An Exploration of Audio Generation and Real-time Latent Space Manipulation with Realtime Audio Variational Autoencoder (RAVE)

Irene Jiaying Xu
23001934
14 Mar 2024

1 Introduction

From the early natural language processing program ‘ELIZA’ in the 1960s to the most recent breakout text-to-video generator ‘Sora’, Generative AI has evolved into one of the most disruptive technological innovations (White, 2023), capable of handling a wide array of learning tasks such as speech recognition, media generation, and much beyond. This project sets out to explore the area of unsupervised audio generation with a special focus on the Real-time Audio Variational Autoencoder (RAVE), which was introduced by Caillon and Esling in 2021. The primary goal is to undertake a comprehensive study on the process of training RAVE models from scratch, as well as to contribute a timely review of the model and its effectiveness in producing fast and high-quality natural audio synthesis (Caillon and Esling, 2021). Based on the trained models, the exploration will further entail experimenting with the generated synthesis through methods such as unconditional generation, latent manipulation, timbre transfer, real-time audio processing, and audio-reactive drawing. The aim is to produce appealing audio and visual results, thereby providing a valuable reference for practitioners interested in exploring the creative use of VAEs in musical applications.

2 Background

Similar to many other areas of Generative AI, a lot of deep-learning models have been applied in the realm of sound production. For instance, WaveNet and Tacotron 2 offer realistic and natural text-to-speech synthesis (Lux and Vu, 2022), whereas GANSynth provides a way to generate high-fidelity and locally coherent audio (Engel et al., 2019). Variational Autoencoders (VAEs), as the focus of this project, also play a significant role in audio generation by providing a flexible approach to creating and manipulating sounds. Operating within a probabilistic framework, VAEs feature continuous latent spaces that allow easy sampling and interpolation (Irhum Shafkat, 2018), which is particularly valuable in creative audio applications where nuanced modification to data is desired. By manipulating the latent space, the characteristics of the generated audio, such as timbre or pitch, can be easily altered, providing a high level of control over the audio generation process.

Building on the VAE framework, RAVE employs a unique two-stage training procedure: *representation learning* and *adversarial fine-tuning*, where the model is first trained as a conventional VAE, and then only the decoder undergoes training with an adversarial objective to improve the audio quality and naturalness (Caillon and Esling, 2021) using a multiscale discriminator. A singular value decomposition is also used to identify the most informative parts of the latent space to allow easier exploration and manipulation of latent trajectories (Caillon and Esling, 2021). Compared to regular VAEs, RAVE's architecture substantially increases synthesis speed, making it suitable for creating real-time interactive applications and even for live performance.

3 Method

Based on the official implementation documentation[1], this section aims to provide a more detailed methodology and best practices for training a RAVE model. Detailed instructions and code are available on GitHub[2] and can be used directly to train a model on a new dataset.

3.1 Datasets

Dataset 1 – Solo Piano. A homogeneous audio dataset that comprises approximately 6 hours of solo piano music, mostly consisting of classical piano pieces from renowned pianists such as Ludwig van Beethoven and Johann Sebastian Bach, along with piano songs that are copyright-free from websites such as musopen.org, chosic.cm, poxabay.com and soundcloud.com.

Dataset 2 – Techno Music. A music audio dataset contains approximately 6.5 hours of copyright-free Techno/EBM/Cyberpunk music by Aim To Head[3]. It is homogenous in terms of music genre but extremely heterogeneous in terms of music objects texture (Roads, 2015), as this genre of music often uses a lot of different instruments and heavy use of synthesisers.

3.1.1 Datasets Collection

All audio files were obtained online and are copyright-free. All audio files use the same format (.mp3), allowing them to be directly merged into one, although using other formats such as WAV and OGG is also acceptable and can be trained directly without converting.

3.1.2 Dataset Preprocessing

[1] Official implementation of the RAVE model, GitHub. Available at: https://github.com/acids-ircam/RAVE
[2] Project Repository, GitHub. Available at: https://git.arts.ac.uk/23001934/AI-4-Media-Rave-Audio-Generation
[3] Aim To Head, an electronic music production / promotion that provides copyright free music for creators.

The audio files from each dataset were merged into a single MP3 file, with the sample rate explicitly set to 44,100 Hz, the number of channels to 1, and the bit depth to 16 bits. The amplitude has also been normalised to a range of -1 to 1, as RAVE will automatically clip the audio if it has an amplitude outside this range, which might result in a loss of information.

The following charts (Fig1, Fig2) present random 20-second audio samples from the processed audio, illustrating the distinct characteristics between the two datasets. Samples of the audio can be found on GitHub. The piano waveform exhibits clear peaks and troughs, and most individual notes can be easily distinguished with a clear starting point and decay pattern. In contrast, the techno music that uses lots of synthesised sounds displays a denser frequency spectrum. Its waveform is more consistent with less dynamic range, and there are fewer silent intervals between notes, making it harder to distinguish individual notes or frequencies.

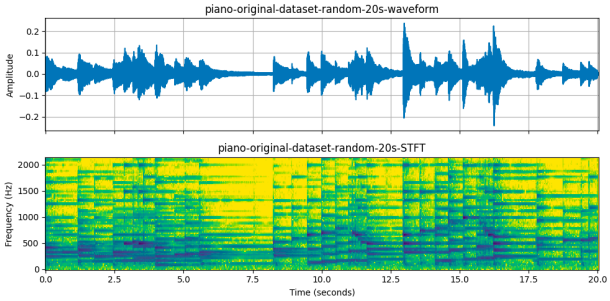


Figure 1 piano-dataset-waveform-STFT

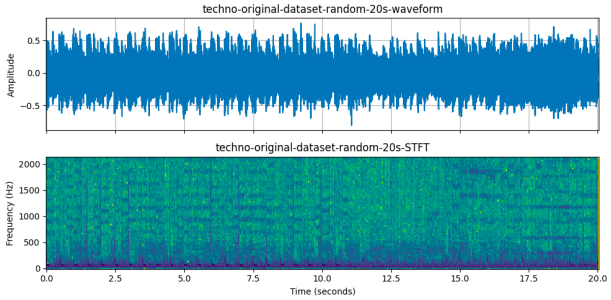


Figure 2 techno-dataset-waveform-STFT

3.2 RAVE

3.2.1 Dataset Preparation

After preprocessing, the audio files should be prepared for training by using the following command, specifying key parameters such as the location of the audio dataset input, the destination location of the processed audio output, and the number of channels of the audio:

```
rave preprocess --input_path /audio/folder --output_path /dataset/path --channels X
```

Note that processing multichannel audio can significantly increase training time, e.g., training a stereo track takes double the time as training a mono track. Although RAVE can train stereo or a higher number of channels, using mono (1 channel) is recommended for compatibility with the current versions of RAVE, which has issues processing multichannel audio and may not successfully export a stereo version, leading to unnecessary waste of training time.

