Generating Music with GANs

An Overview and Case Studies

November 4th, 2019

Hao-Wen Dong
UC San Diego & Academia Sinica

Yi-Hsuan Yang
Taiwan AI Labs & Academia Sinica

salu133445.github.io/ismir2019tutorial/
Outline

Section 1
Overview of music generation research

Section 2
Introduction to GANs
Coding session 1: GAN for images

Section 3
Case studies of GAN-based systems (I)
Coding session 2: GAN for piano rolls
Case studies of GAN-based systems (II)

Section 4
Current limitations
Future research directions

salu133445.github.io/ismir2019tutorial/
About us

Hao-Wen Dong
• Ph.D. student, UC San Diego (2019-)
• Research intern, Yamaha Corporation (2019)
• Research assistant, Academia Sinica (2017-2019)
• First author of MuseGAN & BMuseGAN

Yi-Hsuan Yang
• Chief Music Scientist, Taiwan AI Labs (2019-)
• Research professor, Academia Sinica (2011-)
• Ph.D., National Taiwan University (2006-2010)
• Associate Editor of IEEE TMM and TAFFC (2017-2019)
• Program Chair of ISMIR @ Taipei, Taiwan (2014)
• Tutorial speaker of ISMIR (2012 last time)
About the Music and AI Lab @ Sinica

• About Academia Sinica
  • National academy of Taiwan, founded in 1928 (not a university)
  • About 1,000 full, associate, assistant research professors

• About Music and AI Lab
  • [https://musicai.citi.sinica.edu.tw/](https://musicai.citi.sinica.edu.tw/)
  • Since Sep 2011
  • Members
    • PI [me]
    • research assistants
    • PhD/master students
  • 3 AAAI full papers + 3 IJCAI full papers in 2018 and 2019
About the Music Team @ Taiwan AI Labs

• About Taiwan AI Labs
  • [https://ailabs.tw/](https://ailabs.tw/)
  • Privately-funded research organization (like openAI), founded in 2017
  • Three main research area: 1) HCI, 2) medicine, 3) smart city
  • 100+ employees (late 2019)

• About the “Yating” Music AI team
  • Members
    • scientist [me]
    • ML engineers (for models)
    • musicians
    • program manager
    • software engineers (for frontend/backend)
Outline

Section 1
Overview of music generation research

Section 2
Introduction to GANs

Section 3
Case studies of GAN-based systems

Section 4
Current limitations
Future research directions
From a music production view

- **Composing / songwriting**
  - Melody
  - Chords
  - Lyrics

- **Arranging**
  - Instrumentation
  - Structure

- **Mixing**
  - Timbres/tones
  - Balancing
Use cases of music AI

• Make musicians’ life easier
  • Inspire ideas
  • Suggest continuations, accompaniments, lyrics, or drum loops
  • Suggest mixing presets

• Empower everyone to make music
  • Democratization of music creation

• Create copyright free music for videos or games

• Music education (e.g., auto-accompaniment)
Companies involved in automatic music generation

- Google Magenta
- IBM Watson Beat AI
- Jukedeck
- amper
- AIVA
- Spotify
- Flow Machines
- AI Labs.tw
- Sony CSL

... and more!
Demo: Magenta Studio

https://magenta.tensorflow.org/studio/
Demo: Jamming with Yating

• https://www.youtube.com/watch?v=9ZIJrr6lmHg

• Jamming with Yating [1,2]
  • Input (by human): piano
  • Output: piano + bass + drum

From a deep learning view

• Input representation
• Model
• Output representation
Models

• Rule based methods
• Concatenation based methods
• Machine learning based methods
  • VAE: variational autoencoder
  • GAN: generative adversarial network

• See Section 2: Introduction to GANs
Why GAN?

