

**香港中文大學**

**計算機科學及工程學系**

**Department of Computer Science and Engineering,**

**The Chinese University of Hong Kong**

**Final Project Report**

Group 5: Audio Synthesis with Generative Adversarial Networks

**Name(s)**

**CHU Chun To 1155127149**

**CHOI Sen Ho 1155125303**

Supervised By

**Prof. LEUNG Kwong-Sak**

©2020 The Chinese University of Hong Kong

The Chinese University of Hong Kong holds the copyright of this proposal.

Any person(s) intending to use a part or whole of the materials in the thesis

in a proposed publication must seek copyright release from the University.

Contents

[1. Introduction 3](#_Toc57841947)

[1.1. Motivations 3](#_Toc57841948)

[1.2. Objectives 3](#_Toc57841949)

[2. Background 4](#_Toc57841950)

[2.1. Data representations 4](#_Toc57841951)

[2.2. GAN models 5](#_Toc57841952)

[2.3. Evaluation of models 5](#_Toc57841953)

[3. Methodology 6](#_Toc57841954)

[3.1. Conversion to Spectrogram 6](#_Toc57841955)

[3.2. GAN models 7](#_Toc57841956)

[3.3. Inception Score (IS) 10](#_Toc57841957)

[4. Implementation 10](#_Toc57841958)

[4.1. Input data 10](#_Toc57841959)

[4.2. GAN models 11](#_Toc57841960)

[4.3. Evaluation tools 12](#_Toc57841961)

[5. Experiment Details 13](#_Toc57841962)

[5.1. Experiment Procedure 13](#_Toc57841963)

[5.2. Experiment Results 13](#_Toc57841964)

[6. Results and Discussions 14](#_Toc57841965)

[7. Conclusion 15](#_Toc57841966)

[8. References 15](#_Toc57841967)

1. Introduction
   1. Motivations

Audio synthesizing is important to the development of computer music, electronic music and even AI music generation. Nowadays, synthesizing audio for specific domains has many practical applications in creative sound design for music and film. The demand for audio is high for different purposes, such as musicians finding sound effects for specific scenarios. [1]

One popular approach of audio synthesizing is using Generative adversarial networks (GANs). In current stage, GANs could enable rapid and straightforward sampling of large amounts of audio. Different models such as WaveGan, SpecGan, ParallenGan, GanSynth have their attempts on audio synthesis. They indeed produce some promising result and able to generate a lot of audio using small amount of time. However, the quality of result varies among different models and some of them has a lower matching human perception. Also, since audio is, unlike simple 2D images, time-domain data, the different representations of audio input will cause different restrictions during GANs training. Therefore, although GANs have seen wide success at generating images that are both locally and globally coherent, they have seen little application to audio generation. [1]

Even though today we have some powerful audio representations instead of raw audio waveform, such as Discrete Fourier Transform (DFT), Short-Time Fourier Transform (STFT), aligned linguistic features, to narrow the limitations, we still need to find the best one for different training models, especially the one suiting ours. This is important for us to have a good starting point in order to build the optimal model. Also, in order to generate audio with higher quality and quantity, we need to enhance our training model by using latest technology related to GAN combining with promising audio input pre-processing. Hopefully, we can have some new discovery in the audio synthesis technology, and this would help us chasing the current progress of digital music development.

* 1. Objectives

To achieve higher quality and quantity of generated audio, quantity of generated audio refers to a longer length of audio, while quality refers to a better hearing effect from the human perception.

To measure how good our model when compared to other models can do, several measurements will used in the evaluation. For ease of comparisons, we will focus on the quantitative evaluation by using Inception Score, an objective metric for evaluating the quality of GAN generated results. The details of implementation will be shown in the later parts.

1. Background
   1. Data representations

GANs have seen wide success at generating images that are both locally and globally coherent, but they have seen little application to audio generation due to the time-domain features limitations of audio data representations. [1] Therefore, in our project, we apply two kinds of audio data representations to train our models in order to limit the restrictions and aiming to find the better one.

**2.1.1 Raw Audio Waveform**

The first data representation form of audio data we used is raw audio waveform. Waveform is the most common and easiest representation form of audio data. It is merely a graph that displays amplitude or level changes over time. [2] Amplitude is measured in a bipolar manner, with positive and negative values. It has been widely applied in different kinds of GANs of image-audio synthesis due to the ease of extraction and conversion from input to output. Therefore, we used waveform as the early attempts to our project.

There are some intrinsic differences between audio and images. One way to illustrate the differences between audio and images is by examining the axes along which these types of data vary most substantially, i.e. by principal component analysis. [1] In general, natural audio signals are more likely to exhibit periodicity than natural images. This provides both ease in data extraction (input) and limitation in data generation (output). However, the limitations are mostly can be ignored since the input data are quite short and application of convolutional layers in the training models.

One particular challenging problem of modelling raw audio is the high temporal resolution of the audio data. They are usually at least 16,000 samples per second and the presence of time-domain structure at different timescales with short and long-term dependencies. [3] The features might be difficult to figure out when we use one-dimensional transpose convolution layers to low-dimension feature vector maps into a high dimension of audio output.

**2.1.2 Spectrogram**

Instead of modelling the raw temporal audio (waveform) directly, most approaches simplify the problem by modelling a lower-resolution representation that can be efficiently computed from the raw temporal signal [3]. Also, since deep learning for audio has shifted from using hand-crafted features requiring prior knowledge, to features learned from raw audio data or mid-level representations, this has allowed us to build models requiring less prior knowledge, yet at the expense of data, computational power, and training time [4].

There are several representations of audial data in a deep-learning approach, include aligned linguistic features and spectrograms. The former is usually the intermediate representation given text as input, which is more likely processing speeches. [1] The latter transforms the intermediate representation back to audio, which is more applicable in general cases of audio synthesis. Therefore, in our second attempt, we focus on spectrograms as the intermediate representation. Spectrogram is a frequency domain (magnitude vs frequency) representation of the original time domain (amplitude vs time) audio data. It can be obtained by Fourier analysis such as Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Short-Time Fourier Transform (STFT), etc.. They are commonly used intermediate representations suiting different audio data.

In our project, we mainly use STFT, to transform our audio data from 1D samples into 2D spectrogram representation. For example, for Nsynth dataset, we will transform the one second audio with 16384 samples into 128\*128 spectrogram. Also, since the entries of resulting transformation are with complex number, we decompose into two layers, one with real part and one with imaginary part for training.

