scores or guitar tabs. Such representations have the advantage of low memory requirements as they only contain information about the timing, velocity and pitch of notes played. They can be played by sequencers or musicians that operate one or many instruments at once. The disadvantage of those representations is that they only capture a part of the information on how the piece is played. Not capturing unique details that depend on individual instruments or musicians [2]. Auto regressive models have been used for generating music through this symbolic approach such as RNN-, LSTM-[3] and VAE's[4].

2.2 Autoregressive models

One of the state-of-the-art models that generate audio in the waveform domain is WaveNET [5]. This model uses 1-dimensional convolutions to predict the audio sample at time step $t + 1$ by evaluating its conditional probability given all previously seen samples $p(x_{t+1}|x_1, x_2, ..., x_t)$. By stacking increasingly larger dilated convolutions, the model has a very flexible and potentially large receptive field. This way it is able to capture local dependencies between individual samples at the audio wavelength level. On the other hand, the model is also capable of capturing longer-term multi-second dependencies such as chord progressions or word utterances. The conditioning of the model can be extended to other input features for the purpose of speech generation (TTS). The main disadvantage of this approach is the computational complexity of both training and inference. By parallelizing the WaveNET architecture using Probability Density Distillation inference time is improved towards realtime [6]. Although the use of sparse RNNs can increase inference efficiency by an order of magnitude, inference speed still remains far below real-time [7]. In the meantime, also other auto-regressive models in the waveform domain have been proposed such as SampleRNN [8] which lead to similar inference time issues.

2.3 Adversarial models

Another approach is using Generative Adversarial Networks, or GANs to generate an audio sequence [2]. Two of the first implementations of this are WaveGAN and SpecGAN [9]. Both models utilized DCGAN [10] architectures. In the case of WaveGAN, the original two-dimensional architecture is flattened to single-dimensional convolutions. SpecGAN uses the original 2-dimensional DCGAN model, but here the input data is transformed to a two-dimensional spectrogram representation using a Shortime Fast Fourier Transform (SFFT).

GANs are known to create checkerboard-like artifacts during the deconvolution phase, as described by [11]. In the audio domain, those artifacts lead to overlapping frequencies that can be perceived as pitched noise. Although the authors propose a method for ensuring this artifact is not used as a trivial recognizable feature for the discriminator, the high pitch is still recognizable in the generated audio.

Another GAN-based generative network in the waveform domain is MelGAN. This model is trained to perform mel spectogram to waveform inversion. The model uses the mel spectogram representation as an intermediate step to train conditioned generator models which then can be used for tasks like style transfer [12].

GANSynth is also based on mel spectograms and is focused on creating single instrument sounds [13]. This model was trained using progressive GANS from the StyleGAN architecture as proposed by [14]. They show that latent space interpolations contain recognizable instrument features such as timbre. Although the generated musical sounds are highly convincing, it differs in complexity from generating complete electronic high quality music compositions.

StyleGAN2[15] is a further improvement of the StyleGAN model, making it the most interesting candidate architecture for generating high quality electronic music at the time of this research. In [16], the authors have successfully experimented with applying UNGAN [17], StyleGAN and StyleGAN2 models to clean drum loops. This dataset however consisted of clean drum loop recordings from a sampling database used for music productions. In real world electronic music examples the music samples can consists of a variety of drum loops, synths and other instruments playing together making it potentially harder to generate convincing samples.

2.4 Other applications within music generation domain

Authors of Jukebox [18] used Vector Quantized Variational Auto Encoders (VQ-VAE) [19],[20] to create long sequences of audio of arbitrary length. They use multiple VQ-VAE's on multiple temporal resolutions to generate latent codes used to later reconstruct the audio signal. Furthermore sparse transformers [21], [22], and other auto regressive models are used to reconstruct the individual resolution layers at inference. Although this results in conditioned, highly convincing audio of arbitrary length, the inference time is multiple hours on state-of-the-art hardware to generate a minute of music.

While undertaking this research more interesting approaches have appeared that yielded good results in the domain of creating high fidelity audio samples. Musika [23] uses a two step encoder that first encodes the input STFT to an intermediate latent space compression, which then is further decoded to a more compact latent space. By using the intermediate latent space as a coordinate system during the decoding phase, authors are able of generating multiple adjacent generated outputs to create sequences of arbitrary length. By using a decoder to translate back into a magnitude and phase representation that can

be reversed using an inverse STFT (iSTFT), the use of a separate phase reconstruction method can be omitted.

Another recent approach that may not be let unnoticed is Riffusion [24]. Here the us of Stable Diffusion [25] models enable the model of translating from one representation to another. For example with from text or image inputs to audio. Authors use the Griffin-Lim algorithm to reconstruct phase information. Due to the vast computational and data resources required, this approach is not considered for this research.

3 Contributions

This paper has the following core contributions. We create a the Mixsets dataset, large collection of electronic music containing 2500 hours of electronic music samples. We evaluate various hyperparameter settings for StyleGAN2, to generate long, complex, and comprehensive music samples of electronic music that last tens of seconds. We show that it is attainable to generate near natural-sounding audio pieces with a high-quality sampling rate of 44.1kHz at a 16 bits resolution. We evaluate generated results using Mean Opinion Scores and the Frechet Inception Distance (FID). We show that although the FID score is based on visual images it also is valid alternative for MOS scores when analyzing the performance of GANs that are trained to generate STFT representations for generating music.

4 Fundamentals

In the following section, we explain the working of Generative Adversarial Networks and the development towards at the time of this reserach state of the art, StyleGAN2.