3.2.2 Train

```
rave train --config $config --db_path /dataset/path --out_path /model/out --name
$name_of_your_model --channels X
```

Apart from the key parameters included in the above training command, optional parameters like batch size, the number of workers to spawn, which GPU to use, and checkpoint saving frequency can also be specified. In my case, I have used the configuration v2 and explicitly set the GPU, as sometimes the model fails to automatically select the correct GPU.

In terms of batch size, simply increasing the size does not necessarily increase the training efficiency and model performance, which is attributed to RAVE's unique two-stage training procedure. The first stage focuses on learning a general representation of the data, in which smaller batches may be beneficial to reduce overfitting risk, then in the adversarial fine-tuning stage that aims to improve audio quality, larger batches may be more beneficial for accelerating training speed and improve model performance. In addition, as soon as the training enters stage 2, the GPU usage immediately doubles compared to stage 1 (e.g., 5.4/16.0GB to 12.7/16.0GB), and the training speed also decreases significantly. Therefore, it is not recommended to increase the batch size to the maximum the GPU can handle in stage 1, and a careful experiment with the batch size is needed to find a balance between the generalisation ability and training speed. In my case, I have proceeded with the default setting of 8 as many

RAVE training examples and discussion posts on Discord[4] and GitHub[5] suggested it ensures a rather good stability and training speed overall.

Generally, training will automatically progress to stage 2 after about 1 million steps, according to many discussions on Discord. The exact transition can be monitored using TensorBoard. On an Nvidia GeForce RTX 4080, training a 6-hour long dataset takes about 35 hours to reach 1 million steps, and 100 hours in total to reach 2 million steps. Since stage 2 is dedicated to enhancing the audio quality, it is recommended to train for at least 1.5 to 2 million steps to achieve a reasonably trained model, and 3 million steps for better results. However, the effectiveness of training can vary based on the dataset, a homogeneous dataset might yield satisfactory results at an earlier stage, whereas a more diverse dataset requires much longer training time to adequately learn all the features.

3.2.3 Export

Once trained, the model can be exported using the following command. Different checkpoints can export various models, allowing for comparisons at different training stages.

```
rave export --run /path/to/your/run (--streaming)
```

4 Results

The final results are based on about 100 hours of training for each dataset, amounting to 200 hours in total, with each dataset reaching approximately 2 million steps. Training metrics and scalars from TensorBoard are included in the appendix.

[4] References from RAVE official Discord Server (comments from users: IDDQD, ballerburg9005, Echevarian etc.)
[5] Discussion post on GitHub, Available at: https://github.com/acids-ircam/RAVE/issues/22

4.1 Unconditional Audio Generation

Based on the following waveforms and spectrograms (Fig 3, Fig 4), both models have successfully synthesised audios that share similar characteristics with the training audio files.

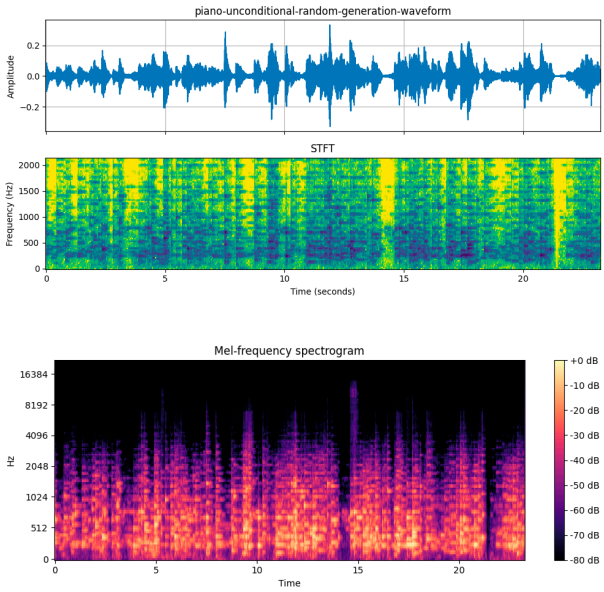


Figure 3 piano-random-generation

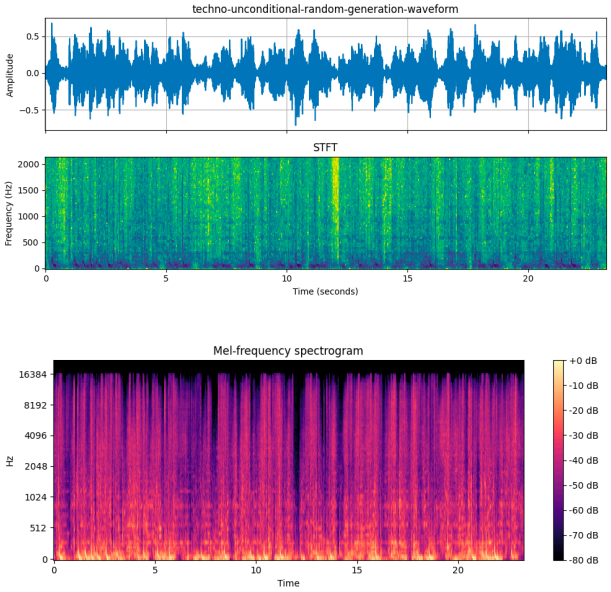


Figure 4 techno-random-generation

The solo piano model, benefiting from its homogeneous training dataset, seems to perform better at sound reconstruction. It also produced piano-like sounds at an earlier stage, even before reaching 1 million steps. In contrast, the techno music model took longer, as the dataset contains much more diverse sound objects. Nevertheless, the performance of both models appeared to improve dramatically in stage 2, culminating in more realistic and less noise-like audio output.

