Audio Generation: WaveGAN with LSTM D-net

Weishang Quan
Team 10
NUID 001855399
quan.w@husky.neu.edu

1 Introduction

1.1 Introduction

The problem I am trying to solve is the automatic audio generation problem with raw audio input data using GANs. Generation problems are very traditional ones in the field of machine learning. The generation of artworks or music can not only assist the artists with their work, but also help people understand the creative process of art.

The problem of automatic audio generation with machine learning belongs to the field of Algorithmic Composition, which is the technique of using algorithms to create music. The application of neural networks to audio generation started in the late 80's by Lewis and Todd. On the one hand, Todd used a Jordan auto-regressive neural network to generate audio sequentially [12]. Many approaches still use the same idea throughout years, such as LSTMs proposed by Eck and Schmidhuber [3]. On the other hand, Lewis used a multi-layer perceptron for algorithmic composition [6]. While these approaches are still valid nowadays, with the poor performance of computational devices back then, they didn't succeed. One of the failure was that the algorithmically composed music lacked global coherence or structure.

Most of the approaches to algorithmic composition use symbolic music representations such as MIDI files. In 2002, the first approach of processing music in the format of raw-audio was taken [8], and studies have shown that waveform-based models can outperform spectrogram-based ones provided that enough training data is available [9].

With the Generative Adversarial Network [4], which makes it possible to train deep neural networks, coming up in 2014, more and more approaches are utilizing GANs in recent studies of algorithmic composition problems.

1.2 Challenges

Input data lacks features. Using the one dimensional raw audio as input data means that we only have one feature at each time step, while symbolic representations of audio such as MIDI file can have several different features. So it will be harder to train a neural net with raw audio.

GANs are hard to train. The model may never converge, and it may even collapse. The discriminator and generator may get unbalance, causing overfitting.

"Checkerboard" artifacts. The CNN generator is known to produce fake samples with artifacts, which is hard for 1D CNN discriminator to classify.

Preprint. Work in progress.

1.3 Contribution

In this report, I will be presenting several combinations of discriminator and loss function of audio generation GANs. The training of my model is based on raw audio of piano music. It is evaluated with inception scores comparing with the proposed trained generator of a recent paper which uses the same input data format as mine.

2 Related Work

The Generative adversarial networks (GANs) [4] are a class of neural network architectures designed with the aim of generating realistic data. The approach involves training two neural models with conflicting objectives, a generator (G), and a discriminator (D), forcing each other to improve. The G net tries to generate fake samples that looks real, while the D net tries to discriminate the fake samples between real samples. The GAN is a zero sum game, and using this framework makes it possible to train deep generative networks. There are many potential advantages of GANs in automatic audio generation. For example, GANs are good at data augmentation and could enable rapid and straightforward sampling of large amounts of audio.

A recent paper proposed WaveGAN [2], which utilizes the image generation model Deep Convolutional GAN [10], to synthesis audio with raw audio as input. It treats audio as one-dimensional continuous sequential data, and uses 1D convolutional layers in both generator and discriminator instead of the traditional RNNs. It has been proven in the paper that WaveGAN outruns the naive approach, which treat input audio data as image-like spectrograms, in qualitative human judgements.

Recurrent neural networks (RNNs) are often used to model sequences of data. They model the conditional probability of the next token given the sequence of preceding tokens. By sampling from this conditional distributions, one can generate reasonably realistic sequences. Moreover, using the Long short-term memory (LSTM) units [5] in RNNs can even help to get a better result. RNNs with LSTM cells are well-suited to classifying, processing and making predictions based on time series data, and thus are suitable when dealing with music data.

3 Model

My model bases on WaveGAN. It takes TFRecord files as input and saves its generative model for evaluating inception score. The generator uses several convolutional layers, which is similar to WaveGAN. The discriminator is modified to use RNNs and LSTM cells in order to overcome the artifacts problem of CNN discriminator. The loss functions I choose for this model are LSGAN and WGAN, and I use the Adam optimizer to train it.

3.1 Generator

The generator 1 takes in a vector of uniformly distributed random numbers between -1 and 1. It applies a full connection layer to the input and followed by 5 convolution layer. The generator outputs a 1D wave form vector as the fake sample. This structure is same as that of WaveGAN which inherits from DCGAN. The reason for having a $[16384, 1]$ shape output is because the generator is synthesizing audio of about 1 sec long for testing, and the rate of the audio is at 16000 while 16384 is a power of 2 which is close to 16000.