4.1 Convolutional Neural Network

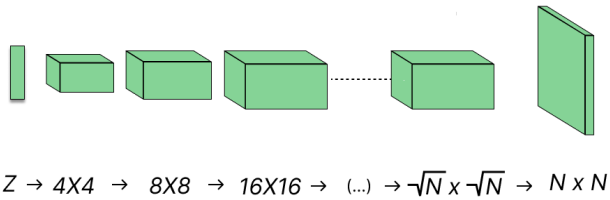
A convolutional neural network (CNN) is a type of deep learning architecture that has been extensively used in image processing and computer vision tasks and first applied to a representational learning problem by [26]. The basic idea behind CNNs is to learn feature representations of input data by applying a series of convolutional filters to it. These filters scan the input data, identify patterns and features, and transform them into meaningful representations that are useful for the desired task.

The structure of a CNN has similarities to the biological process of visual perception in the human visual cortex, where visual cells are pooled to deeper cells in the cortex [27]. The convolutional layers of a CNN perform a series of computations on the input data, using the weights of the filters to extract relevant features from it. These features are then passed on to the next layers of the network, which perform additional processing, such as pooling or flattening, to further refine the feature representations.

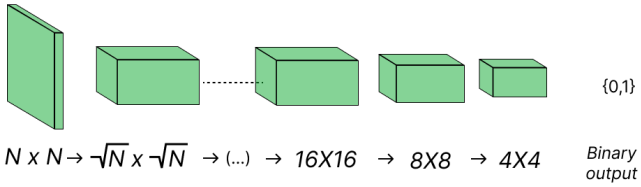
One of the key benefits of CNNs is their ability to learn hierarchical feature representations of the input data. By stacking multiple convolutional layers on top of each other, the network can learn increasingly complex and abstract features, which enables it to handle more complex and challenging visual tasks. Additionally, CNNs can leverage the spatial relationships between pixels in an image, allowing them to capture local patterns and structures that are critical for accurate image classification and segmentation. Once a representation is learned, a CNN can also be utilized in generative models.

4.2 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a deep neural network architecture designed to learn to generate new content similar to a given target dataset. It was first introduced by [28] and consists of two primary components: a generator G and a discriminator D, both of which compete in a min-max game. The generator takes as input a latent random vector z and has the same output as the shape of the training samples, often images. The discriminator takes as input either a generated or a real image and outputs if a given batch of images is real or fake. The generator's objective is to produce a batch of images that are indistinguishable from real ones, whereas the discriminator aims to distinguish images between real and fake ones. By randomly showing the discriminator batches of either real images or fake images from the generator, both networks can learn from each other's mistakes.



(a) Visualization of the DCGAN Generator architecture. The network starts with a random latent vector Z as input vector which is successively up-sampled with increasingly large convolutional layers until the desired $N \times N$ size is obtained.



(b) Visualization of the DCGAN Discriminator architecture. Has $n \times m$ input which is down-sampled using successive convolutions to eventually end up with a binary output. Where the binary output determines if the input image is real or fake.

Figure 1: DCGAN [10] Generator and Discriminator architectures.

The use of convolutional layers [26] further optimizes the GAN architecture for generating images [10]. With this architecture, the generator again takes a latent vector z as input and furthermore is connected to a sequence of convolutional layers $\{(2 \times 2), (4 \times 4), (\dots), (n \times n)\}$ as explained in (Figure 1a). The output of the generator now is a two-dimensional $(n \times n)$ image.

For the discriminator the opposite holds, here the input image is of the size $n \times n$ and is subsequently down sampled (Figure 1b) until it reaches a binary output that determines if the image is real or fake.

4.2.1 Loss functions

The original GAN loss, the adversarial loss function was initially proposed by [28]. Equation 1 shows that this loss function consists of two components. Here D is the discriminator and $D(x)$ the probability that x came from real data, and $G(x)$ is the generated data from generator G. The first part is the discriminators' prediction that a batch of real-world samples is real, the second part is the inverse log probability that a batch of generated samples is real. The generator uses the loss function values to update it's weights with the goal to maximize this probability, whilst the discriminator aims to reach the opposite.

$$\min_G \max_D V(D,G) = [\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))] \quad (1)$$

In practice, this min-max loss saturates the weights in the generator making it hard for the generator to keep up with the discriminator. This happens when the generator produces samples that are easily recognized as fake by the discriminator, leading to a very high loss for the generator. As a result, the generator can become stuck in a state where it generates only a limited set of samples that the discriminator can easily classify as fake, without improving its ability to generate more realistic samples.

To prevent this a R_1 Regularization gradient penalty was introduced that only penalizes when the discriminator evaluates real data [29], [30]. This loss function optimizes the generator for maximizing the probability that a produced batch is deemed real, instead of it minimizing the probability of not being deemed fake. This way the generator is also updated with powerful gradients without considering losses from the discriminator assessing real-world examples. The regularization parameter γ for R_1 Regularization controls the amount of penalty that is given whenever the discriminator moves towards a Nash Equilibrium [31]. Therefore with a higher value for γ the stability of training is expected to increase while the diversity of the output should decrease. The influence of γ on the quality of the results is studied and experimentally demonstrated in our work.

4.3 StyleGAN

The StyleGAN models and their predecessors provide various improvements to the GAN architecture and are based on the Wasserstein GAN with R_1 Regularization gradient penalty (WGAN-GP) [29] [32]. The use of a gradient penalty greatly improves the stability of the training of the GANs. By penalizing the discriminator's gradients, it prevents extreme large values. Gradient penalty is an improvement on weight clipping, where both have the same goal of maintaining Lipschitz continuity of the discriminator. Choosing the right proportion of gradient penalty is important. If too much gradient penalty is allowed within the discriminator the risk occurs of vanishing gradients, training instability and eventually mode collapse. On the other hand choosing too little gradient penalty may also lead to unstable training due to too large gradients and a discriminator that is unable to learn a meaningfull representation of the presented data.

The stability and speed of training is further improved in Stylegan by using progressively growing GANs [33]. At the start of training both the discriminator and generator begin with the lowest possible resolution images (2×2) and both progressively grow during training. This allows for significantly larger images above (1024×1024) and further improves training speed.

