Audio Generation using GAN

Frida Sjögren
M.Sc. Data Science & AI
DAT341, Group 51
sjogrenf@chalmers.se

Johanna Norell
M.Sc. Data Science & AI
DAT341, Group 51
jnorell@chalmers.se

Abstract

This project explores the use of generative adversarial networks (GANs) for audio generation, limited to the genre "classical". Despite extensive model adjustments and training strategies, the GAN model struggled to generate coherent audio, producing only random noise. While a predefined WaveGAN architecture demonstrated some promise, further progress was limited by the computational constraints. The project concludes that generating audio from raw files presents significant challenges, and future experiments may benefit from using alternative audio formats, such as MIDI.

1 Introduction

Deep neural networks are a subset of machine learning designed to identify complex, non-linear patterns. Their power lies in uncovering subtle features in data that are difficult to detect or predict using human perception alone. In the context of music generation, deep neural networks present an exciting opportunity to train models capable of replicating audio across various genres. Unlike humans, who can easily classify music into genres such as rock, hip hop, or pop without expert knowledge, defining precise rules for these genres without a deep understanding of music theory is challenging. Thus, it is fascinating to explore how well a machine learning model can learn genre-defining features purely from data. Taking this further, generative models offer the possibility of producing entirely new audio samples in these genres.

With the recent advancements in generative models, such as Generative Pre-trained Transformers (GPTs), the role of machine learning in creative fields has rapidly expanded. Although audio generators may not become as widely adopted as text-based models, they could prove highly valuable in several settings. For instance, in professional environments, they can inspire music producers by generating novel ideas or genre-blending sounds. Additionally, these models could be used in interactive applications such as personalized soundtracks for video games or tailored audio experiences in virtual environments.

2 Background & Related Work

Machine learning has long been employed by musicians in various ways, from recognizing and identifying musical patterns to discovering new sounds and relationships that may not be obvious, even to experienced musicians. One key advantage of using machine learning in this context is the ability to bypass the need for explicitly defined rules, which can often be impractical or time-consuming to develop (Fiebrink and Caramiaux, 2016).

Architectures like WaveNet and Anticipation-RNN have been applied in music and melody generation (Briot and Caramiaux, 2018). GAN models are favorable in audio generation as they manage to generate audio with both local and global consistency (Bastianello, 2023). Additionally, GAN models offer efficient training and generation due to their ability to process data in parallel. However, as Bastianello points out, there are challenges associated with the periodic nature of waveforms, which can lead to training instability.

3 Method

This section will give details about the solution used in our specific project.

3.1 Preparing the Data

The data was sourced from Kaggle[1] and contains 1,000 samples across 10 classes. Initially, we

[1] https://www.kaggle.com/code/basu369victor/generate-music-with-variational-autoencoder/notebook/

aimed to train a model to generate audio based on genre, but due to complexity, we limited the project to one genre. The genre "classical" was chosen as it contained no vocals. Given the small dataset and the subjective evaluation of GANs, we opted to train on the entire dataset without using validation data, prioritizing the quality of generated samples over validation metrics.Since validation metrics are less effective for GANs, which are typically assessed by sample quality, we opted to use the entire dataset for training. This decision was also driven by the limited dataset size, making it more beneficial to maximize training data.

Following, the waveform data was converted into a floating point time series as well as normalized to have mean 0 and standard deviation 1.

3.2 Establishing a Baseline

Establishing a baseline performance was challenging, as there are no direct comparisons to human accuracy and we found no existing tools performing similar tasks. For converting audio files into features that a model can learn, Hugging Face has developed a classification model on the same dataset using the pre-trained DistilHuBERT[2] model, which achieves an accuracy of 82% in correctly classifying the genre of the audio files. When examining other baseline performances of generative models for audio generation, even simpler audio generation tasks have proven to be quite difficult.