4.2 Timbre Transfer

In terms of performing timbre transfer tasks, the quality of generated audio seems to depend heavily on the similarities between the source audio and the training audio dataset. For instance, the solo piano model was able to perform timbre transfer on xylophone or solo guitar instrumental pieces effectively, as the sounds of those instruments share a similar characteristic

to that of the piano. However, it fails to produce meaningful transfer for the sound that is completely different from the sound in the original dataset, such as human voice, sometimes leaving large blocks of silence or pure noise sound in the transferred result.

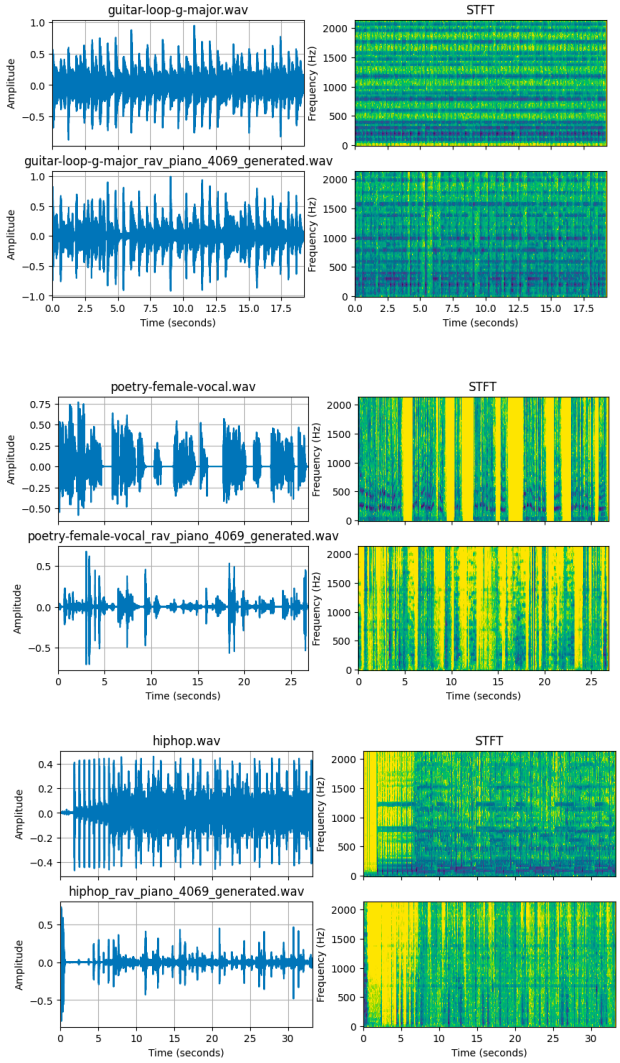


Figure 5 piano-timbre-transfer on guitar, voice, and hip-hop

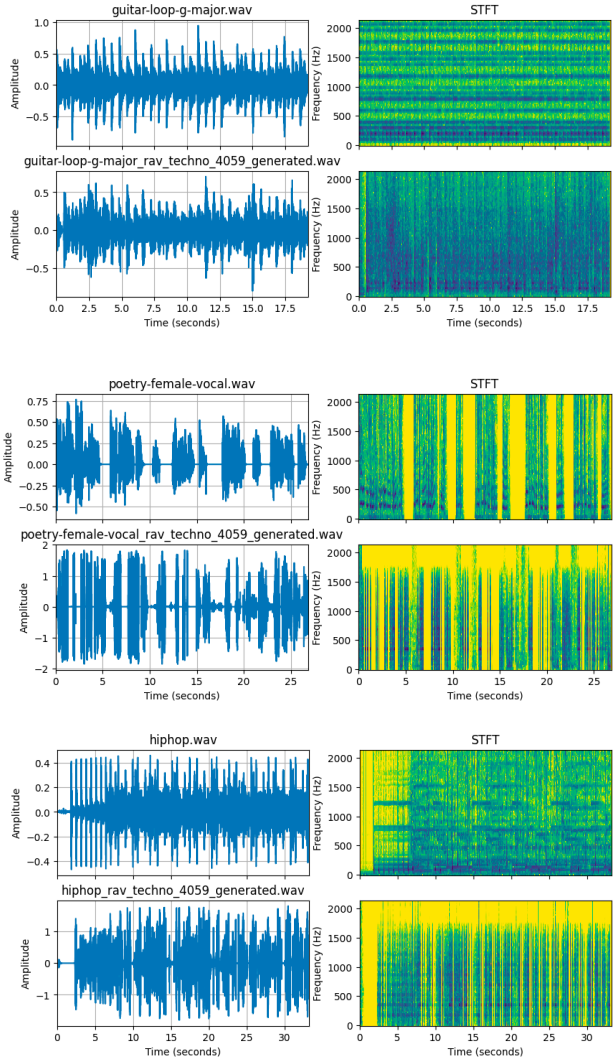


Figure 6 techno-timbre-transfer on guitar, voice, and hip-hop

The techno music model, which incorporates multiple timbres in its training dataset, managed to perform timbre transfer on a slightly wider variety of sounds. However, the quality of the audio was not as satisfactory as the solo piano model, many times resulting in almost noise-like output. This issue might arise because a heterogeneous dataset requires much longer

training time than a homogeneous dataset, and 2 million steps may not be sufficient for it to train a model capable of performing timbre transfer effectively.

Both models are more effective at performing such tasks on sounds that have a single audio source or distinguishable features in terms of pitch, timbre, and rhythm, such as solo instrumental music. This contrasts with synthesised audio or sounds with mixed sound objects, which often involve more complex sound layering and therefore, are naturally more challenging for the model to apply the timbre characteristics.