- **State-of-the-art model** in:
  - Image generation: BigGAN [1]
  - Text-to-speech audio synthesis: GAN-TTS [2]
  - Note-level instrument audio synthesis: GANSynth [3]
  - Also see ICASSP 2018 tutorial: “GAN and its applications to signal processing and NLP” [4]

- Its potential for music generation has not been fully realized

- **Adversarial training has many other applications**
  - For example, source separation [5], domain adaption [6], music transcription [7]

---

Input/output representations

• **Symbolic output**
  • Piano rolls
  • MIDI events
  • Score

• **Audio output**
  • Spectrogram
  • Waveform
I/O representations

• Symbolic output
  • Piano rolls (image-like): easier for GANs to work with
  • MIDI events (text-like)
  • Score (hybrid)

• Audio output
  • Spectrogram (image-like)
  • Waveform
I/O representations

• Symbolic output
  • Piano rolls (image-like):
    MidiNet [1], MuseGAN [2]
  • MIDI events (text-like):
    Music Transformer [3], MuseNet [4]
  • Scores (hybrid):
    Thickstun [5], measure-by-measure [6]

[3] https://openreview.net/pdf?id=rJe4ShAcF7, ICLR 2019
Scope of music generation

• **Generation from scratch**
  • X $\rightarrow$ melody
  • X $\rightarrow$ piano roll
  • X $\rightarrow$ audio

• **Conditional generation**
  • melody $\rightarrow$ piano roll (*accompaniment*)
  • piano roll $\rightarrow$ audio (*synthesis*)
  • piano roll $\rightarrow$ piano roll‘ (*rearrangement*)
  • audio $\rightarrow$ audio‘

• **See Section 3: Case studies**
  • and, [https://github.com/affige/genmusic_demo_list](https://github.com/affige/genmusic_demo_list)
We will talk about

1. **Symbolic melody generation**: MidiNet [1], SSMGAN [2]
2. **Arrangement generation**: MuseGAN [3], BinaryMuseGAN [4], LeadSheetGAN [5]
3. **Style transfer**: CycleGAN [6], TimbreTron [7], Play-as-you-like [8], CycleBEGAN [9]

[4] “Convolutional GANs with binary neurons for polyphonic music generation,” *ISMIR* 2018
[8] “Play as You Like: Timbre-enhanced multi-modal music style transfer,” *AAAI* 2019
[9] “Singing style transfer using cycle-consistent boundary equilibrium GANs,” *ICML workshop* 2018
Outline

Section 1
Overview of music generation research

Section 2
Introduction to GANs

Section 3
Case studies of GAN-based systems

Section 4
Current limitations
Future research directions
What is a GAN?

A generative model and a deep neural network play an adversarial game between two competitors.
A loss function for training generative models

A loss function for training generative models

Generator
Make $G(z)$ indistinguishable from real data for $D$

Discriminator
Tell $G(z)$ as fake data from $x$ being real ones

$z \sim p_z$ $\rightarrow G \rightarrow G(z)$

fake samples

$D$ $\rightarrow$ $1/0$

real samples

$\log(1-D(G(z)))$

$\log(1-D(x)) + \log(D(G(z)))$

Problems of unregularized GANs

- **Key**—discriminator provides generator with gradients as a guidance for improvement
  - Discrimination is easier than generation
  - Discriminator tends to provide large gradients
  - Result in unstable training of the generator

- Common failure cases
  - Mode collapse
  - Missing modes

(Colors show the outputs of the discriminator)
Regularizing GANs

Advantages of gradient regularization

- provide a smoother guidance to the generator
- alleviate mode collapse and missing modes issues

Unregularized

Locally regularized

Globally regularized

Coding session I—GAN for images

Google Colab notebook link
https://colab.research.google.com/drive/1Cnq9z3QvxIsVntlXKjPjbwttxeDH47Xl

You can also find the link on the tutorial website
https://salu133445.github.io/ismir2019tutorial/

Click [Open in playground]
Deep convolutional GAN (DCGAN)

# GANs vs VAEs

<table>
<thead>
<tr>
<th>Objective</th>
<th>GAN</th>
<th>VAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(generator) fool the discriminator (discriminator) tell real data from fake ones</td>
<td>reconstruct real data using pixel-wise loss</td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td>GAN: tend to be sharper</td>
<td>VAE: tend to be more blurred</td>
</tr>
<tr>
<td>Diversity</td>
<td>GAN: Higher</td>
<td>VAE: Lower</td>
</tr>
<tr>
<td>Stability</td>
<td>GAN: Lower</td>
<td>VAE: Higher</td>
</tr>
</tbody>
</table>