3.3 Generative Adversarial Networks (GANs)

The design of GAN models consists of two main components: the generator and the discriminator. The generator's role is to transform random noise into the desired output. The discriminator's task is to classify the generator's output as real or fake, thus it can simply be seen as a classifier. During training, the interplay between these two components refines and improves the model. This architecture is illustrated in Figure 1.

In the context of audio generation, the generator transforms random noise into an audio sample and is trained over multiple epochs to progressively enhance the quality of the generated audio. Meanwhile, the discriminator is trained to improve its ability to distinguish between real and fake audio samples. Both models are optimized to minimize their respective losses.

[2]https://huggingface.co/learn/audio-course/en/chapter4/fine-tuningfine-tuning-the-model

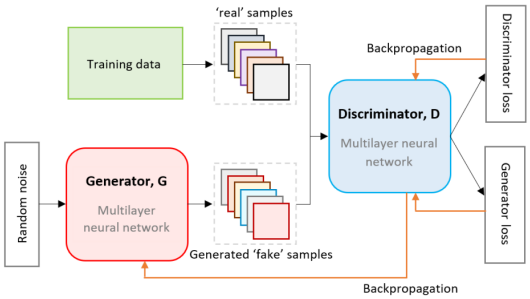

Figure 1: GAN model architecture, Source: (Little et al., 2021)

3.3.1 Training

Training a GAN model is challenging because of the simultaneous training of the generator and the discriminator. While training the discriminator is relatively straightforward, because of it functioning like a classifier, the interplay between both models complicates the process. The discriminator's task is to classify real data as real and fake data as fake, summing the two losses for back-propagation. Its training is influenced by the quality of the fake data generated, but is otherwise independent of the generator's loss.

The generator, however, relies on feedback from the discriminator, using its classification of fake outputs to calculate its own loss and back-propagate. This creates a dependency where the generator's improvement depends on the discriminator's performance. Although back-propagation passes through both networks, only the generator's weights are updated using the gradients in this training phase.

Training typically alternates between the discriminator and generator over multiple epochs, making convergence difficult. Initially, the generator improves as it learns from the discriminator, but as the generator gets better, the discriminator's accuracy decreases, making its feedback less valuable. Ideally, the discriminator should reach 50% accuracy, signaling that it can no longer distinguish real from fake data. However, continuing to train the generator beyond this point often leads to performance degradation, as the feedback becomes less reliable. Hence, convergence in GAN models is unstable rather than consistent.

3.3.2 Loss function

GAN models aim to replicate the distribution of real data when generating new outputs, so the loss should reflect the distance between the generated and real data. While separate loss functions can be used for the generator and discriminator, a common choice is the Wasserstein Loss, which derives from a single distance. With this loss, the discriminator's output is modified: instead of predicting probabilities for "real" or "fake," it outputs higher values for real instances and lower values for fake ones. Both models try to maximize the losses as follows:

Discriminator loss: $D(x) - D(G(z))$
Generator loss: $D(G(z))$

$D(x)$: discriminator output for a real instance
$G(z)$: generator output given noise z
$D(G(z))$: discrimintor output of fake instance

3.4 Using the existing WaveGAN architecture

To establish a baseline performance to compare our custom-built GAN model with, we used a pre-defined architecture called WaveGAN (Donahue et al., 2019). This architecure has been adopted in several contexts when using GAN models in audio applications. While we aimed to use a pre-trained version and do only fine-tuning of the model, we found no pre-trained weights.

The pre-defined WaveGAN is a GAN architecture capable of generating up to 4 seconds of audio at a sampling rate of 16kHz. The discriminator comprises multiple convolutional layers followed by a fully connected layer, with no activation functions. The generator consists of a fully connected layer followed by several transposed convolutional layers. ReLU is used as the activation function between the transposed convolutions, while the output layer uses the tanh activation function. The Adam optimizer and the Wasserstein loss are used. The full specification of the architecture can be found in the git-repo.[3]

3.4.1 Training a Custom GAN Model

Lastly we developed a GAN model from scratch, using inspiration from the architectures we had already seen.