The introduction of the 'Style' part of StyleGAN [34] originates from changes to the architecture inspired by the style transfer literature. In StyleGAN, the random input vector z of the generator is replaced with a learned constant, and a new mapping network is introduced that injects into the various layers of the

generator. This new mapping input provides control over various levels of features in the generated images. When for example generating human faces with a StyleGAN model, low-level features change coarse-grained features such as the shape of the face, while high-level features only affect hair color. Compared to vanilla progressive GANs, StyleGAN further improves quality and efficiency by adding noise inputs and introducing mixing regularization. Apart from disentangling different style features, this also improves overall image quality.

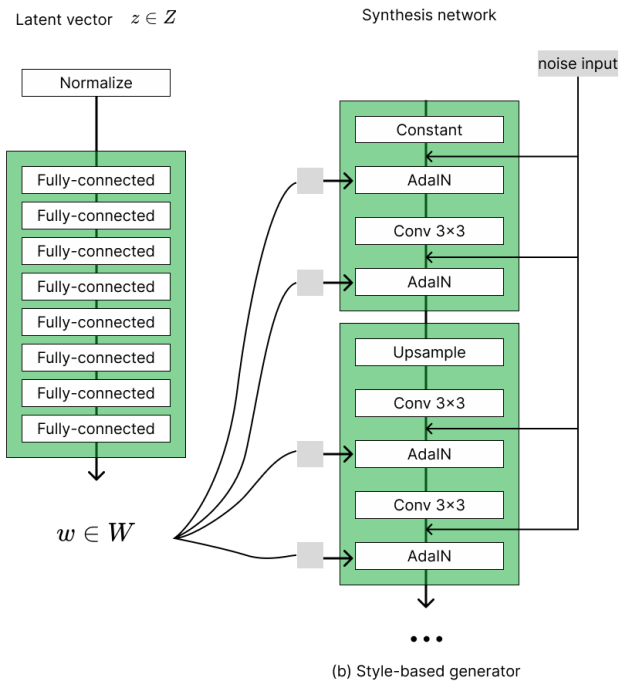
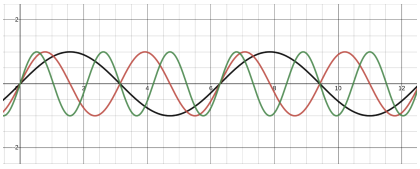


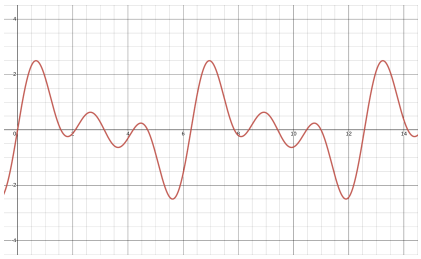
Figure 2: StyleGAN architecture as proposed by [34]. The generator (right) starts with a learned constant vector where a noise input is added, this is followed by adaptive instance normalization that is combined with the latent vector W through a learned affine transformation. The latent vector W is an intermediate latent space which is the output of a fully connected 'mapping' network (left). This network maps the latent space to each individual convolution.

4.4 StyleGAN2

Further improvements in generated image quality and training speed are achieved by reducing regularization term update frequency and introducing an improved alternative for the progressive growth of both the generator and discriminator [15]. Together with improving the size of the network, this configuration yields



(a) Seperate frequencies: $y = \sin(t)$ (black), $y = \sin(2t)$ (red), $y = \sin(3t)$ (green).



(b) The accumulation of frequencies: $y = \sin(t) + \sin(2t) + \sin(3t)$.

Figure 3: Comparison of different frequencies.

better results than its predecessor [34].

4.5 Discrete audio representation

Music and other sounds are de facto frequencies of air vibrations that are observable by hair cells in the human capula. An audio waveform is an accumulation of several base continuous sinusoidals with each their own frequency and magnitude as shown in Figure 3. Real world sounds too are a combination of a large number of stacked frequencies.

In order to store, manipulate and read audio with computers the continuous audio signal must be converted to a digital discrete signal. Here the continuous accumulation of frequencies is chunked into separate discrete samples (Figure 4). The sampling rate or sampling frequency is of high influence when it comes to obtaining the resolving power to distinguish different base frequencies from each other. The Nyquist frequency is the amount of samples that are required to convert a continuous signal to a discrete signal without loosing the resolving power to identify all frequencies independently in the original continuous signal. This required sampling frequency is equal to twice the highest frequency that occurs in the frequency band for the specific application. In case of electronic music, all frequencies audible to the human ear are in uses (50Hz until 22.000 Hz). Therefore the required Nyquist frequency is at least 44.000 samples per second. The reason why twice as many samples are required, is that in order to identify the highest possible frequency in a given spectrum, at least a sample left and right from the peak of this frequency has to be taken.

4.6 Short-Time Fourier Transform (STFT)

The Short Time Fourier Transform is a two dimensional representation of a one dimensional signal such as audio. The signal is decomposed into a two dimensional matrix containing frequency and phase information of the original signal. By creating a two dimensional representation of a signal, one can treat it as an image and enable visual evaluation.

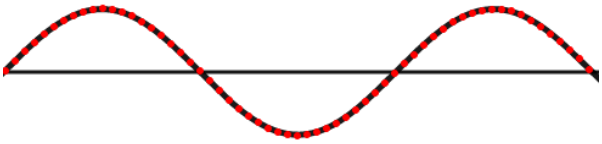


Figure 4: Taking samples from frequency wave. Each red dot represents an individual sample that is taken.