The output result of GANs with spectrograms are also spectrograms. In order to generate the audio data output, we need to convert the output spectrograms back to raw audio waveform. It is a relatively simple process by existing libraries, so the overall implementation difficulties are same of waveform and spectrogram approaches.

* 1. GAN models

In order to find out a better GAN model for audio synthesis, we involved three different kinds of GAN models to train. They are DCGAN, WGAN and WGAN-GP. We first test for the waveform approach to these three models, and then test for the spectrogram approach on the best model among these three GANs. The details of these models will be discussed in the later parts.

* 1. Evaluation of models

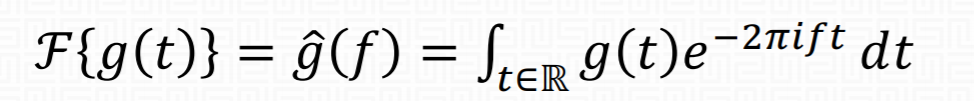
In our project, we are using two evaluation tools to test our models, which are human evaluation and Inception Score (IS). Inception Score is the main focus since it provides more quantitative results to compare with the past works. The details will also be discussed in the later parts.

1. Methodology
   1. Conversion to Spectrogram

Recall that spectrogram is a frequency domain (magnitude vs frequency) representation of the original time domain (amplitude vs time) audio data. It can be obtained by Fourier analysis such as Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Short-Time Fourier Transform (STFT), etc.. Among them, STFT is the most common in use to extract spectrograms so we are also using it. But before applying STFT directly, some techniques of DFT are required in practice.

**3.1.1 Discrete Fourier Transform**

The basic idea of Fourier Analysis is to extract waves of different frequencies and magnitudes consisted in an audio soundwave. Here is a mathematical equation of how it works:

[5]

Notice that , so the equation can be viewed as counting the occurrence of frequencies in the waveform by spinning the signal around a circle at the particular frequency. However, in a digital system, the input values (audio samples) are equally spaced. This function is continuous for f and t, which cannot be applied to digital signals. Therefore, we have to deal with the sample in a discrete approach, to find the sum of finite series of sinusoidal waves. For a sequence of complex samples , we can turn it into a sequence of complex numbers:

Here, the series are called the DFT coefficients of frequency bins. We can easily find the magnitudes (), phase () and bin frequency with some easy mathematical approaches.

DFT is popular in digital signal processing because of its simplicity. DFT is usually applied simply as a black box in different programming libraries, without understanding the math behind. Also, it is usually implemented as Fast Fourier Transform (FFT) with complexity , where traditional DFT is . However, performing the DFT on a time series will give us the overall frequency components for the entire time series. It can only show the general “histogram” of frequencies, which cannot tell us how the frequency appears to be changing over time.

**3.1.2 STFT**

By Short-Time Fourier Transform (STFT), we can break the process into multiple DFT (as well as FFT) in time segments to analyze the frames. STFT decomposes a signal as a weighted sum of complex sinusoidal basis vectors with linearly spaced center frequencies, unveiling the time-frequency structure of an audio signal.

It is commonly decomposed into magnitude and phase components. The latter is typically noisy, which makes it difficult for neural networks to model. This problem is mitigated by using the Instantaneous Frequency (IF), providing a measure of the rate of change of the phase information over time. The STFT of a finite, real signal , with the analysis window , time step and frequency channels is given by:

一張含有 文字 的圖片

自動產生的描述[5]

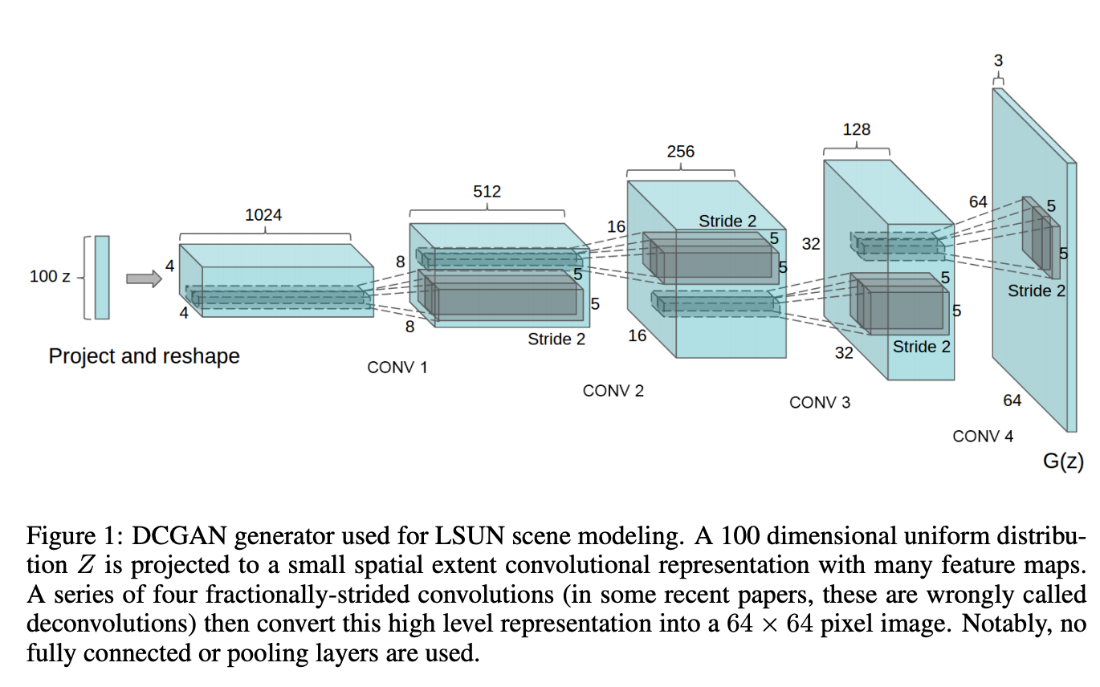
We can get a spectrogram by using STFT, which shows the magnitude and frequency changes within a time period (magnitude vs frequency vs time). Sometimes, spectrogram with Mel scale (log scale) is applied due to the features of changing in sound frequencies. STFT is cheap to compute and perfectly invertible, which makes it popular for audio synthesis.

* 1. GAN models

It is hard to look for the best representation in general cases, but the options can be narrowed down after discovering the GANs training modal. In our project, we involved three kinds of GAN models.