3.2 Discriminator

WaveGAN also suffers from the "checkerboard" artifacts produced by convolutional layers in the generator. Unlike image generative models, the discriminator using convolutional layers in audio models with 1D input fails to classify fake samples with "checkerboard" artifacts. The paper proposes a way of randomly shuffling the phase of the input audio data to solve this problem.

I propose a different discriminator structure 2 that aims to deal with the artifacts using RNNs with LSTM cells. LSTM cells are well-suited for classifying time based sequences, and thus should out

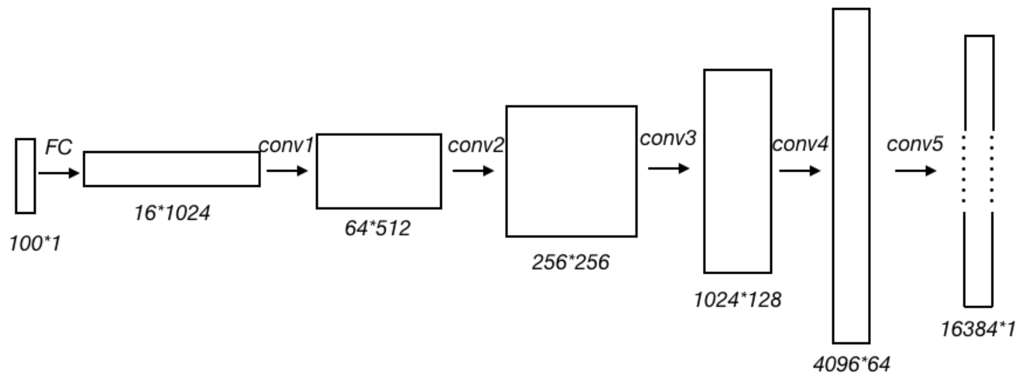


Figure 1: The CNN generator

perform convolutional layers in classifying fake samples with artifacts. Because the input vector of the discriminator is too big in time size while lacks feature at each time step, I first reshape the input vector to "compress" its length, and then feed it to RNN layers with LSTM cells. Finally I use down convolution and full connection layers to get the scalar output.

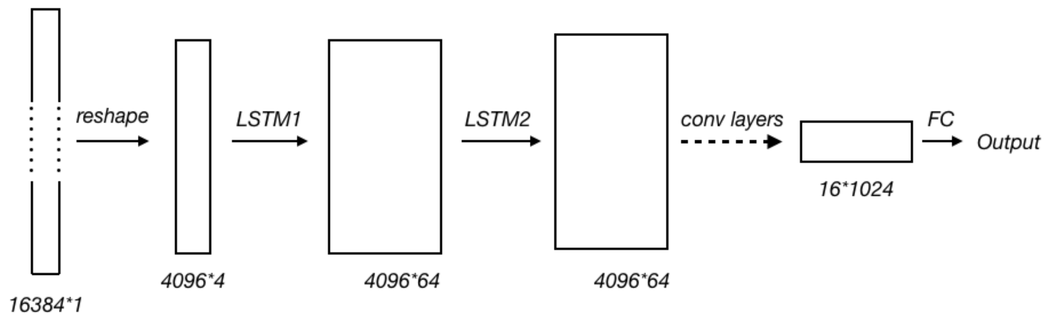


Figure 2: One example of the LSTM discriminator. The tested discriminators typically contain one to two LSTM-RNN layers and zero to five CNN layers

3.3 Loss Function

The original GAN learns mapping from random vectors $z \in Z$, i.i.d. samples from known prior P_Z, to data points in the space of real data set X. The generator $G : Z \longmapsto X$ is pitted against the discriminator $D : X \longmapsto [0, 1]$. The D net is trained to determine the probability of whether a data point x belongs to the real dataset X, while G is trained generate fake data points in X. That is say, G is trained to minimize the following value, while D is trained to maximize the value function.

$$\min_G \max_D V(D, G) = E_{x \sim P_X}[\log D(x)] + E_{z \sim P_Z}[\log(1 - D(G(z)))]$$

In my model, the loss functions I choose are LSGAN and WGAN. In fact, it should be WGAN-GP, a recently proposed loss function that enhances training stability. However, the RNN layer in Tensorflow built-in library does not support WGAN-GP.