To understand the construction of an STFT, it is essential to first grasp the concept of the Discrete Fourier Transform (DFT). The DFT is a mathematical tool used to convert a discrete sequence of time-domain samples into its equivalent frequency-domain representation, decomposing the signal into its constituent sinusoidal components. However, the DFT assumes stationary, treating the entire signal as a single entity. With audio and speech cases, the signal varies in frequencies and amplitude over time which makes the DFT not suitable analysing non-stationary signals. The DTF (Equation 2) transforms a sequence of audio samples $\mathbf{x} := \{x_0, x_1, ..., x_{N-1}\}$ to its complex valued form $\mathbf{X} := \{\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{N-1}\}$.

$$\mathcal{F}(x_k) = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn} \tag{2}$$

Where N is equal to the total number of samples and k is the index of the frequency component that is calculated. The Short-Time Fourier Transform (STFT) overcomes this limitation by analyzing the signal in small overlapping segments, thereby capturing its time-varying nature (Equation 3). By applying the DFT to each segment, we obtain a time-frequency representation of the signal, revealing how the spectral content evolves over time. The STFT is expressed as follows:

$$X(n,k) = \sum_{m=0}^{N-1} x_{n+m} \cdot w(m) \cdot e^{-\frac{i2\pi}{N}kn} \tag{3}$$

Where N is equal to the total number of samples, k is the index of the frequency component that is calculated and $w(m)$ is the window that is applied to the signal. The ISTFT reconstructs the time-domain signal from its STFT representation by inverting the process of segmenting and overlapping. Given the STFT spectrogram, denoted as $X(n,k)$, where n represents the time index and k denotes the frequency bin, the ISTFT operation aims to recover the original time-domain signal $x(n)$.

The ISTFT can be mathematically expressed as:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(n,k) \cdot e^{\frac{i2\pi}{N} kn} \cdot w^*(k), \tag{4}$$

Where $w^*(k)$ is the complex phase information of the window function $w(k)$ used during the STFT. Additionally, N denotes the window length, representing the number of samples within each segment.

In order to reconstruct the signal magnitude as well as the complex valued phase part of the audio signal must be know. When applying the short time fourier transform X, only the magnitude part of the signal shows visually correlated features. The phase part of the signal is seemingly chaotic and visually looks like noise. Because GANs and CNNs require visual features to be trained on, only a representation of the magnitude part is learned. In order to reconstruct the signal the phase part of the signal $w^*(k)$ must be reconstructed.

4.7 Griffin-Lim Algorithm

The Griffin-Lim algorithm [35] is a widely used and influential technique for reconstructing a time-domain signal from its magnitude spectrogram, making it particularly valuable in audio signal processing and speech synthesis. This algorithm addresses the problem of phase retrieval, where the phase information of a signal is lost during the process of obtaining its magnitude spectrogram. As phase contains crucial information about the temporal structure of the original signal, phase retrieval is essential for accurate signal reconstruction.

The Griffin-Lim algorithm operates iteratively, attempting to converge the complex-valued time-domain signal from its magnitude spectrogram. The initial phase of the signal is randomly generated or, alternatively, estimated from other sources if available. In each iteration, the algorithm computes the complex-valued STFT (Short-Time Fourier Transform) of the current time-domain estimate, keeping the magnitude spectrogram fixed and using the phase information from the previous iteration.

Next, the inverse STFT is applied to obtain a new time-domain estimate, reconstructing the signal with the updated phase information. However, this reconstructed signal usually suffers from inconsistencies, as the inverse STFT does not precisely preserve the magnitude spectrogram. To address this issue, the algorithm refines the reconstructed signal by applying the STFT again to obtain a new magnitude spectrogram. The magnitude of this new spectrogram is then replaced with the original magnitude, and the phase information is retained from the previous iteration.

By iteratively refining the time-domain signal through alternating STFT and inverse STFT operations, the Griffin-Lim algorithm gradually improves the reconstruction's phase information. As the iterations progress, the signal's temporal structure becomes more accurately represented, leading to improved perceptual quality and reduced artifacts.

Although the Griffin-Lim algorithm is effective and computationally efficient, it is essential to note that it may not always yield perfect phase retrieval,

Input: Magnitude spectrogram $|X|$ and number of iterations T
Output: Reconstructed time-domain signal x
Initialize the phase of the signal randomly or by estimation;
for $t = 1$ *to* T **do**
    Compute the complex-valued STFT of the current time-domain estimate: $X_t = |X| \cdot e^{j \cdot \text{angle}(X_{t-1})}$;
    Apply the inverse STFT to obtain a new time-domain estimate: $x_t = \text{ISTFT}(X_t)$;
    Compute the STFT of the new time-domain estimate to obtain a new magnitude spectrogram: $|X_t| = \text{STFT}(x_t)$;
end
Reconstructed time-domain signal: $x = x_T$

Algorithm 1: Griffin-Lim Algorithm

particularly for complex signals with intricate temporal patterns such as with music. In this research we do not focus on the quality of phase reconstruction, this is addressed in other research [36].

5 Mixsets dataset

For this research, a custom dataset is created. We drew upon a large online catalog [37] containing high-quality recordings from DJ performances (DJ sets) from which we sourced 1472 DJ sets that contained tags in a close range of specified electronic music genres (Techno, House, Electro). This dataset contains about 2500 hours of music and is stored in 320 kilobytes per second MP3 format. Each DJ set is sliced into minute-long segments. We first apply beat-finding analysis to the first segment to find the tempo and the first onset beats. After that the segment is time stretched to 130 beats per minute. Then, the segment is sliced into audio clips with 1024×1024 samples (23.77 seconds at 44.1kHz). We use a stride of 8 beats to slice consecutive audio clips. When the end of the one-minute sample is reached, we repeat the process for the next one-minute sample.

Most audio samples were slower than 130 BPM, meaning that on average we did not suffer quality loss when adjusting the speed. We also discarded samples outside the 120 and 145 bpm range to avoid unacceptable quality loss during the time-stretching. Following this method, a total of 546.256 music samples were generated.