**3.2.1 DCGAN (Radford, 2015)**

The audio is constructed with multiple periodic sine wave. To extract the periodic bias in audio, convolution layers is used to extract those features. Deep Convolutional Generative Adversarial Network (DCGAN) will include inside the model, it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator. [6]



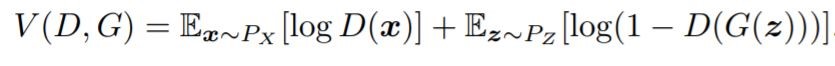
(Radford, 2015)

As shown in the paper, the convolution layers will allow both the generator and discriminator to become strong feature extractor. The convolution layer in the generator can achieve up-sampling from the low dimension input into the high dimension of desired output, while the convolution layer in the discriminator can achieve down-sampling to determine whether a desired output is true or fake. The discriminator can learn features for determining the output, just as supervised CNN on object classification, while the generator learns specific representations for major components. For example, the unsupervised models can learn to convincingly model object attributes like scale, rotation, and position. [6]

Therefore, deep convolution layers will be included in the model for better training result. The detailed implementation will be shown in later part.

**3.2.2 WGAN** **(Arjovsky, 2017) and WGAN-GP (Gulrajani, 2017)**

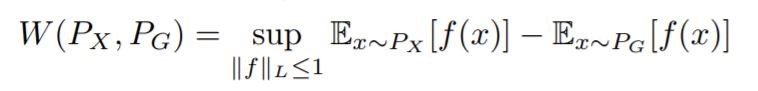
Generative adversarial network is an unsupervised learning that given a certain data set, they can produce new data that resemble the given data. Those generated data can reach a high level of realism. The network consists of two parts: a generator G and a discriminator D. The generator learns to generate plausible data to deceive the discriminator, while the discriminator learns to distinguish the generator's fake data from real data. Goodfellow et al. [7] uses the following objective function where G is trained to minimize it and D is trained to maximize it.



Px represents the real data distribution and Pz represents the random data with Gaussian distribution. The generator will be trained to map those random data to data like real data.

However, this training model is difficult to train, the unstableness brought by the serious vanishing gradient due to Jensen-Shannon divergence [7].

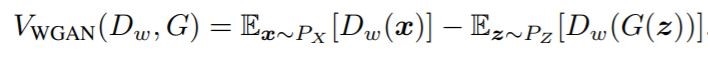
Therefore, a recent improvement on GAN is using Wasserstein-1 distance between generated and data distributions. The Wasserstein distance is the minimum cost of transporting mass in converting the data distribution Px to the data distribution Pg (Arjovsky, 2017).



where ||f||L ≤ 1 is the family of functions that are 1-Lipschitz where it follows Lipschitz constraint.

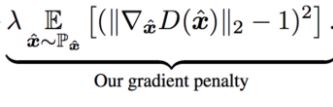


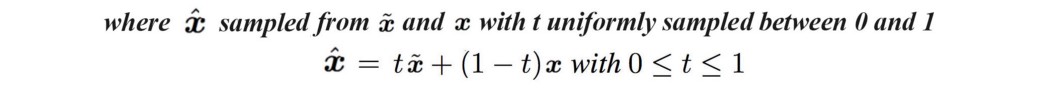
To minimize Wasserstein distance, the objective function is changed as below (Arjovsky, 2017).



This represents Wasserstein Generative adversarial network. To maintain Lipschitz continuity, weight clipping is used as enforcing that Dw is 1-Lipschitz. The incorrect tuned hyperparameter in weight clipping will produce poor result.

Proven in paper of Improved Training of Wasserstein GANs (Gulrajani, 2017), a differentiable function f is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. Gradient penalty make use of this feature and it replaces weight clipping. It will allow the model to penalize the model if the gradient norm moves away from its target norm value 1. The additional term of the objective function will be the following (Gulrajani, 2017).



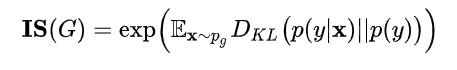


This will be the whole picture of WGAN-GP. The training will be based on WGAN-GP and the detailed implementation will be shown in later part.

* 1. Inception Score (IS)

**3.3.1 Inception Score (IS) (Barratt,2018)**

Inception Score reflects both distinction and variety of the group of generated audios. It provides more quantitative results which to compare with the past works. It is defined as the mean KL divergence between the label distribution p(y|x), and the marginal distribution p(y) using the predictions of an Inception classifier (Barratt, 2018).



The Inception classifier return the probability distribution of the input data. The Inception Nets will be trained on the tasks of instrument and pitch classification. The label distribution of an output represents the probability of output belonging to a certain category, which indicates the distinction of generated outputs, while the marginal distribution is given by summing the label distribution of many generated samples, which indicates the variety of generated outputs. KL divergence can measure how different two probability distributions are. High KL divergence indicates the ideal situation.

**3.3.2 Nearest Neighbor (Borji,2019)**

The nearest neighbor method can be used to qualitatively summarize generated audio. Using Nearest Neighbors model, we can fit with a certain distribution of input data. Then, we can calculate the average distance between the input data and other data. It is common to detect overfitting in GAN model, whether the generated output is close to the training set.

1. Implementation
   1. Input data

We use audio sample from NSynth dataset for our GAN models. The dataset contains 300,000 musical notes from 1,000 different instruments aligned and recorded in isolation (Engel, 2019). NSynth is a difficult dataset composed of highly diverse timbres and pitches, but it is also highly structured with labels for pitch, velocity, instrument, and acoustic qualities. Each sample is four seconds long, and sampled at 16kHz, giving 64,000 dimensions.

Under STFT transformation, audio data will transfer into the format like Spectrogram and further processing using the magnitude of frequency with time will become the sample data for training.

* 1. GAN models

For audio synthesis, we have our four different GAN models in attempting, and compare their results in order to find the best promising model in generating a high quality with great variety of audio.

**4.2.1 DCGAN**

In general, natural audio signals have a stronger periodicity. Correlations across large windows are common in audio. To extract the periodicity of the audio, the audio is processed with filters with larger receptive fields. For our DCGAN for audio, our architecture is based on DCGAN which is for image synthesis [6]. Our DCGAN generator uses the one-dimensional transpose convolution layers to iteratively upsample low-dimension feature vector maps into a high dimension of audio output.

To widen its receptive fields, the longer one-dimensional filters of length 25 is used, and it upsample by a factor of 4 at each layer. The discriminator is modified in a similar way, using length 25 filters in one-dimensional and increasing stride to 4. For the output of generator model, it is with dimension of 16384, which is slightly more than one second of real audio in 16kHz, where audio is in waveform representation. The length of generated audio is enough for us to verify the efficiency our GAN model on synthesis of audio. Future work will attempt to expand the audio output into a longer period. This architecture will be adopted for the remaining models which can be seen in Appendix.