Larsen et al., “Autoencoding beyond pixels using a learned similarity metric,” ICML 2016
State of the arts—BigGANs

[Colab notebook demo]

Brock et al., “Large scale GAN training for high fidelity natural image synthesis,” ICLR 2019
Interpolation on the latent space

Brock et al., “Large scale GAN training for high fidelity natural image synthesis,” ICLR 2019
Conditional GAN (CGAN)

**Conditional GAN (CGAN)**

![Diagram of Conditional GAN (CGAN)]

- **Random noise**: $z \sim p_Z$
- **Conditions**: $y \sim p_Y$
- **Generator**: $G\left(z, y\right)$
  - **Fake samples**: $G\left(z, y\right)$
  - **Real samples**: $\left(x, y\right) \sim p_{X,Y}$
- **Discriminator**: $D$
  - **Output**: $1/0$

---

Conditional GAN (CGAN)

Generator now generate samples based on some conditions

Discriminator now examine whether a pair \((x, y)\) or \((G(z), y)\) is real or not

Key—Feed conditions to both G and D

Conditional GAN—Samples

Dog

Cat

Tiger
pix2pix

Key—Use pixel-wise loss for supervisions

Isola et al., “Image-to-image translation with conditional adversarial nets,” CVPR 2017
pix2pix—Samples

Isola et al., “Image-to-image translation with conditional adversarial nets,” CVPR 2017
Cycle-consistent GAN (CycleGAN)

Cycle-consistent GAN (CycleGAN)

Cycle-consistent GAN (CycleGAN)

Cycle-consistent loss

Key—Use cycle-consistency loss for supervisions

CycleGAN

pix2pix

Unpaired samples in two domains

Require paired \((x, y)\) samples

Isola et al., “Image-to-image translation with conditional adversarial nets,” CVPR 2017
**CycleGAN—Samples**

(\textit{pix2pix}) \textit{We do not need a Monte’s version for each photo }\rightarrow \textit{ hard to acquire}

(CycleGAN) \textit{We only need a collection of Monet’s paintings and a collection of photos }\rightarrow \textit{ easier to acquire}

The many other GANs ... and more!

<table>
<thead>
<tr>
<th>Training criterions</th>
<th>Optimization constraints</th>
<th>Training strategies</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSGAN</td>
<td>WGAN</td>
<td>UnrolledGAN</td>
<td>DCGAN</td>
</tr>
<tr>
<td>WGAN</td>
<td>WGANGP</td>
<td>LAPGAN</td>
<td>InfoGAN</td>
</tr>
<tr>
<td>EBGAN</td>
<td>McGAN</td>
<td>StackedGAN</td>
<td>CGAN</td>
</tr>
<tr>
<td>BEGAN</td>
<td>MMDGAN</td>
<td>StackGAN</td>
<td>ACGAN</td>
</tr>
<tr>
<td>GeometricGAN</td>
<td>FisherGAN</td>
<td>PGGAN</td>
<td>pix2pix</td>
</tr>
<tr>
<td>RaGAN</td>
<td>DRAGAN</td>
<td>StyleGAN</td>
<td>CycleGAN</td>
</tr>
<tr>
<td></td>
<td>SNGAN</td>
<td>BigGAN</td>
<td>CoGAN</td>
</tr>
</tbody>
</table>

See more at https://github.com/hindupuravinash/the-gan-zoo
Open questions about GANs

• [https://distill.pub/2019/gan-open-problems/](https://distill.pub/2019/gan-open-problems/)

1. What are the trade-offs between GANs and other generative models?
2. What sorts of distributions can GANs model?
3. How can we Scale GANs beyond image synthesis?
4. What can we say about the global convergence of the training dynamics?
5. How should we evaluate GANs and when should we use them?
6. How does GAN training scale with batch size?
7. What is the relationship between GANs and adversarial examples?
Comparative studies on GANs