Dataset limitations The music dataset comprises mixed music pieces sourced from individual DJ sets, reflecting the diverse and dynamic nature of real-world music performances. While this approach offers valuable insights into the practical application of music in various contexts, it does introduce certain challenges.

One important issue is the potential for incorrect beat matching due to human DJ performance errors. DJing involves seamlessly transitioning between tracks, but errors in timing or tempo adjustments can result in mismatched beats. This disrupts the overall cohesiveness of the audio compilation. As a result those artefacts could also be learned by the generator.

Another artifact commonly encountered during DJ'ing is the simultaneous peaking of audio signals within specific frequency ranges. This can lead to oversaturation of the audio signal causing distortion artifacts. The presence of such artifacts can also be learned in the trained generator. Furthermore, when mixing two music pieces together, there is a possibility of encountering mismatched musical keys. When combining pieces with disparate keys, clashes in harmonies and dissonances can arise, leading to an unpleasant auditory experience.

Regarding the audio format utilized in this study, the MP3 format was chosen due to the fact that almost all music online is in this format. However, it is essential to acknowledge that the MP3 format is lossy, meaning that it compresses audio data to reduce file size, resulting in a loss of some audio information. While this format provided a practical solution for managing large volumes of data, it may not fully capture the nuances of the original audio, potentially impacting the accuracy of analyses dependent on subtle audio features.

To address these limitations and explore potential avenues for improvement, future research could investigate alternative data collection and preparation

methods. Considering lossless audio formats, such as WAV or FLAC, could offer higher fidelity recordings for advanced analyses, albeit harder to source.

5.1 Pre processing and audio reconstruction

All audio samples were processed to a two-dimensional representation. The audio samples were transformed using Short Time Fourier Transform (STFT) with a window length of 2048 and an equal amount of frequencies, a hop length of 1025, and center-aligned padding. This resulted in a 1024×1024 two-dimensional matrix containing both frequency and phase information of the sample in a 16 bit resolution. The phase part is then discarded. We follow the strategy proposed by [2] taking the log of the power spectrogram before normalizing with a pre-calculated dataset mean and standard deviation. We clip the result between three standard deviations from the mean and rescale between 0 and 255. This leaves us with a greyscale image of size 1024×1024 (Figure 5). In this research, we have limited ourselves to mono audio. Prior to the pre-processing phase, we evaluate the dataset moments for normalization and denormalization during the pre-processing and reconstruction. To reconstruct an image we first reverse the above process. To recover the audio samples we reconstruct the phase signal using the Griffin-Lim Algorithm (GLA) [35] with 10 iterations.

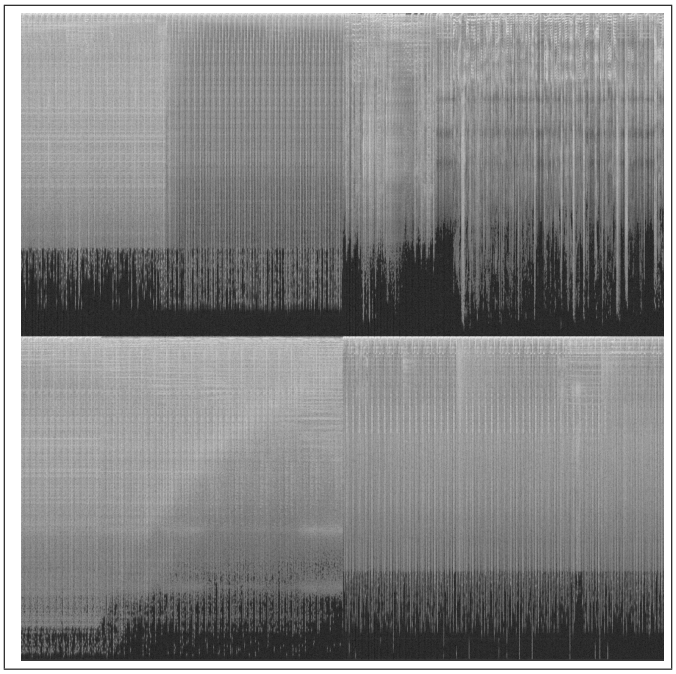


Figure 5: Short Time Fourier Representation of original dataset sample after preprocessing (2x2 images).

6 Methods

6.1 Training StyleGAN

In order to train StyleGAN for the purpose of generating audio samples we first adjust the network to output a 1024 x 1024 x 1 vector. We apply a search on the hyperparameters learning rate α and the regularization factor γ. As the input of the model we use the frequency part of the STFT representation of the sliced dataset samples. Because the models outputs those representations rather than audio samples a conversion step is performed later.

We train train the model until the FID score starts to degrade. An important indicator is that the subsequent steps show signs of mode colapse. This is apparent if the network outputs a series of images that clearly show a single skewed archetype of the data (Figure 6).

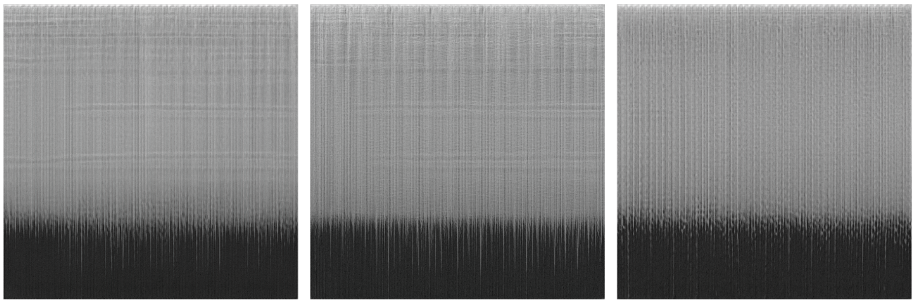


Figure 6: Example of training images that show mode collapse, the images almost look exactly the same and the network starts to degrade.

6.2 Fréchet Inception Distance

A method for measuring the similarity between real and fake images is the Fréchet Inception Distance score (FID) [38].