Given the complexity of generating fake audio, we anticipated runtime to be extensive. To address this, we took an iterative approach in building the architecture, starting with a simple generator and discriminator, both consisting of two fully connected layers each, ReLU activation between layers and sigmoid activation for the output of the discriminator. While the model successfully trained, the output was still just random noise.

In an effort to improve performance, we made incremental adjustments to various parameters, as described in Table 1 under attachments. Despite these extensive changes and guidance from our supervisor, we encountered significant difficulties in enhancing the model's performance. After all incremental adjustments we still did not manage to produce anything else the random noise. The full architecture of the final, still not functional, GAN model are found in Figure 3 and Figure 4 under attachments.

3.5 Evaluation & Result

While the initial ambition was to evaluate the model's performance through both a structured, subjective assessment by humans as well as by a quantitative score, this approach became irrelevant when the output was merely noise. Due to the lack of meaningful output, the focus of this project has remained on improving the model itself, rather than exploring methods to evaluate the output. In addition to subjectively evaluating audio output, we closely monitored loss evolvement throughout training. With the final architecture trained over 200 epochs, a slight decrease in both losses was achieved as visulized in figure 2, still clearly indicating model convergence was not achieved.

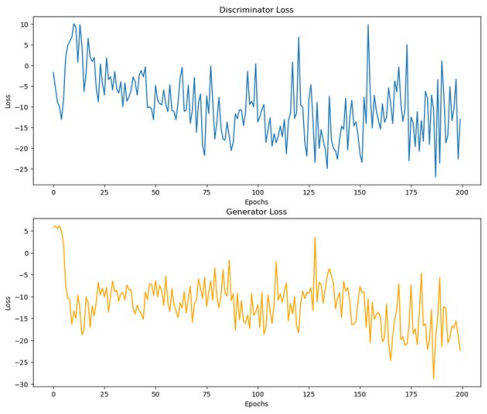

Figure 2: Discriminator and Generator loss during the final training of 200 epochs.

On a theoretical note, Donahue et al. (2019) describe the process evaluating samples using quantitative metrics as challenging, as these measure-

[3] https://github.com/mostafaelaraby/wavegan-pytorch/tree/master

ments may not align with human perception. To address this, they combined an *Inception Score*, using a pre-trained Inception classifier to measure diversity and semantic discriminability, along with human judgment by answering three specific questions in the context of their WaveGAN audio generator. A similar approach would have been appropriate if the model's output had been more refined.

4 Discussion and Possible Future Improvements

A significant challenge faced during this project was the generator's failure to produce distinguishable audio from the random noise input. This highlights the special dynamics between the discriminator and generator requiring balance to achieve the model's objectives, as well as a key issue in the generator's learning process, where it struggled to capture patterns in the audio data.

Several factors may explain the generator's lack of progress. A primary reason could be an overly strong discriminator, leading to weak or vanishing gradients that hinders the generator's learning. To counter this, we explored using Wasserstein loss, which provides meaningful gradients even with a strong discriminator. Additionally, we trained the generator twice as often as the discriminator and applied a higher learning rate to boost its learning. However, these adjustments did not improve generator performance.

Ultimately, it is hard to make sure GAN models converge and in some sense hard to determine when training is done and peak performance is reached. As mentioned by Donahue et al. (2019) their WaveGAN model converged after four days of training. The scope and duration of this project might simply not have been enough to train the model to convergence.

Another reason for the unsatisfactory results may be the limited training data. Initially, the dataset contained 1,000 samples of 30-second audio clips, but focusing on a specific genre reduced it to 100 samples. To address this, we segmented the 30-second clips into 700 shorter, 4-second samples. Despite this increase, the dataset size may still be insufficient. Future improvements could include applying sliding windows on existing data to create more samples or gathering additional, suitable data for the generator to learn from.