4.4 Latent Space Manipulation

Since RAVE allows easier exploration and manipulation of latent space (Caillon and Esling, 2021), it provides an opportunity to improve the quality of synthesised audio and to create novel sounds. Max/MSP is utilised here to achieve real-time manipulation of latent variables, and the quality of the synthesised audio has indeed improved after experimenting with various latent dimensions. Sometimes, the improvements can emerge in an unexpected yet captivating way. For example, the piano model struggles to achieve the expected timbre transfer effect on percussion, but with manipulation of latent space, it was able to transfer some rock beats to sand-vibrating-like sounds, which can be used to create interesting auditory experiences.

In addition to performing timbre transfer tasks, integrating Max/MSP with multiple RAVE models can further facilitate a unique synthesis effect. This integration allows one to virtually assemble their own symphony orchestra alone by combining different models. An example of Max Patch that synthesises the two trained models is available on GitHub with recommended parameter settings included. Users are encouraged to explore and experiment with the model's latent spaces, or even with more trained models, to create innovative audio pieces.

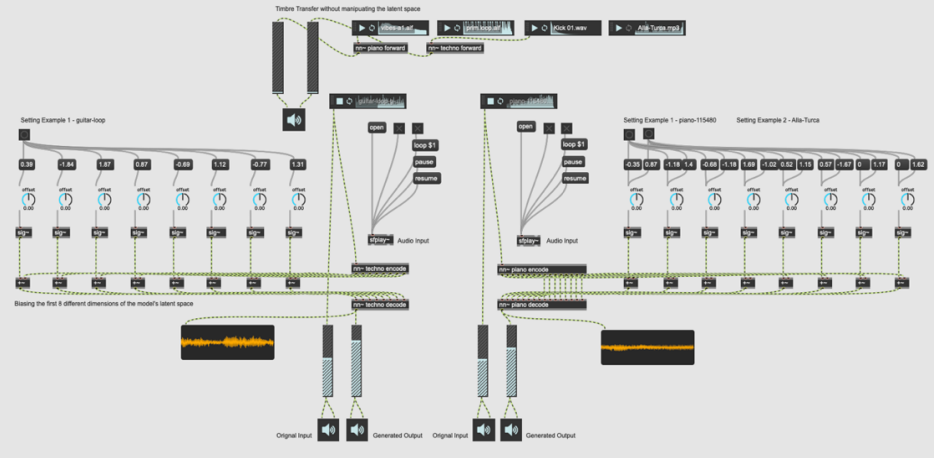


Figure 7 Max/MSP real-time latent space manipulation, initial template provided by Irini Kalaitzidi

4.4 Audio Visualisation

RAVE models can also be used to create AI audio-reactive drawings. The following graphs show a brief attempt at a Fast Fourier transform (FFT) based visualisation using Dorothy[6] Library alongside the code that I previously submitted for the STEM course.

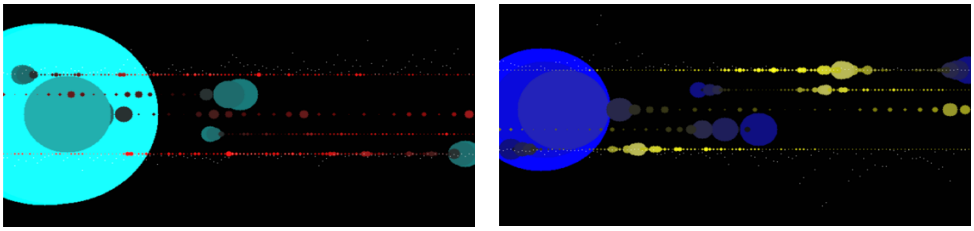


Figure 8 Audio Reactive Drawing with Dorothy - Example code and demo videos are available on GitHub.

5 Discussion

Overall, RAVE does demonstrate promising potential in creating realistic and novel audio pieces. Its ability to produce fast audio synthesis makes it well-suited for use in real-time interactive applications and live performances. However, both models in this project are still far

[6] Dorothy, a Creative Computing Python Library for Interactive Audio Generation and Audio Reactive Drawing, Copyright (c) 2023 Louismac, Available at: https://github.com/Louismac/dorothy/

from achieving high-quality results, and intensive manipulation of the latent space and post-processing adjustments, such as noise removal, are required to produce satisfactory audio pieces.

It may be beneficial to have a larger training dataset at the beginning, or with more extended training time, and to carefully experiment with the optimal batch size. Moreover, the training dataset could be further refined by removing silent parts and employing data augmentation techniques. For timbre transfer tasks, it may also be worth trying to preprocess the source audio, such as normalising its amplitude to match those of the training dataset and see how it affects the models' ability to reduce noise and produce higher-quality sound.

Regardless, the use of Max/MSP demonstrates that, even with an imperfect model, the flexibility and high control over the latent space can help produce interesting audio pieces, implying a highly promising potential in creating high-quality live audio synthesis with a better-trained model. The real-time audio processing could also be used in conjunction with the audio visualisation, possibly through the Open Sound Control (OSC) Protocol which is capable of streaming real-time audio control messages (Schmeder et al., 2010). This offers an interesting path to explore and produce highly interactive visual and sound applications in the future.

6 Conclusion

In conclusion, this project offers a valuable reference for training a RAVE model from scratch, with several suggestions provided for data processing and model training. Although the trained models are not yet fully refined to produce high-quality results, they successfully produced the expected sound effects and created some interesting audio and visuals. Training on a larger dataset as well as conducting real-time audio processing and visualisation using OSC between Max/MSP and other programs are proposed for further exploration.

LLM Disclaimer

ChatGPT has been used in this project for proofreading short passages of text, debugging code, and searching for codes and training issues not covered in class. Annotations have been included in the Jupyter notebooks where ChatGPT and external resources were used.

Bibliography

[1] Caillon, A. and Esling, P. (2021). RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. *arxiv.org.* [online] Available at: https://arxiv.org/abs/2111.05011 [Accessed 3 Mar. 2024].

[2] Engel, J., Agrawal, K.K., Chen, S., Gulrajani, I., Donahue, C. and Roberts, A. (2019). GANSynth: Adversarial Neural Audio Synthesis. *arXiv.org.* [online] Available at: https://arxiv.org/abs/1902.08710 [Accessed 10 Mar. 2024].