The metric aims to calculate the difference between two dataset samples by calculating the distance of the feature embeddings of each dataset when feeding through a classification network, the Inception V3 classification network [39].

The Inception V3 network is a state of the art image recognition model that is trained on the ImageNet dataset. The ImageNet dataset is a large-scale collection of labeled images that has been widely used as a benchmark in computer vision tasks, particularly for image classification. It contains over 1,000,000 images belonging to 1,000 different classes. Each image in the dataset is manually annotated with one of the 1,000 class labels.

Before the network is fed into a one-hot encoded feature vector of size 1×1000, the second to last layer is a vector of size $8 \times 8 \times 2024$ which is used as embedding to compare. To calculate the metric two batches of 50k images are fed through the Inception network. The second to last embeddings of the

network are then compared using the Fréchet Distance Equation 5 where μ_P and mu_Q are the mean vectors of sets V and Q. Here P represents the feature vector of the intermediate feature vector after processing images from the real dataset into the network, and Q of the generated samples respectively.

$$FID(P,Q) = \|\mu_P - \mu_Q\|_2^2 + \mathrm{Tr}\left(\Sigma_P + \Sigma_Q - 2\left(\Sigma_P\Sigma_Q\right)^{1/2}\right) \tag{5}$$

A lower distance means a higher similarity between two feature embeddings and thus indicating similarity between the original and generated images. Using the FID score to measure the distance between STFT representations of audio may appear to be counter-intuitive, the Inception network was trained for the task of classifying items on photographic images, a completely different modality than the STFT images. Using the FID score on STFT images nevertheless seems to translate into reasonable measurements when informally listening to the resulting audio examples and has been widely used in comparing generated audio samples. In this research, we also show that the FID score for audio yields similar results compared to other metrics.

6.3 Human Judgement: MOS

The primary evaluation method for this paper is judgment by humans. It is however not trivial to find sufficient and the right humans to judge the audio samples for this particular task. For judging universal audio samples such as speech, it suffices to have unfiltered survey respondents from crowdsourcing platforms such as MTurk. For validating the quality of electronic music, listeners that are at least familiar with the genre are required.

We sent out an online questionnaire to a relevant audience of approximately 25.000 electronic music festival visitors, who can be considered very familiar with the music genre. From this survey in total of 514 respondents complete the survey. We ask the human judges to determine if the audio sounds like real electronic music (Excellent, Good, Fair, Poor, Bad). With this five-point ordinal scale, we calculate a Mean Opinion Score (MOS) on the audio quality 7.

The survey consists of 20 audio examples [1]. The examples where randomly drawn from a generated batch of samples. The samples where generated with the best model configuration (Conf A) at 2200 ticks and two earlier checkpoints (1600 and 800) for comparison. Also we added the Real samples that went through the process of conversion and reconstruction using the Griffin Lim method to account for the impact on quality of those steps.

We evaluate if the changes in the FID score are also measurable with the MOS scores. We also ask on what medium the judges listen to the samples (speakers, headphones, or speakers from a laptop or phone) to evaluate if this has an effect on the outcome. It could be expected that listeners on higher fidelity audio systems or installations such as headphones or speakers.

[1] Audio samples used in Questionnaire:
https://drive.google.com/drive/folders/1dGmPBsx2QaeCOIdTkvFsESqVvFWIyQkp

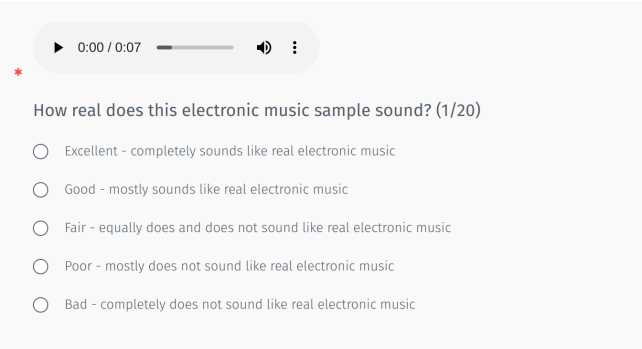


Figure 7: Example question in survey

6.4 Audio mastering

In general music productions often undergo post-production mastering treatment. Because we are curious what the best achievable level of generated audio is, we also process a small batch of generated audio samples using conventional non-synthetic mastering techniques in a conventional music studio for evaluation. In a music studio we apply EQ corrections, compressions and loudness corrections.

7 Experimental Setup

We focus on the trade-off between training stability and diversity and it's impact on quality. It is hypothesized that for each representation (audio, images etc) a different learning speed and gradient penalty is required. On the one side, we experiment with different learning rate parameters, controlling the overall speed of convergence. On the other hand, we evaluate the effects of altering the rate of gradient penalty, which is expected to have an effect on stability and diversity. We apply a grid search strategy for evaluating various hyperparameter settings. For the learning rate we evaluate $\{0.001, 0.002\}$ and for the regularization constant γ we evaluate $\{1, 5, 10\}$.

We furthermore do an investigation on the quality of the samples generated from the model configuration (Configuration A) with the lowest found FID score. We compare the best trained network snapshot with two earlier snapshots with lower FID scores to compare if the MOS score changes with a similar effect with the FID score. We also apply mastering techniques on a selection of samples created by the best-performing model and aquire those MOS scores. Because the pre-processing steps also has some lossy elements in it, we evaluate the MOS scores from original data set that went through the preprocessing steps. For each of the generated samples, we hand-picked 4 best samples out of a set of 20 samples. We strongly encourage the reader to listen to the used audio clips [2].

A single step in training is one *kimg*, meaning that a thousand real samples have been processed by the network.

Computational costs of the complete research including exploratory work are 2554 hours on a RTX3090 GPU. To train the best hyperparameter configuration to the best achievable score the model was trained for 137 hours.