Given these limitations, we explored pre-trained GAN models, with MuseGAN as a promising option. However, MuseGAN operates on MIDI data, which differs from raw audio. Unlike raw audio, which captures detailed waveforms, MIDI files act like a musical score, storing only instructions for notes, duration, tempo, pitch, and instrument type, not the sound itself.

Converting raw audio to MIDI is complex, as it requires distinguishing and transcribing multiple instruments, vocals, and sounds into a format that MIDI can represent. This made it too challenging to use MuseGAN for our project. In hindsight, choosing a dataset in a format like MIDI, where distinct musical features have already been separated from the audio, could have made the task more manageable.

During our search for pre-trained models, most of them operated on formats simpler than raw audio and achieved better results. In retrospect, starting with a simpler format could have been a more practical approach.

5 Conclusion

The initial purpose of this project, to be able to generate genre specific audio was quickly reconsidered once the difficulty of audio generation with GAN models became apparent. As a result, the focus shifted to producing any form of audio using the GAN model built from scratch. Despite extensive iterations of architectural adjustments, training strategies, and pre-processing changes, the results remained unsatisfactory, with the generator failing to produce coherent audio outputs distinguishable from noise.

A pre-defined WaveGAN model was tried out and an output different from mere noise was obtained for the first time. However, it was later discovered that the model required huge computational resources and time for convergence. Thus, on the search for a pre-trained model the MuseGAN was found, requiring a different data format MIDI. With all the experience in mind, we would have tried another data set with a different representation of the data than raw audio. A final reflection for future improvements is the desire of a larger data set to evaluate GAN models for audio generation further.

References

Eduardo Bastianello. 2023. Sound generation using gan models. https://thesis.unipd.it/retrieve/

365a3a3b-c7ce-482d-8ffd-487eb71fcf78/Bastianello_Edoardo.pdf. Accessed: 2024-10-16.

Jean-Pierre Briot and Baptiste Caramiaux. 2018. Music generation by deep learning – challenges and directions. https://arxiv.org/abs/1712.04371. Accessed: 2024-10-16.

Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial audio synthesis. https://arxiv.org/pdf/1802.04208v3. Accessed: 2024-10-22.

Rebecca Fiebrink and Baptiste Caramiaux. 2016. The machine learning algorithm as creative musical tool. https://arxiv.org/pdf/1611.00379. Accessed: 2024-10-16.

Claire Little, Mark Elliot, Richard Allmendinger, and Sahel Shariati Samani. 2021. Generative adversarial networks for synthetic data generation: A comparative study. https://arxiv.org/abs/2112.01925. Accessed: 2024-10-18.