[3] Irhum Shafkat (2018). Intuitively Understanding Variational Autoencoders. *Medium.* [online] Available at: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf [Accessed 9 Mar. 2024].

[4] Lux, F. and Vu, N.T. (2022). Language-Agnostic Meta-Learning for Low-Resource Text-to-Speech with Articulatory Features. [online] *arXiv.org.* doi: https://doi.org/10.48550/arXiv.2203.03191 [Accessed 9 Mar. 2024].

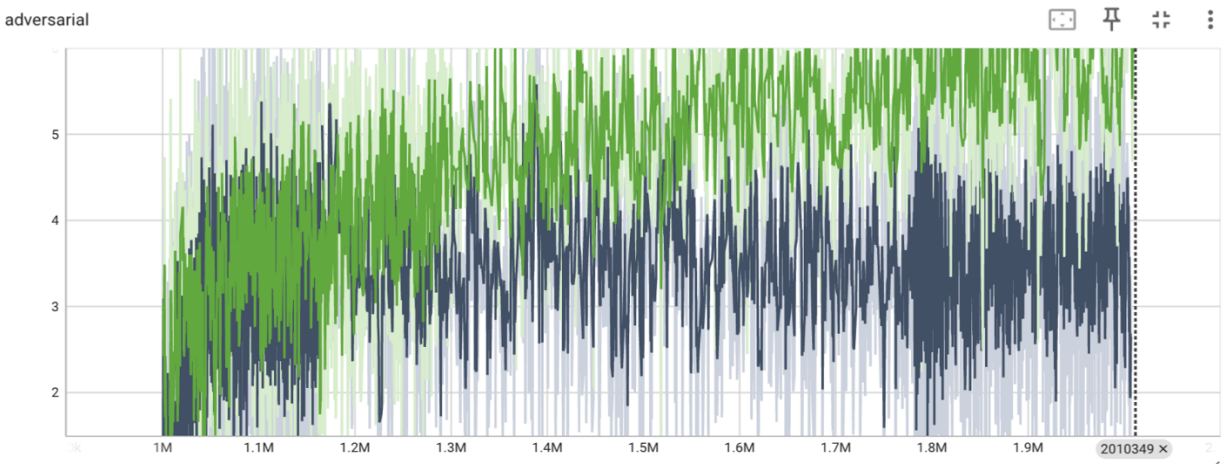
[5] Roads, C., 2015. Composing electronic music: a new aesthetic. *Oxford University Press*, p.8. Available at: https://books.google.co.uk/books/about/Composing_Electronic_Music.html?id=Ipw_CQAAQBAJ&redir_esc=y [Accessed 10 Mar. 2024].

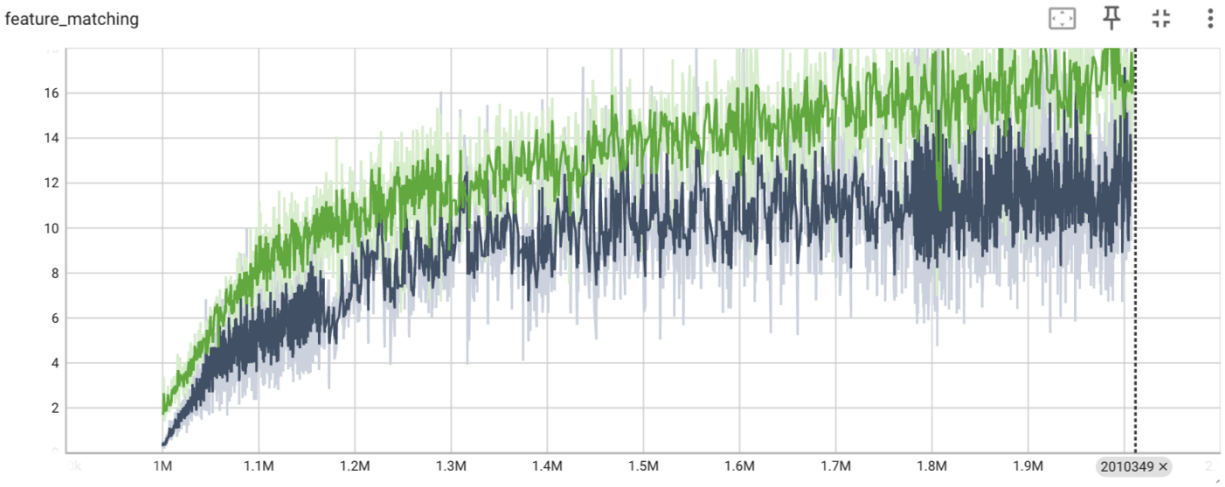
[6] Schmeder, A., Freed, A. and Wessel, D., 2010, May. Best practices for open sound control. *Linux Audio Conference (Vol. 10).* Available at: https://lac.linuxaudio.org/2010/papers/37.pdf [Accessed 12 Mar. 2024].