For training of this model, we adopt the same approach in the original paper of GAN (Goodfellow, 2014), the mentioned objective function of Goodfellow is used. The generator and discriminator are updated in equal amount.

**4.2.2 WGAN**

For our second model, the architecture is same as the previous DCGAN model. The major difference is that we adopt the modification of WGAN (Arjovsky, 2017), which is an improvement of GAN. The output layer of the discriminator uses a linear activation function.

In the training process, we update the discriminator model five more times than the generator each iteration. After each mini batch update in discriminator, the weights of discriminator model are constrained within [-0.01,0.01] in order to fulfill the Lipschitz constraint, which is weight clipping approach. Wasserstein loss function replaces the objective function in previous DCGAN to train the critic and generator models that promote larger difference between scores for real and generated images, where in implementation, the class label for real audio is 1 while for fake audio is –1.

**4.2.3 WGAN-GP**

For our third model, the main difference from previous WGAN model is the replacement of gradient penalty (Gulrajani, 2017) instead of weight clipping. The additional term from above is added into the error function, to penalize the model if the gradient norm moves away from its target norm value 1, where the gradient is taken from the random sample point between the real audio and fake audio, which can ensure the Lipschitz constraint. We set the gradient penalty coefficient λ as 10 in our model.

**4.2.4 WGAN-GP-SPEC**

In recent research in discriminative audio classification tasks, most of the approaches are using spectrogram representations of audio. Therefore, for our last model, we take advantage on the spectrogram representations. By Short-Time Fourier Transform, we transform the one second audio with 16384 samples into 128\*128 spectrogram. Since the entries of resulting transformation are with complex number, we decompose into two layers, one with real part and one with imaginary part. This model follows similar architecture as the previous WGANGP and use the training method as WGANGP. The generated output from the generator will be transformed back into audio by the inverse of Short-Time Fourier Transform.

* 1. Evaluation tools

**4.3.1 Inception score (Barratt, 2018)**

To measure the inception score, we train an audio classifier on Nsynth. Our classifier takes input directly with audio in waveform. We process this waveform audio input with five layers of convolution with batch normalization and max pooling, projecting the result to softmax layer with 11 classes, which is same as the number of instrument family in the dataset. We train the model to the convergence (around 100000 epochs) and the model achieve 70% accuracy on the test sets. The architecture can be seen in Appendix. The score is calculated based on a large number of generated audio (e.g. 5k). [8]

**4.3.2 Nearest neighbor comparison**

Inception score has two trivial failure cases that a poor generative model might result in high score. [8] First one is that the model that outputs a single example of each class with uniform probability, where it reflects the problem of model collapse. Second one is that model that overfits the training data by outputting inputs on which the classifier was trained will get a high score.

We introduce two metrics to determine whether those two failure cases happen in our model. Our first indicator, measures the average Euclidean distance

of a set of 1k examples to their nearest neighbor within the set (Donahue,2018). The higher the value, the higher the diversity among the samples. Our second indicator, measures the average Euclidean distance of a set of 1k examples in the real training data. [8] If the model simply produces examples from the training set, this measure will be 0.

1. Experiment Details
   1. Experiment Procedure

For audio synthesis, our experimentation focuses on the Nsynth dataset. This audio dataset contains 305,979 musical notes, each with a unique pitch, timbre, and envelope. The audio comes from 1006 different instruments belong to 11 instrument family. Each audio contains labels of other characteristics, such as pitch and velocity. The length of each audio is 4s. The dimension of audio will be 64000 by taking 16000 samples per second. We only extract the first second of audio for our data.

Our baseline configuration is our first DCGAN model. We will compare it with the remaining three models (WGAN, WGANGP, WGANGP-SPEC) to find out which one will be best promising model in generating a high quality with great variety of audio.

Due to large training set of audio data, we train our networks using batches of size 64 on Google Colab with GPU so that we can take advantage on loading dataset from public cloud storage in TensorFlow. All models are trained to convergence. Our DCGAN converges within 21000 epochs. Our WGAN converges within 54000 epochs. Our WGANGP converges within 21000 epochs and produces instrument-like audio within 10000 epochs. Our WGANGP-SPEC converges within 10000 epochs. The training result can be seen in Appendix.

* 1. Experiment Results

The following table (Table 1) shows the quantitative results for our different GAN models experiments with Nsynth dataset comparing real and generated data. A higher inception score suggests that semantic modes of the real data distribution have been captured. indicates the intra-dataset diversity relative to that of the real test data. indicates the distance between the dataset and the training set relative to that of the test data; a low value indicates a generative model that is overfit to the training data.

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment | Inception score |  |  |
| Real (train) |  |  | -- |
| Real (test) |  |  |  |
| DCGAN |  |  |  |
| WGAN |  |  |  |
| WGAN-GP |  |  |  |
| WGAN-GP-SPEC |  |  |  |

(Table 1)

1. Results and Discussions

Results for our evaluation appear in Table 1. We also evaluate our metrics on the real training data, the real test data. The generated sound examples from all models can be found.

While the maximum inception score for Nsynth is 10, any score higher than test score of 7.5 should be seen as evidence that a generative model has overfit [1]. Our best GAN model is the WGAN-GP with waveform as data representation, which achieves an inception score of 7.12. The score is much higher than that of the DCGAN and WGAN. We can see that the improvements of from DCGAN, WGAN to WGAN-GP are remarkable.

All experiments produced (diversity) and (distance from training data) values are lower than that of the test data, and for our best model (WGAN-GP with waveform), the scores are very close to the test data. While these measures indicate that our generative models produce examples with statistics that close to those of the real data, which is a good sign.

For , DCGAN, WGAN and WGAN-GP-SPEC all have a relatively low score when compared to that of train data. These three models indicate that the models are with low variety of generated samples brought by the model collapse. The model collapse in DCGAN is believed to be because of Kullback–Leibler divergence. Weight clipping in WGAN will be a reason for model collapse. WGAN-GP-SPEC is due to in sufficient training time.

For , DCGAN, WGAN and WGAN-GP all have a large value, indicates that the generated output of these models is not similar to that of train set. No overfitting occurs in these models.

Finally, since we built the WGAN-GP-SPEC at the last stage, we do not have sufficient time to train it and the results are relatively low comparing with WGAN-GP (raw audio waveform). However, the Inception Score of WGAN-GP-SPEC is still higher than that of DCGAN and WGAN even with much shorter training time. This again shows the power of WGAN-GP model.