6 Attachments

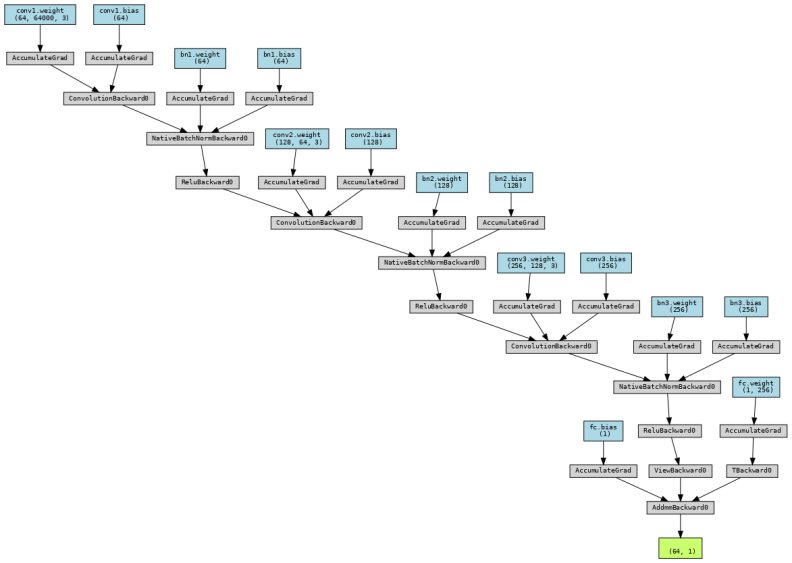

Figure 3: Detailed Discriminator Architecture

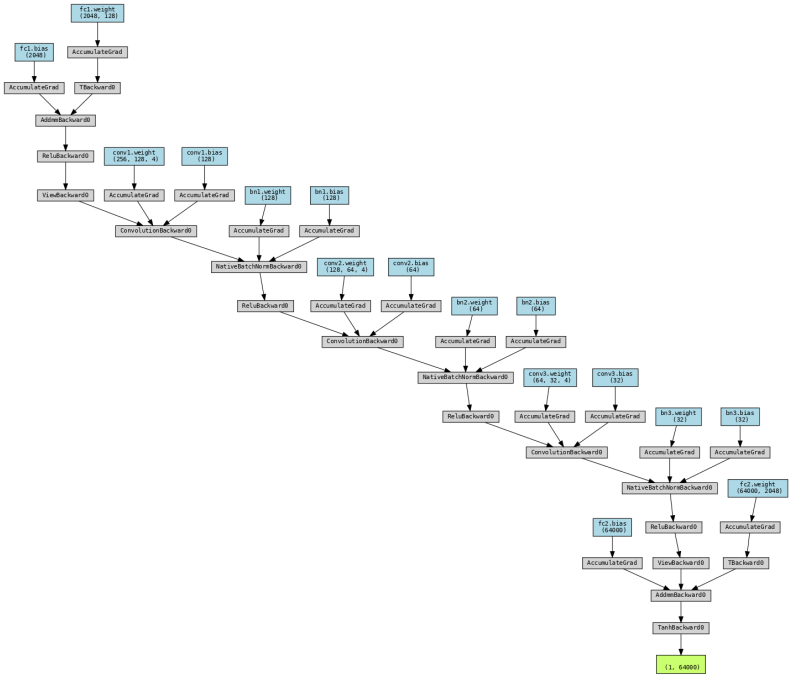

Figure 4: Detailed Generator Architecture

Change	Specification	Aim	Result
Pre-processing	Change duration of audio samples from 30 to 5 seconds	Simplify the training process by reducing the complexity of the data and reduce runtime	Decrese in runtime
Pre-processing	Change from 10 genres to one (classical)	Simplify the training process by reducing the complexity of the data	No notable improvements in losses or generated audio, decrease in runtime
Pre-processing	Split training original training samples into 4 second samples	Obtain more training data	Went from 100 to 700 samples, enabled larger batch sizes, helped stabilizing training
Architectural	Add more hidden units	Increase model complexity to capture patterns in data	No notable improvements in losses or generated audio, increased runtime
Architectural	Add deconvolutional layers in the generator	Improve the quality and structure of the generated data	No notable improvements in losses or generated audio
Architectural	Add convolutional layers in the discriminator	Improving feature extraction	No notable improvements in losses or generated audio
Architectural	Add batch norm between all layers	Stabilize training	No notable improvements in losses or generated audio
Training	Introducing Wasserstein loss	Stabilize training	No notable improvements in losses or generated audio
Training	Train generator twice (or more) as often as the discriminator	Prevent generator loss to increase	Balanced discriminator and generator loss
Training	Introduced separate learning rates for discriminator and generator	Address the issue of increasing generator loss	Prevented increase of generator loss, leading to a more stable training
Training	Increasing latent dimensions	To consider if input dimensions have an effect on the capability of the generator to produce audio	Increased runtime, no notable improvements in losses or generated audio
Training	Add smoothing of labels	For the discriminator to not be overly confident	Balanced discriminator and generator loss, but lead to less decrease in losses overall

Table 1: Changes in data set and GAN model