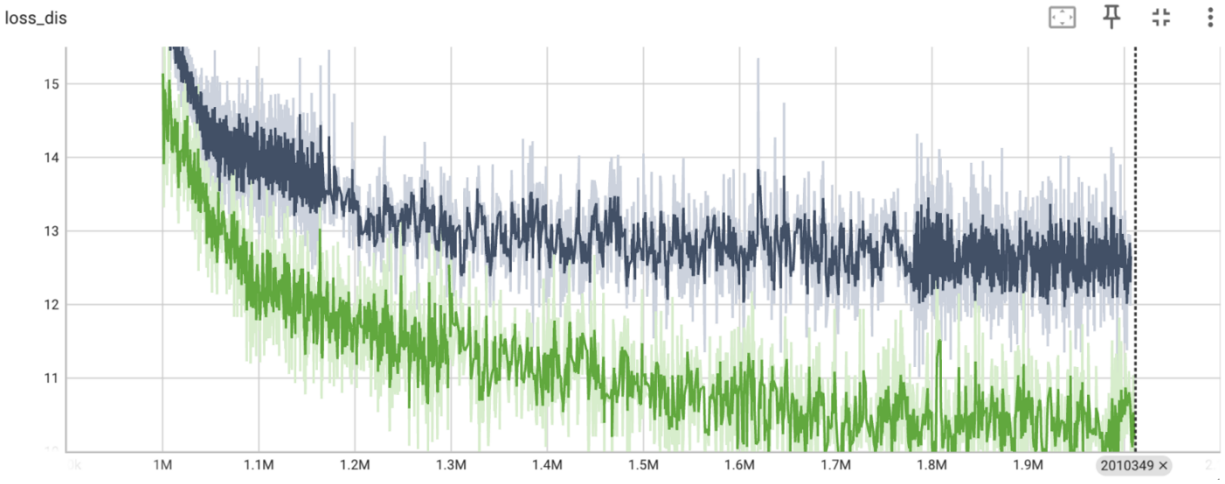
[7] White, M. (2023). A Brief History of Generative AI. *Medium.* [online] Available at: https://matthewdwhite.medium.com/a-brief-history-of-generative-ai-cb1837e67106 [Accessed 11 Mar. 2024].

Appendix

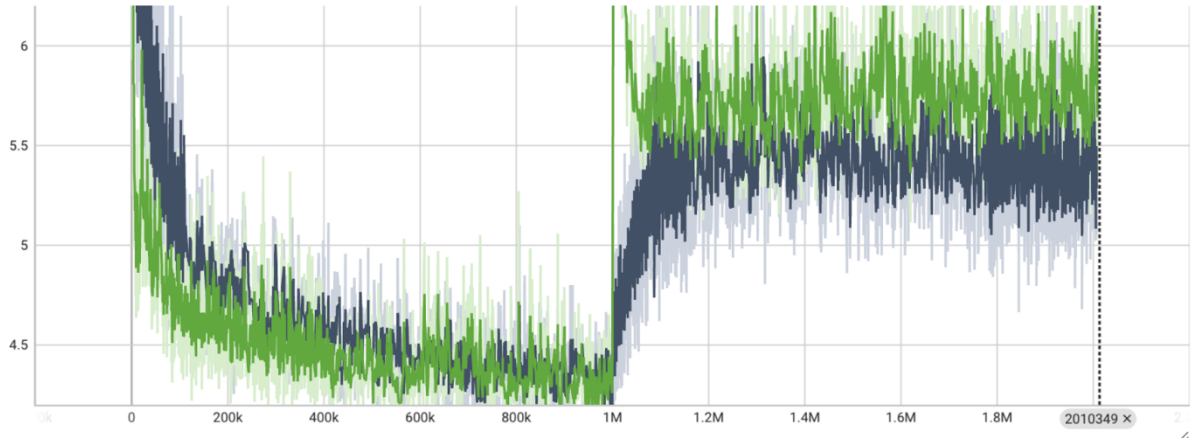
**TensorBoard Metrics (Scalar)* – rav_piano_model, rav_techno_model