Generating Music with GANs

Section 1
Overview of music generation research

Section 2
Introduction to GANs

Section 3
Case studies of GAN-based systems

Section 4
Current limitations & Future research directions

Or try the next Google Colab notebook
https://colab.research.google.com/drive/1WrFtqo5LW8QfhiuhHmge9QLexWwS2BcM
Outline

Section 1
Overview of music generation research

Section 2
Introduction to GANs

Section 3
Case studies of GAN-based systems

Section 4
Current limitations
Future research directions
Scope of music generation

1. Symbolic melody generation
   • $X \rightarrow$ melody

2. Arrangement generation
   • $X \rightarrow$ piano roll
   • melody $\rightarrow$ piano roll

3. Style transfer
   • piano roll $\rightarrow$ piano roll’
   • audio $\rightarrow$ audio’

4. Audio generation
   • $X \rightarrow$ audio
   • piano roll $\rightarrow$ audio
## Some symbolic datasets to starts with

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th>Format</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikifonia</td>
<td>lead sheet</td>
<td>XML</td>
<td><a href="http://www.wikifonia.org">www.wikifonia.org</a></td>
</tr>
<tr>
<td>HookTheory</td>
<td>lead sheet</td>
<td>XML</td>
<td><a href="http://www.hooktheory.com/theorytab">www.hooktheory.com/theorytab</a></td>
</tr>
<tr>
<td>Lakh MIDI Dataset</td>
<td>multitrack</td>
<td>MIDI</td>
<td><a href="https://colinraffel.com/projects/lmd">https://colinraffel.com/projects/lmd</a></td>
</tr>
<tr>
<td>Lakh Pianoroll Dataset</td>
<td>multitrack</td>
<td>npz</td>
<td>salu133445.github.io/lakh-pianoroll-dataset</td>
</tr>
<tr>
<td>Groove MIDI Dataset</td>
<td>drum</td>
<td>MIDI</td>
<td>magenta.tensorflow.org/datasets/groove</td>
</tr>
<tr>
<td>Midi Man</td>
<td>drum</td>
<td>MIDI</td>
<td><a href="http://www.reddit.com/r/WeAreTheMusicMakers/comments/3anwu8/the_drum_percussion_midi_archive_800k/">www.reddit.com/r/WeAreTheMusicMakers/comments/3anwu8/the_drum_percussion_midi_archive_800k/</a></td>
</tr>
</tbody>
</table>

(Only datasets with miscellaneous genres are presented here)

See more at https://github.com/wayne391/symbolic-musical-datasets
GANs for Symbolic Melody Generation

\[ X \rightarrow \text{melody} \]
Key—Design CNN kernel sizes to match our understanding of music

MidiNet—Samples

1D + 2D conditions
(previous bar and chords)

1D condition only
(chords only)

More samples can be found at https://richardyang40148.github.io/TheBlog/midinet_arxiv_demo.html
Jhamtani and Berg-Kirkpatrick, “Modeling self-repetition in music generation using structured adversaries,” *ML4MD* 2019
Jhamtani and Berg-Kirkpatrick, “Modeling self-repetition in music generation using structured adversaries,” ML4MD 2019
Key—Use the SSM discriminator to improve global musical structure

SSM discriminator (all measures)

LSTM discriminator (window of K measures)

Jhamtani and Berg-Kirkpatrick, “Modeling self-repetition in music generation using structured adversaries,” *ML4MD* 2019
SSMGAN—Samples

Jhamtani and Berg-Kirkpatrick, “Modeling self-repetition in music generation using structured adversaries,” ML4MD 2019
Other GAN models for melody generation

• **C-RNN-GAN** [1]: use RNNs for the generator and discriminator
• **JazzGAN** [2]: given chords, compose melody; compare 3 representations
• **Conditional LSTM-GAN** [3]: given lyrics, compose melody
• **SSMGAN** [4]: use GAN to generate a self-similarity matrix (SSM) to represent self-repetition in music, and then use LSTM to generate melody given the SSM

---

GANs for Symbolic Arrangement Generation

X → piano roll
melody → piano roll
Challenges of arrangement generation

• Temporal evolution
  • Dynamics, emotions, tensions, etc.