In many relative research papers, the results using spectrogram as input data representation produce better results than that using waveform. For example, for the WaveGAN (GAN with attempt of waveform) and SpecGAN (GAN with attempt of spectrogram) research paper in 2019, SpecGAN produced a better quantitative result [1]. We believe that WGAN-GP-SPEC would give us more surprise results proven by time.

1. Conclusion

We present DCGAN, WGAN and WGAN-GP as the three types of GANs to unsupervised audio generation. Also, we tried to give a comparison between waveform and spectrogram as input data representation, although the results are not so complete due to time limit. What can be confirmed is, WGAN-GP give us an amazing result even it is just modified a little bit from WGAN. It shows the possibilities of optimization of existing GAN models and provides more directions for us to discover. In our future work, we plan to verify the results comparing waveform and spectrogram as input data representation, by training more and testing more different datasets. In conclusion, we hope that this work catalyzes future investigation of GANs for audio synthesis.

1. References

[1] Chris Donahue, Julian McAuley, Miller Puckette. 2019. “ADVERSARIAL AUDIO SYNTHESIS”. ICLR 2019. [online] Available at: <https://arxiv.org/pdf/1802.04208.pdf> [Accessed 2 December 2020].

[2] SoundBridge. 2020. What Are Waveforms And How Do They Work? - Soundbridge. [online] Available at: <https://soundbridge.io/what-are-waveforms-how-they-work/> [Accessed 2 December 2020].

[3] Kundan Kumar, Rithesh Kumar. 2019. “MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis”. ArXiv 2019. [online] Available at: <https://arxiv.org/pdf/1910.06711.pdf> [Accessed 2 December 2020].

[4] Javier Nistal, Stefan Lattner, Gael Richard. 2020. “Comparing Representations for Audio Synthesis Using Generative Adversarial Networks” ArXiv 2020. [online] Available at: <https://arxiv.org/pdf/2006.09266.pdf> [Accessed 2 December 2020].

[5] Müller, Meinard. 2015. “Chapter 2: Fourier Analysis of Signals”, “Fundamentals of Music Processing”. Publisher: Springer.

[6] Pytorch.org. 2020. DCGAN Tutorial — Pytorch Tutorials 1.7.0 Documentation. [online] Available at: <https://pytorch.org/tutorials/beginner/dcgan\_faces\_tutorial.html> [Accessed 2 December 2020].

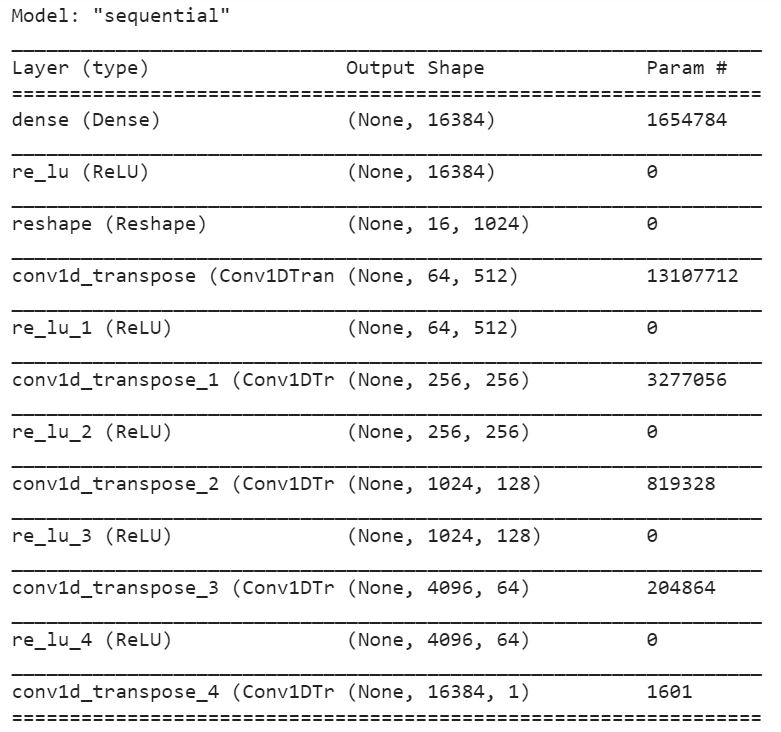
[7] Martin Arjovsky, Soumith Chintala, L´eon Bottou. 2017. “Wasserstein GAN”. ArXiv 2017. [online] Available at: <https://arxiv.org/pdf/1701.07875.pdf> [Accessed 2 December 2020].

[8] Shane Barratt, Rishi Sharma. 2018. “A Note on the Inception Score” ArXiv 2018. [online] Available at: < https://arxiv.org/pdf/1801.01973.pdf> [Accessed 2 December 2020].

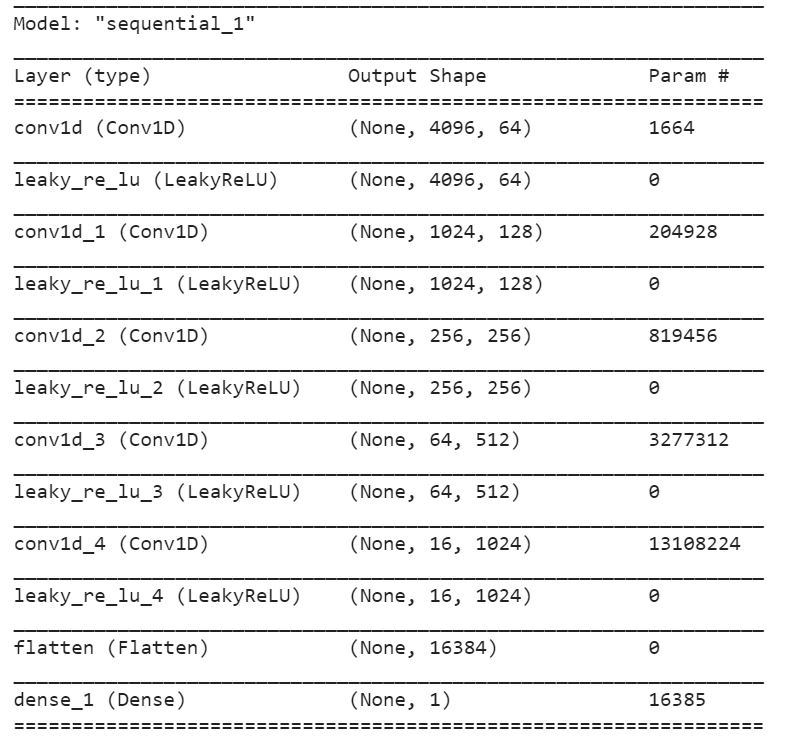
1. Appendix

Architecture of DCGAN, WGAN, WGAN-GP:

Generator:

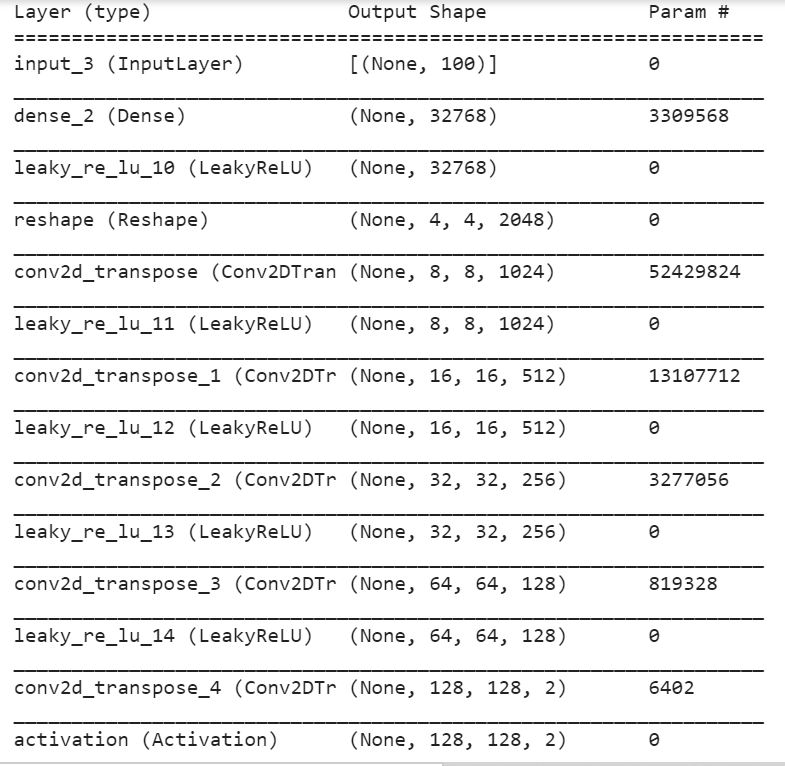


Discriminator:

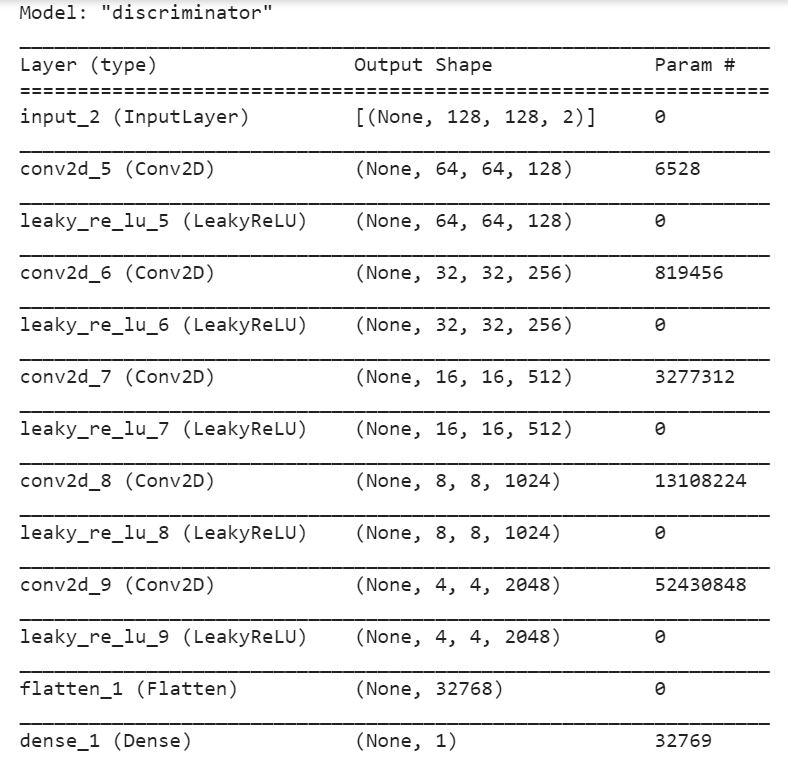


Architecture of WGAN-GP-SPEC:

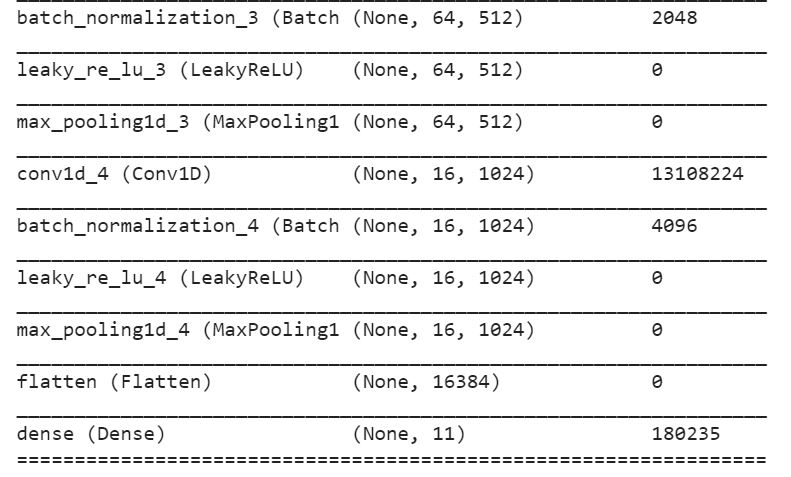
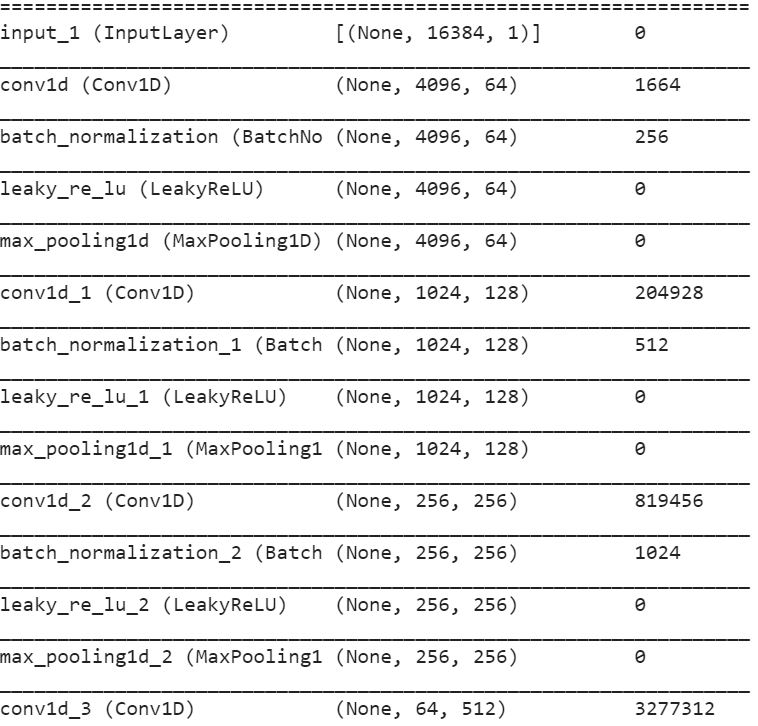
Generator:



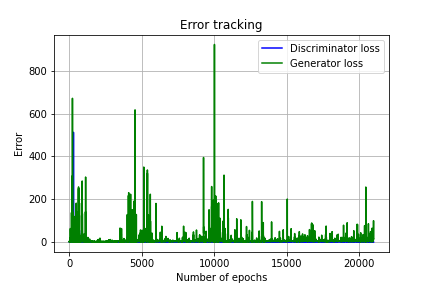
Discriminator:



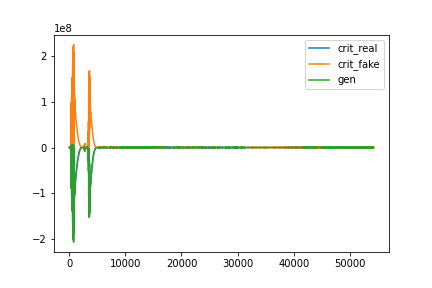
Architecture of audio classifier for inception score:



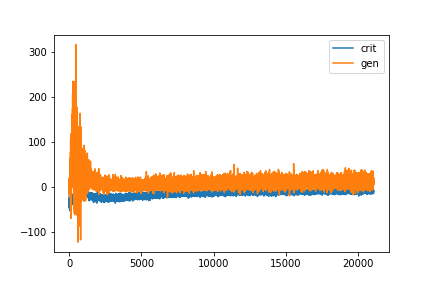
Training Error of DCGAN:



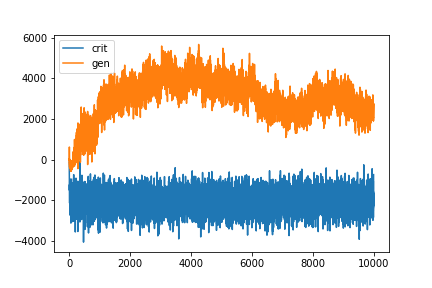
Training Error of WGAN:



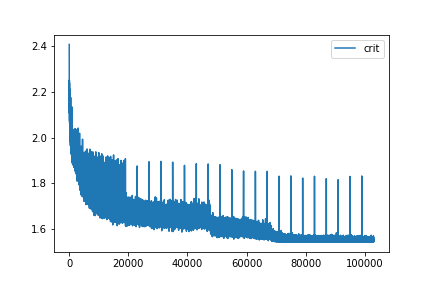
Training Error of WGAN-GP:



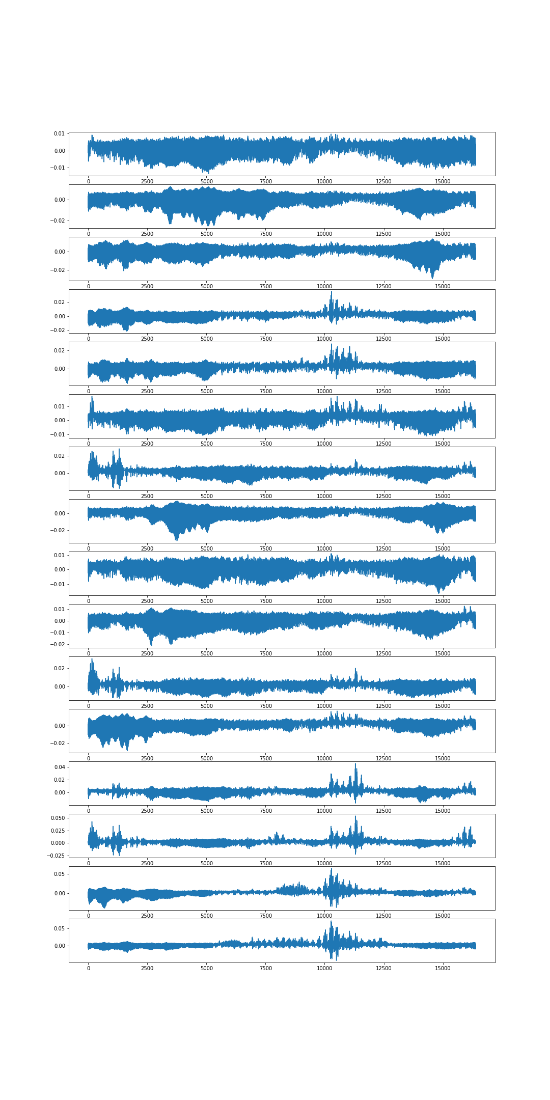
Training Error of WGAN-GP-SPEC:



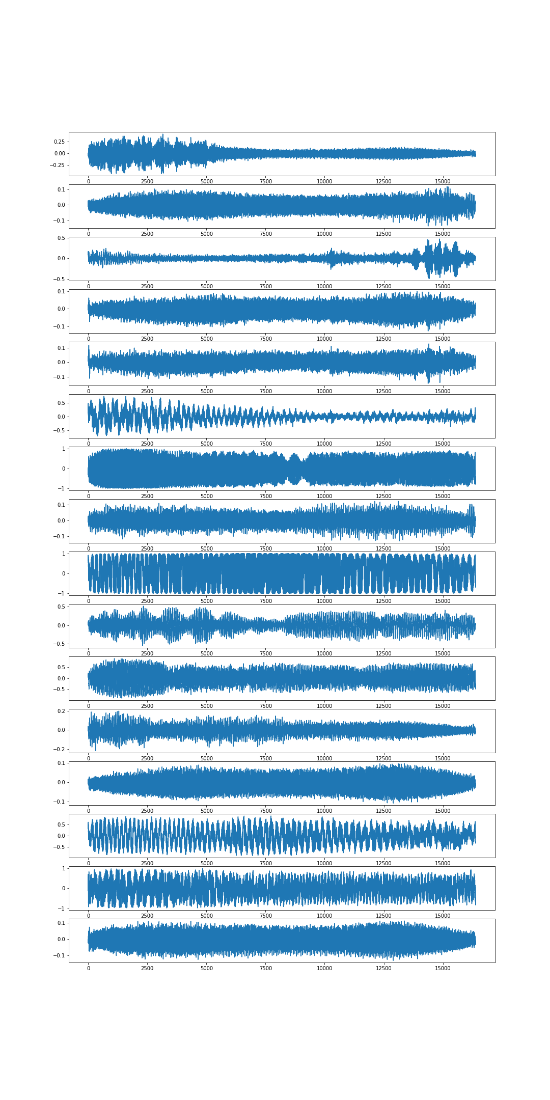
Training Error of audio classifier:



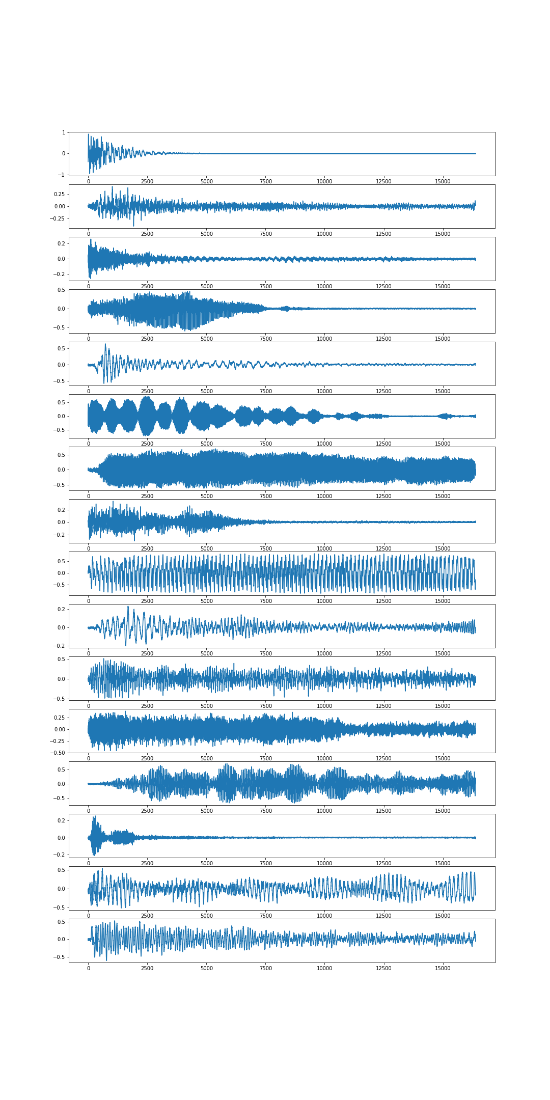
Some generated audio of DCGAN (in waveform image):



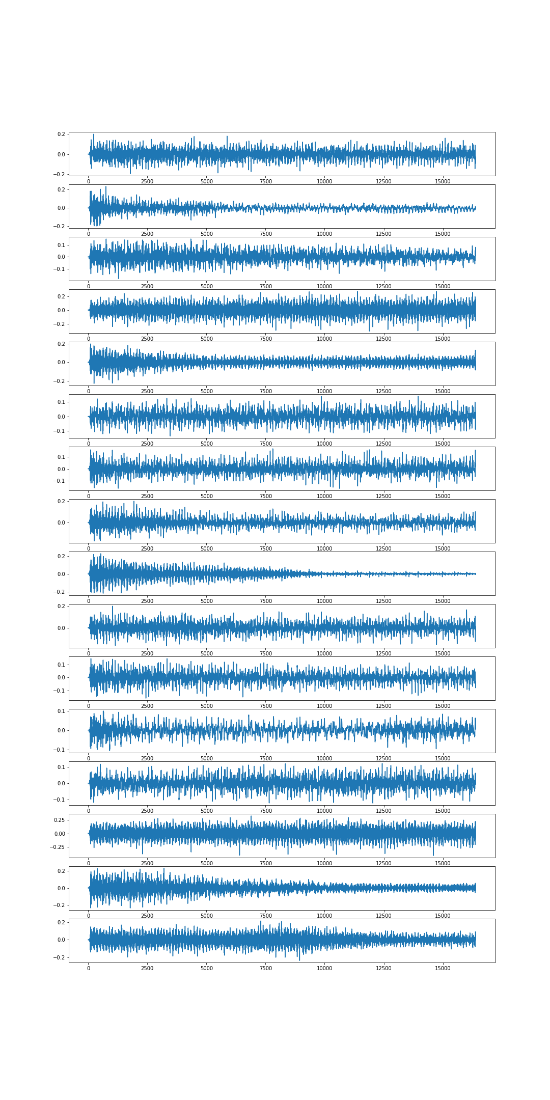
Some generated audio of WGAN (in waveform image):



Some generated audio of WGAN-GP (in waveform image):



Some generated audio of WGAN-GP-SPEC (in waveform image):



DCGAN:

<https://colab.research.google.com/drive/1lzDVLUtxt2MaxdSTAHCLPH47nWbDRxvE?usp=sharing>

WGAN:

<https://colab.research.google.com/drive/15EwB1_Duy-qAifO_dEv9mIThkdVn99eI?usp=sharing>

WGAN-GP:

<https://colab.research.google.com/drive/1HvktFZTAgx39BYReC0xV5TVQRE7ksiGq?usp=sharing>

WGAN-GP-SPEC:

<https://colab.research.google.com/drive/19xwhAMNVgWt78HmQklimrgyuW1DHKDmD?usp=sharing>

Audio classifier:

<https://colab.research.google.com/drive/1XUNfg65bLViGTZjUK7YN_Go47NOCY5z7?usp=sharing>

Evaluation code:

<https://colab.research.google.com/drive/1iiyFk_VQgG9CwjqkhPBdjC0lGkK48R1x?usp=sharing>

Generator in all models (can directly load to generate sound in keras model):

<https://drive.google.com/drive/folders/1YwblobEyXn_OG3Lz1kmbck5MljkPxrcC?usp=sharing>