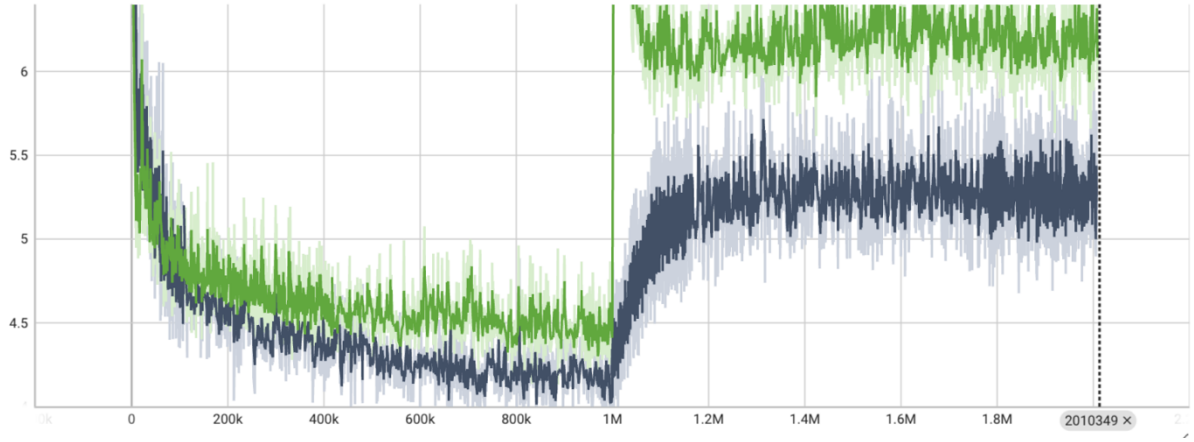




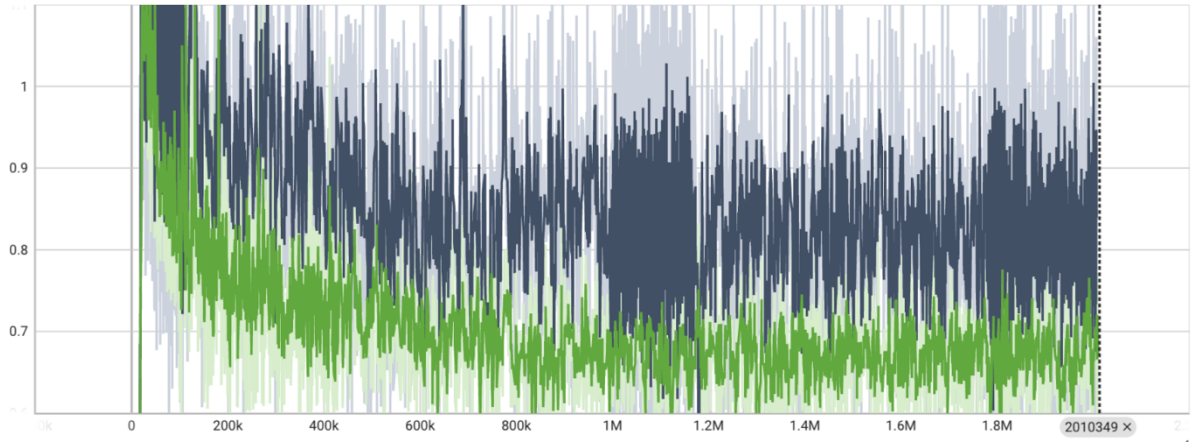
multiband_spectral_distance

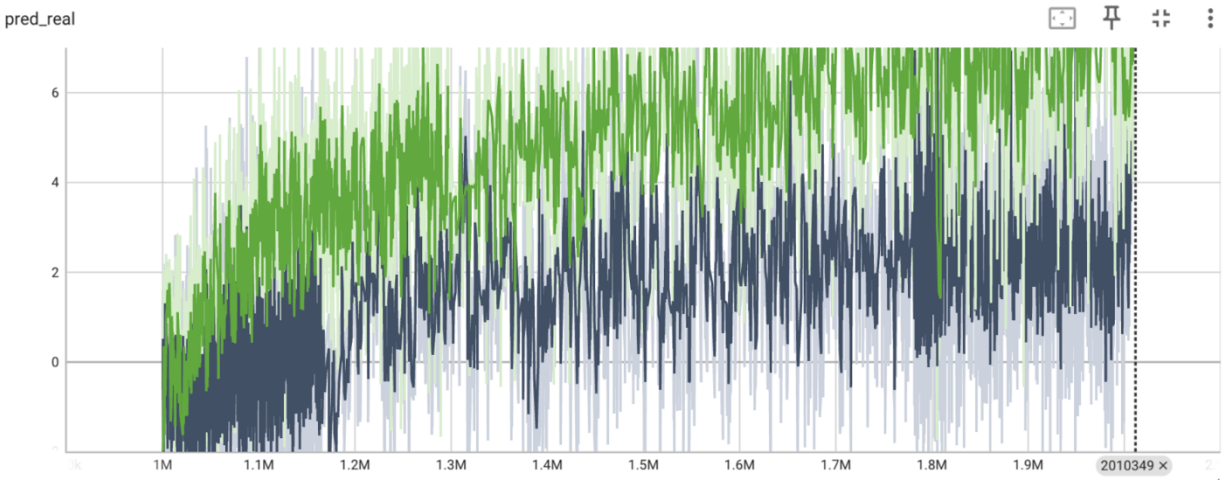


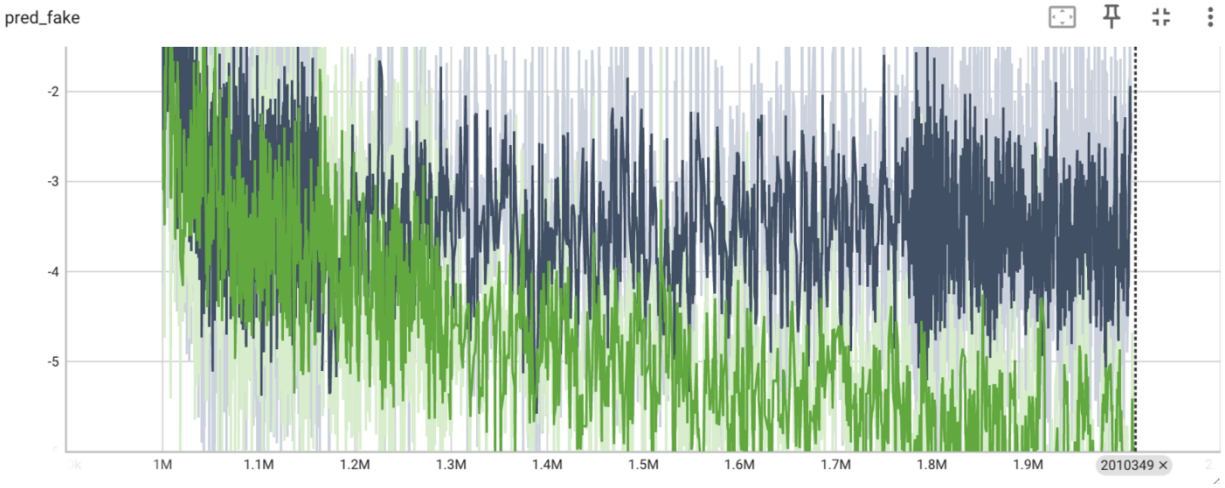
fullband_spectral_distance

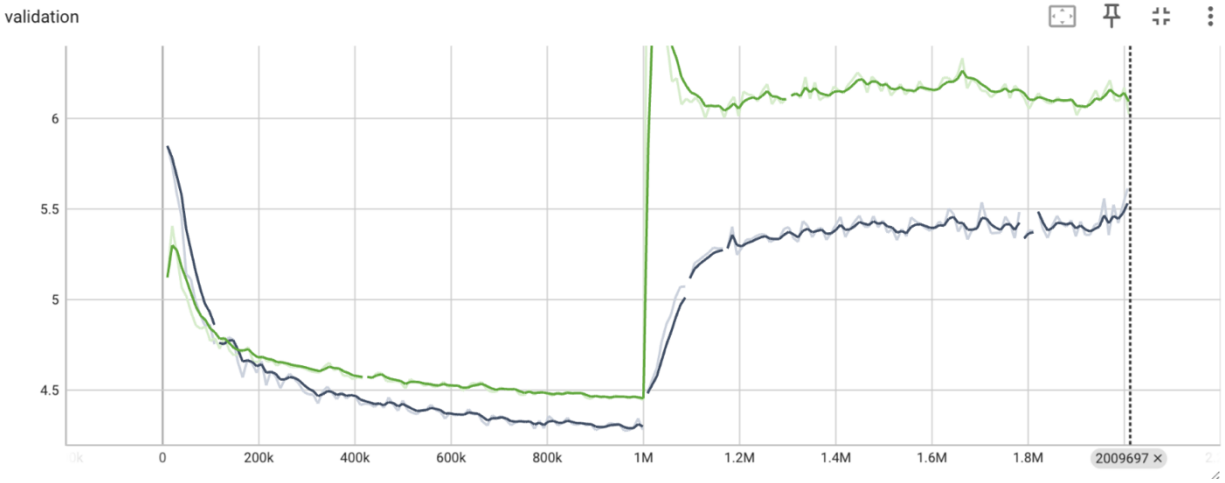


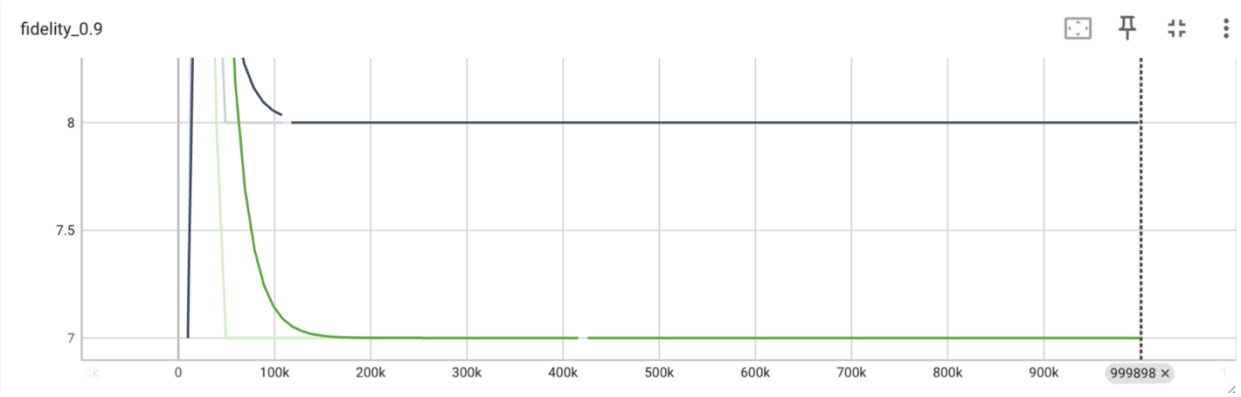
regularization

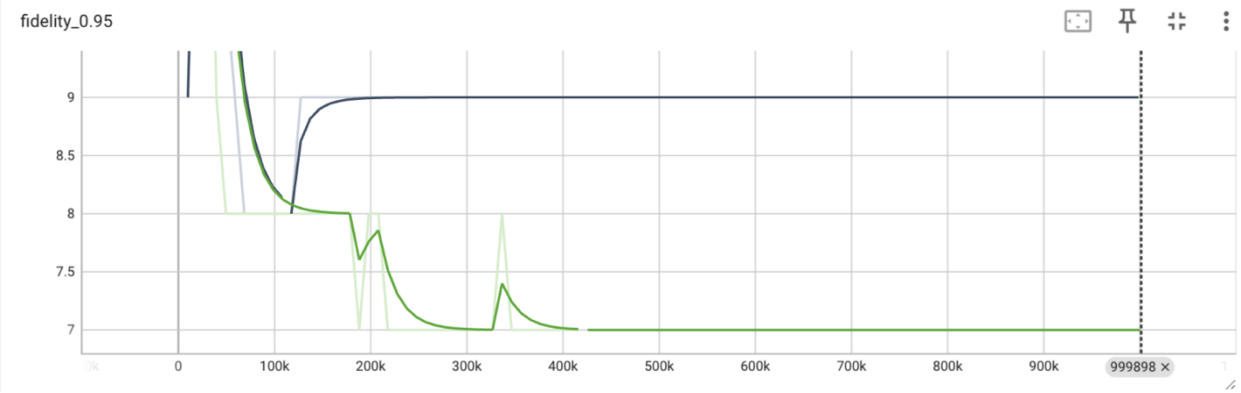


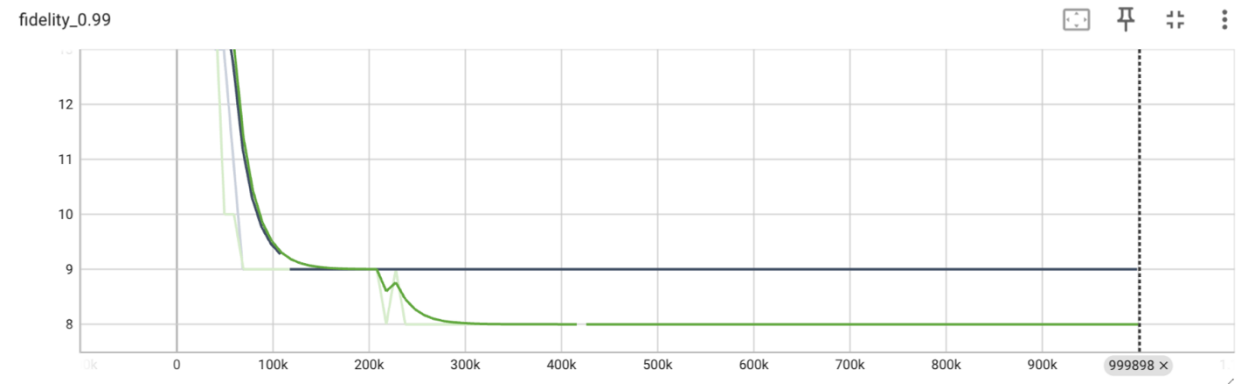

pred_real
6
4
2
0
1M
1.1M
1.2M
1.3M
1.4M
1.5M
1.6M
1.7M
1.8M
1.9M
2010349


pred_fake
-2
-3
-4
-5
1M
1.1M
1.2M
1.3M
1.4M
1.5M
1.6M
1.7M
1.8M
1.9M
2010349


validation
6
5.5
5
4.5
0
200k
400k
600k
800k
1M
1.2M
1.4M
1.6M
1.8M
2009697


fidelity_0.9
8
7.5
7
0
100k
200k
300k
400k
500k
600k
700k
800k
900k
999898


fidelity_0.95
9
8.5
8
7.5
7
0
100k
200k
300k
400k
500k
600k
700k
800k
900k
999898


fidelity_0.99
12
11
10
9
8
0
100k
200k
300k
400k
500k
600k
700k
800k
900k
999898