• Structure (temporal)
  • Short-term structure → can somehow be generated with special models
  • Long-term structure → super hard

• Instrumentation
  • Multiple tracks
  • Functions of instruments

Couple with one another in a complex way in real world music
Why pianorolls?

• Deep learning friendly format \(\rightarrow\) basically matrices
• Easier for GANs to work with
Pros and cons of pianorolls

• Pros
  • Can be purely symbolic $\rightarrow$ quantized by beats (or factors of beats)
  • Repetition and structure can be observed easily
  • No need to serialize polyphonic compositions

• Cons
  • Memory inefficient $\rightarrow$ mostly zero entries (i.e., sparse matrices)
  • Missing the concepts of “notes”
  • Hard to handle performance-level information unless using high resolution
Multitrack pianoroll

Drums  Guitar  Piano  Bass  Strings
MuseGAN—Generator

Dong et al., “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” AAAI 2018
MuseGAN—Generator

Dong et al., “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” AAAI 2018
**MuseGAN—Generator**

Dong et al., “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” AAAI 2018
**MuseGAN—Generator**

Key—Use different types of latent variables to enhance controllability

<table>
<thead>
<tr>
<th>Time</th>
<th>Dependent</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track</td>
<td>Dependent</td>
<td>Melody</td>
</tr>
<tr>
<td></td>
<td>Independent</td>
<td>Groove</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chords</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Style</td>
</tr>
</tbody>
</table>

Dong et al., “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” AAAI 2018
MuseGAN—Samples

<table>
<thead>
<tr>
<th>Drums</th>
<th>Piano</th>
<th>Guitar</th>
<th>Bass</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Sample 1]</td>
<td>![Sample 2]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Drum patterns

Chords

Bass line

More samples can be found at https://salu133445.github.io/musegan/results

Dong et al., “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” AAAI 2018
Before the coding session...

**LPD (Lakh Pianoroll Dataset)**
- 174,154 multi-track piano-rolls
- Derived from Lakh MIDI Dataset*
- Mainly pop songs
- Derived labels available

**Pypianoroll (Python package)**
- Manipulation & Visualization
- Efficient I/O
- Parse/Write MIDI files
- On PYPI (pip install pypianoroll)

_We will use them in the next coding session!_

---

[Pypianoroll] https://salu133445.github.io/pypianoroll
[Lakh Pianoroll Dataset] https://salu133445.github.io/lakh-pianoroll-dataset
Coding session II—GAN for pianorolls

Google Colab notebook link
https://colab.research.google.com/drive/1WrFtqo5LW8QfhiuhHmge9QLexWwS2BcM

You can also find the link on the tutorial website
https://salu133445.github.io/ismir2019tutorial/

Click [Open in playground]
Dong and Yang, “Convolutional generative adversarial networks with binary neurons for polyphonic music generation,” ISMIR 2018
**BinaryMuseGAN**

• use *binary neurons* at the output layer of the generator

• use *straight-through estimator* to estimate the gradients for the binary neurons (which involves nondifferentiable operations)

<table>
<thead>
<tr>
<th>Generator’s outputs</th>
<th>Real data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MuseGAN</td>
<td>real-valued</td>
</tr>
<tr>
<td>BinaryMuseGAN</td>
<td>binary-valued</td>
</tr>
</tbody>
</table>

Dong and Yang, “Convolutional generative adversarial networks with binary neurons for polyphonic music generation,” *ISMIR* 2018
Liu and Yang, “Lead sheet generation and arrangement by conditional generative adversarial network,” ICMLA 2018
Liu and Yang, “Lead sheet generation and arrangement via a hybrid generative model,” ISMIR-LBD 2018
Liu and Yang, “Lead sheet generation and arrangement by conditional generative adversarial network,” ICMLA 2018
Liu and Yang, “Lead sheet generation and arrangement via a hybrid generative model,” ISMIR-LBD 2018
Liu and Yang, “Lead sheet generation and arrangement by conditional generative adversarial network,” ICMLA 2018
Liu and Yang, “Lead sheet generation and arrangement via a hybrid generative model,” ISMIR-LBD 2018
LeadSheetGAN