[2] Audio samples from research: https://soundcloud.com/havenaarbd/sets/electrogan-towards-high-fidelity-electronic-music-generation-with-stylegan2

8 Results

The different hyperparameter settings for the learning rates and γ have a significant influence on the best achievable FID score and the number of processed kimg's possible until the networks started to degrade (Figure 1). We see some indication of a trade-off between a low learning rate and high γ and vice versa. This too seems a logical consequence of the higher learning rate allowing for more rigorous adjustments to the networks, while a higher γ allows more diversity.

Configuration		Best FID score (lower is better)	
Learning Rate	γ	FID	kimg
0.001	**10**	**3.21**	**2200**
0.001	5	7.57	7200
0.001	1	13.06	2600
0.002	10	40.5	300
0.002	5	48.1	200
0.002	1	7.39	2000

Table 1: FID scores for various training runs with Learning Rate $\in \{0.001, 0.002\}$, $\gamma \in \{1, 5, 10\}$

The winning configuration is a learning rate of 0.001 and a γ of 10. This yields an FID score of 3.21. Notably this is the same exact the same configuration and approximate FID score achieved for generating faces in [34]. A big difference is that convergence seemingly happens an order of magnitude faster in this case, for audio. Inference speed of generating STFT images and the post processing pipeline (normalization, GLA) cost 1.01 seconds per 23.77 seconds of audio. After 2200 kimg's this network does not improve any further and starts to gradually degrade (Figure 8). We see this happening when the Wasserstein loss for the Generator and the Discriminator start to converge and cross. Also this is the same point at which the gradient penalty starts to overcompensate (Figure 9). This behaviour is expected when the generator starts outputting very similar images (mode colapse) and the discriminator start to easily recognize fake images.

The best configuration generates samples with a MOS score of 2.46 (Table 2) with a standard deviation of 1.02 and a FID score of 3.33 (Figure 10). The audio samples that where post-processed using conventional studio mastering techniques achieved a MOS score of 2.87. The original audio samples that went through the pre- and post processing pipeline achieved a MOS score of 3.94 with a standard deviation of 0.92. The human judges mostly use headphones 49.6% or speakers 19.8%. The remaining judges use the sound of their laptop or phone 30.5%. We did not see significant difference in MOS scores across the different used listening media devices.

Visually the generated samples are hard to distinguish from the original

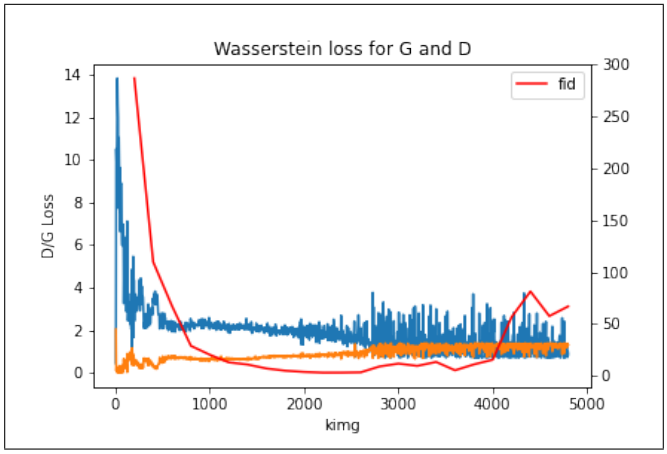


Figure 8: Training run with LR = 0.001, $\gamma = 10$. Wasserstein loss for generator (orange) and discriminator (blue) with FID score (red). The best result was achieved after 2200 Kimg after which the network deteriorated

dataset. When informally listening to the samples we hear clear features of the electronic music such as the 4/4 rhythm of bass drums and percussion such as hi-hats. This also comes with a noisy background and disordered phases. It sounds as if the generated samples lack silence in any frequency, making the audio sound very crowded across the complete spectrum. After professionally mastering the audio samples in a music production studio with the use of common mastering techniques the samples become much easier to listen to. Also, we sporadically recognize archetypes in generated audio samples. These archetypes share similar sound features such as a certain percussive line or other musical features. This is not the same as the mode collapse described earlier because this is also observed while the model is still converging.

Dataset / configuration	**MOS**		**FID**
	mean	std.	
Real*	3.94	+/- 0.92	-
Mastered	2.87	+/- 1.02	-
Conf A - 2200 ticks	2.46	+/- 1.02	3.33
Conf A - 1600 ticks	2.24	+/- 1.05	7.35
Conf A - 800 ticks	1.94	+/- 0.97	29.04

Table 2: MOS score for real audio, mastered audio, and best-performing configuration (LR = 0.001, $\gamma = 10$) with FID scores for various snapshots. *Real samples include conversion to STFT and Griffin Lim audio reconstruction as described in section 5.

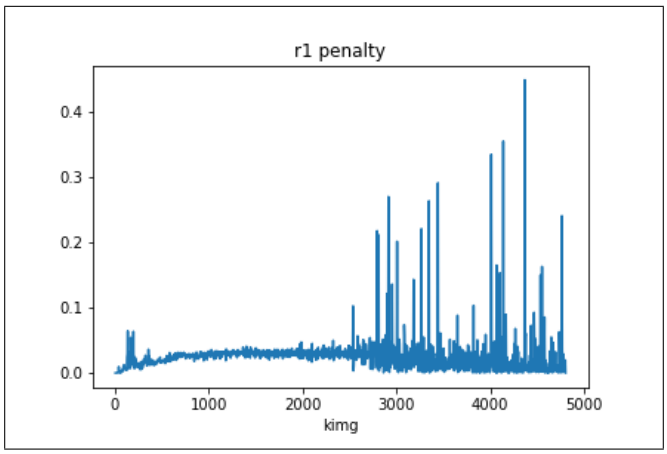


Figure 9: After convergence at 2200 steps the model kept stable until approximately 2750 steps. Hereafter the gradient penalty significantly increases as mode collapse occurs

9 Conclusion and discussion

Generating long, complex, and comprehensive electronic music samples using the methods described in this paper is a promising direction. The generated music samples clearly show the structure of electronic music, are tens of seconds long, and have a sampling rate with sufficient resolving power to accommodate high-quality audio.