The LSGAN [7] optimizes based on square error loss function. It is designed to help the generator better. Mathematically, it is defined as

$$\min_D V_{LSGAN}(D) = \frac{1}{2} E_{x \sim p_{data}(x)}[(D(x) - b)^2] + \frac{1}{2} E_{z \sim p_z(z)}[(D(G(z)) - a)^2]$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2} E_{z \sim p_z(z)}[(D(G(z)) - c)^2]$$

where a and b are the target discriminator labels for the generated images and the real images, and c is target generator labels for the generated images. Intuitively, LSGAN wants the target discriminator label for real images to be 1 and generated images to be 0. And for the generator, it wants the target label for generated images to be 1.

WGAN [1] uses Wasserstein distance in its loss function defined as

$$V_{WGAN}(D_w, G) = E_{x \sim P_x}[D_w(x)] - Ez \sim P_z[D_w(G(z))]$$

It has no sign of mode collapse in experiments, and the generator can still learn when the discriminator perform well.

4 Experiment

4.1 Dataset

The dataset is the raw-audio data of piano music from the WaveGAN paper. It is consist of TFRecord-format piano music pieces of Bach played by the same performer. The data loader returns a [batch size,16384, 1] tensor to the discriminator as the input of real samples.

4.2 Baseline

The base line of this model is the WaveGAN with its default setting but using WGAN as loss function because WGAN-GP can't be used in RNN GANs as mentioned above. The native approach which treats raw audio data as spectrogram images (called SpecGAN) is considered as another baseline.

4.3 Evaluation Method

Salimans propose the inception score [11], which uses a pre-trained inception classifier to measure both the diversity and discriminability of generated images, finding that the measure correlates well with human judgement. To measure inception score, I use the trained audio classifier proposed by the WaveGAN paper.

4.4 Quantitative Results

Results for the evaluation appear in the comparison table 1. It shows that the naive model preforms the worst. The baseline WaveGAN's performance is similar to some combinations of models using LSTM, but outran by the best LSTM model slightly. This performance of WaveGAN is worse than that in the WaveGAN paper mainly because of using WGAN.

Experiments	mean	std	epoch to reach
SpecGAN (baseline)	1.028	0.009	204
WaveGAN with WGAN (baseline)	1.076	0.005	52
pure LSTM D-net with WGAN	1.093	0.012	319
LSTM & conv D-net with WGAN	1.084	0.031	197
LSTM & conv D-net with LSGAN	1.085	0.017	243

Table 1: The result comparison table of the inception score of several combinations of discriminators and loss functions, other combinations don't converge or have unbalance generator loss and discriminator loss

4.5 Analysis

With the result in table 1, even the best of my model only runs the baseline WaveGAN with WGAN for 1%. Because of the lack of powerful machine, some of models still need more epochs of training.

Although the result cannot prove that LSTM discriminator beats CNN discriminator, at least it has shown that LSTM discriminator in WaveGAN worth a better try.

The model I propose from table 1 is LSTM & conv D-net with LSGAN. Although its inception score is not the best, it is more stable in training and has its generator loss and discriminator loss in balance. The architecture description and training hyperparameters of this proposed model are listed in tables 2, 3 and 4. In these tables, n is the batch size.

Table 2: WaveGAN generator

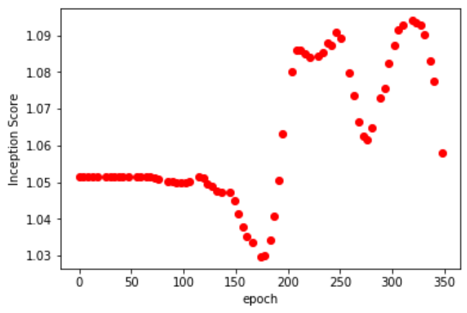
Operation	Kernel Size	Output Shape
Input $z \sim Uniform(-1, 1)$		$(n, 100)$
Dense 1	$(100, 16384)$	$(n, 16384)$
Reshape		$(n, 16, 1024)$
ReLU		$(n, 16, 1024)$
Trans Conv1D (Stride=4)	$(25, 1024, 512)$	$(n, 64, 512)$
ReLU		$(n, 64, 512)$
Trans Conv1D (Stride=4)	$(25, 512, 256)$	$(n, 256, 256)$
ReLU		$(n, 256, 256)$
Trans Conv1D (Stride=4)	$(25, 256, 128)$	$(n, 1024, 128)$
ReLU		$(n, 1024, 128)$
Trans Conv1D (Stride=4)	$(25, 128, 64)$	$(n, 4096, 64)$
ReLU		$(n, 4096, 64)$
Trans Conv1D (Stride=4)	$(25, 64, 1)$	$(n, 16384, 1)$
Tanh		$(n, 16384, 1)$