Key—First generate the lead sheet, then the arrangement

Stage 1: Lead sheet Generation

Stage 2

Stage 3: Arrangement Generation

(Unconditional) GAN

Conditional GAN

Liu and Yang, “Lead sheet generation and arrangement by conditional generative adversarial network,” ICMLA 2018
Liu and Yang, “Lead sheet generation and arrangement via a hybrid generative model,” ISMIR-LBD 2018
LeadSheetGAN—Samples

Marron 5
Payphone

latent space interpolation

The Beatles
Hey Jude

Liu and Yang, “Lead sheet generation and arrangement by conditional generative adversarial network,” ICMLA 2018
Liu and Yang, “Lead sheet generation and arrangement via a hybrid generative model,” ISMIR-LBD 2018
LeadSheetGAN—Samples

Arrangement generation given an “Amazing Graze” lead sheet

More samples can be found at https://liuhaumin.github.io/LeadsheetArrangement/results
Liu and Yang, “Lead sheet generation and arrangement by conditional generative adversarial network,” ICMLA 2018
Generating Music with GANs

Section 1
Overview of music generation research

Section 2
Introduction to GANs

Section 3
Case studies of GAN-based systems

Section 4
Current limitations & Future research directions
**Scope of music generation**

1. **Symbolic melody generation**
   - $X \rightarrow$ melody

2. **Arrangement generation**
   - $X \rightarrow$ piano roll
   - melody $\rightarrow$ piano roll

3. **Style transfer**
   - piano roll $\rightarrow$ piano roll’
   - audio $\rightarrow$ audio’

4. **Audio generation**
   - $X \rightarrow$ audio
   - piano roll $\rightarrow$ audio
CycleGANs for Music Style Transfer

piano roll $\rightarrow$ piano roll’

audio $\rightarrow$ audio’
Music style transfer

• Alter the “style,” but keep the “content” fixed
• Three types of music style transfer [1]
  1. composition style transfer for score
  2. performance style transfer for performance control
  3. timbre style transfer for sound
• Little existing work on performance style transfer (e.g., [2]) uses deep learning

Composition style transfer (musical genre)

• Example: [https://www.youtube.com/watch?v=buXqNqBFd6E](https://www.youtube.com/watch?v=buXqNqBFd6E)

• Re-orchestrations of Beethoven's *Ode to Joy* by a collaboration between human and AI (Sony CSL Flow Machines)

---

Composition style transfer (musical genre)

• Transfer among Classic, Jazz, and Pop [1,2]

• Model: standard convolutional CycleGAN

• I/O representation: single-track piano roll (64 x 84)
  • merge all notes of all tracks (except for drums) into a single track
  • discard drums
  • 7 octaves (C1-C8; hence 84 notes)
  • 4/4 time signature, 16 time steps per bar, 4 bars as a unit (hence 64 steps)
  • 10k+ four-bar phrases for each genre (no paired data)

Recap—CycleGAN

• Adversarial loss + identity mapping loss ("cycle consistency")

Composition style transfer (musical genre)

• https://www.youtube.com/channel/UCs-bI_NP7PrQaMV1AJ4A3HQ

• Possible extensions:
  • Consider different voices (including drums) separately (instead of merging them)
  • Add recurrent layers to better model sequential information
  • Identify the melody line [1] and take better care of it

• Related:
  • Supervised genre style transfer (not using GANs) using synthesized data [2]

**Timbre style transfer (instrumental sounds): TimbreTron**

- Transfer among *piano, flute, violin, harpsichord* solos
  

- Model: modified version of **CycleGAN**

- I/O representation: 4-second **CQT** (257 x 251)

---

Huang et al., “TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) pipeline for musical timbre transfer,” *ICLR 2019*
Drawback of cycleGAN

• Can work for only two domains at a time
  • For example, piano↔flute, piano↔violin, piano↔harpsichord, etc
MUNIT instead of cycleGAN