We have shown that it is feasible to achieve similar FID scores with audio as with the original StyleGAN2 experiments on faces. For evaluating the quality of the audio both MOS score and the FID score move in the same direction. At the same time, the quality of the audio is still not close to what sounds like real electronic music, while the FID scores indicate almost perfect similarity between the generated samples and the real dataset. The relatively low scores (MOS 3.94) of the real audio samples that have been subject to pre-processing and phase recovery (Table 2) also indicate that there is a lot of room for improvement in the phase reconstruction part. Furthermore, it is not clear why some generated samples tend to linger toward certain archetypes.

The choice to work with extra large networks with 1024×1024 output could be debated. The increased amounts of parameters and time to train such a network might not be necessary if one would accept shorter audio samples. For a 512×512 STFT this could provide 5.9 seconds of (possibly looped) audio compared to 23.77 seconds. Using a smaller network may also have effects on the stability of the training. This would be interesting for further research.

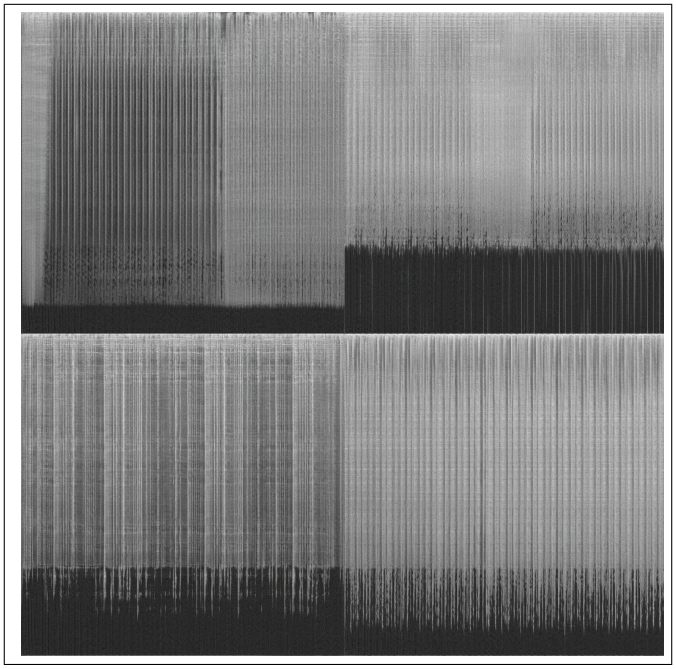


Figure 10: STFT results from StyleGAN 2 trained model with LR = 0.001, γ = 10 with 2200 training steps (2x2 images).

10 Acknowledgments

I want to express my gratitude to my supervisor, Erwin Bakker, whose invaluable guidance and patience has helped me shape this long lasting research. Also I want to thank my family for their support along the road. I extend my appreciation to the Investment in Culture Fund of the Municipality of The Hague for co-funding the hardware that significantly enhanced the execution of this experimental research.

References

[1] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: modelling raw audio at scale," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[2] S. Dieleman, "Generating music in the waveform domain," 2020. Blog article: https://benanne.github.io/2020/03/24/audio-generation.html, last time visited 30/07/2023.

[3] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," *arXiv preprint arXiv:1604.08723*, 2016.

[4] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*, pp. 4364–4373, PMLR, 2018.

[5] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, p. 125, 2016.

[6] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *International conference on machine learning*, pp. 3918–3926, PMLR, 2018.

[7] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 2410–2419, PMLR, 10–15 Jul 2018.

[8] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *Conference on Learning Representations*, vol. abs/1612.07837, 2016.

[9] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2018.

[10] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[11] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

[12] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," vol. 32, Advances in neural information processing systems, 10 2019.

[13] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2019.

[14] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2019.

[15] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of styleGAN," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116, IEEE, 2020.

[16] T.-M. Hung, B.-Y. Chen, Y.-T. Yeh, and Y.-H. Yang, "A benchmarking initiative for audio-domain music generation using the freesound loop dataset," International Society for Music Information Retrieval, 08 2021.

[17] J.-Y. Liu, Y.-H. Chen, Y.-C. Yeh, and Y.-H. Yang, "Unconditional audio generation with generative adversarial networks and cycle regularization," *Interspeech*, pp. 1997–2001, 2020.

[18] P. Dhariwal, H. Jun, C. Payne, J. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," 04 2020.

[19] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[20] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in neural information processing systems*, vol. 32, 2019.

[21] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2018.

[22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners." OpenAI blog.

[23] S. Pasini, "Musika! fast infinite waveform music generation," *Proceedings for ISMIR 2022*, 2022.

[24] S. Forsgren and H. Martiros, "Riffusion - Stable diffusion for real-time music generation," 2022. blog ariticle: https://riffusion. com/about 6. Last visited 30/07/2023.

[25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] J. Antolik and J. A. Bednar, "Development of maps of simple and complex cells in the primary visual cortex," *Frontiers in computational neuroscience*, vol. 5, p. 17, 2011.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[29] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[30] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing training of generative adversarial networks through regularization," *Advances in neural information processing systems*, vol. 30, 2017.

[31] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?," in *International conference on machine learning*, pp. 3481–3490, PMLR, 2018.

[32] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[33] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[34] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[35] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[36] Y. Masuyama, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada, "Deep griffin–lim iteration," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 61–65, IEEE, 2019.

[37] M. D. Website, "Electronic music mixes dataset." Website, 2023. Accessed on August 15, 2023.

[38] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.