Table 3: LSTM discriminator

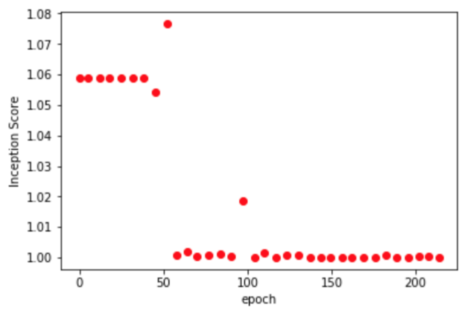
Operation	Kernel Size	Output Shape
Input x or $G(z)$		$(n, 16384, 1)$
Reshape		$(n, 4096, 4)$
LSTM (unit size = 64)		$(n, 4096, 64)$
LSTM (unit size = 64)		$(n, 4096, 64)$
ReLU		$(n, 4096, 64)$
Conv1D (Stride=4)	$(25, 64, 128)$	$(n, 1024, 128)$
ReLU		$(n, 1024, 128)$
Trans Conv1D (Stride=4)	$(25, 128, 256)$	$(n, 256, 256)$
ReLU		$(n, 256, 256)$
Trans Conv1D (Stride=4)	$(25, 256, 512)$	$(n, 64, 512)$
ReLU		$(n, 64, 512)$
Trans Conv1D (Stride=4)	$(25, 512, 1024)$	$(n, 16, 1024)$
ReLU		$(n, 16, 1024)$
Reshape		$(n, 16384)$
Dense	$(16384, 1)$	$(n, 1)$

Table 4: WaveGAN hyperparameters

Name	Value
Input data type	16-bit PCM (requantized to 32-bit float)
Model data type	32-bit floating point
Num channels	1
Batch size (b)	64
Model dimensionality	64
LSTM cell hidden unit size	64
Loss	LSGAN
D updates per G update	5
Optimizer	$Adam(\alpha = 1e-4, \beta_1 = 0.5, \beta_2 = 0.9)$



(a) pure LSTM with WGAN model



(b) WaveGAN with WGAN model

Figure 3: epoch to inception scores

Another thing worth analyzing is about whether or not using LSTM end classification in the 1D audio case. Several combinations of discriminators and loss functions in my experiments that fail show that, no matter how the convolutional layers in the discriminator and the type of loss functions change, the training will converge in few epochs with a low generator and discriminator loss. The LSTM RNN layers output tensor with size $[batch\_size, max\_time, channels]$, and using end classification means that most of the neuron units are dropped. This caused the training to end early. So I do not propose to use the end classification technique in my model.

4.6 Future Work

As I mentioned a lot times above, the training of WaveGAN takes huge amount of time and computational power, which I underestimated when I pick this topic. Even the model with the best inception score and the proposed model in WaveGAN paper with WGAN turn out to be kind of unstable 3. So, if there is a chance, I would like to continue the training on a more powerful machine.

The other thing that needs future efforts is about using WGAN-GP in RNNs. In my experiments, many models suffer from the unbalance and unstable loss of both generator and discriminator. It is very likely that choosing WGAN-GP as the loss function will help solve this problem. Although the RNN layers of Tensorflow does not support WGAN-GP, it is possible to construct a RNN layer rather than using the one provided by the Tensorflow library to train the model.

Reference

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis, 2018.

[3] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE, 2002.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] JP Lewis. Creation by refinement: A creativity paradigm for gradient descent learning networks. In *International Conf. on Neural Networks*, pages 229–233, 1988.

[7] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[8] Matija Marolt, Alenka Kavcic, and Marko Privosnik. Neural networks for note onset detection in piano music. In *Proceedings of the 2002 International Computer Music Conference*, 2002.

[9] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M Schmidt, Andreas F Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*, 2017.

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[12] Peter Todd. A sequential network design for musical applications. In *Proceedings of the 1988 connectionist models summer school*, pages 76–84, 1988.