- **MUNIT** [1]: an advanced version of cycleGAN that incorporates encoders/decoders to get disentangled “content” and “style” codes
- Can work for **multiple domains** at the same time

---

[1] Huang et al., “Multimodal unsupervised image-to-image translation,” ECCV 2018
Timbre style transfer (instrumental sounds): Play-as-you-like

- Transfer among piano, guitar solos, and string quartet (https://tinyurl.com/y23tvhjx)
- Use MUNIT
- I/O representation: Mel-spectrogram + spectral difference + MFCC + spectral envelope
  - Cycle consistency among the channel-wise features (as regularizers)
- Style interpolation: https://soundcloud.com/affige/sets/ismir2019-gan-tutorial-supp-material

Lu et al., “Play as You Like: Timbre-enhanced multi-modal music style transfer,” AAAI 2019
Timbre style transfer (singing): CycleBEGAN

Wu et al., “Singing style transfer using cycle-consistent boundary equilibrium generative adversarial networks,” ICML workshop 2018
Timbre style transfer (singing): CycleBEGAN

• Transfer between female and male singing voices [1]
  [http://mirlab.org/users/haley.wu/cybegan/](http://mirlab.org/users/haley.wu/cybegan/) (check the outside test result)

• Model: **BEGAN** instead of GAN [2]
  • train the generator $G$ such that the discriminator $D$ (an encoder/decoder net) would reconstruct fake data as nicely as real data
  • have a mechanism to balance the power of $G$ and $D$

  + **skip connections** (also used in [3]) + a **recurrent layer**

---

**Timbre style transfer (singing): CycleBEGAN**

- **Skip connections** contribute to sharpness, lyrics intelligibility, and naturalness.
- **Recurrent layers** further improves everything, especially pitch accuracy.

Wu et al., “Singing style transfer using cycle-consistent boundary equilibrium generative adversarial networks,” *ICML workshop 2018*
Timbre style transfer (singing): CycleBEGAN

Also use an encoder/decoder architecture for the generator

“Singing style transfer using cycle-consistent boundary equilibrium generative adversarial networks,” ICML works. ’18
GANs for Music Audio Generation

X $\rightarrow$ audio
piano roll $\rightarrow$ audio
Generating instrument sounds using GANs

• Generate spectrograms
  • SpecGAN [1], TiFGAN [2], GANSynth [3]

• Generate waveforms
  • WaveGAN [1]

• There are also approaches that do not use GANs [4, 5, 6, 7]

SpecGAN and WaveGAN

• Generating 1-second audio at 16kHz (16,000 points)
  https://chrisdonahue.com/wavegan/

• Model: based on DCGAN
  • Flatten 2D convolutions into 1D (e.g., 5x5 2D convolution becomes length-25 1D)
  • Increase the stride factor for all convolutions (e.g., stride 2x2 becomes stride 4)
  • DCGAN outputs 64x64 images; add one more layer so that the output has 16,384 points

Donahue et al., “Adversarial audio synthesis,” ICLR 2019
**GANSynth**

- Generating 4-second audio at 16kHz [1]: large output dimension
  
  https://magenta.tensorflow.org/gansynth

- **Model:** based on **PGGAN** [2]
  
  - Progressively grow the GAN, from low to high resolution
  
  - During a resolution transition, interpolate between the output of two resolutions, with weight $\alpha$ linearly increasing from 0 to 1

---


PGGAN: progressive growing of GANS

Karras et al., “Progressive growing of GANs for improved quality, stability, and variation,” ICLR 2018
**GANSynth**

- Generating **4-second** audio at 16kHz
  [https://magenta.tensorflow.org/gansynth](https://magenta.tensorflow.org/gansynth)

- Model: based on **PGGAN (conv2d)**

- Output: **mel-spectrogram** + **instantaneous freq (IF)**
  - Use IF to derive the phase, and then use inverse STFT to get the waveform
  - STFT window size 2048, stride 512 (so, about 128 frames)
  - 1024-bin mel-frequency scale
  - Target output tensor size: **128 x 1024 x 2**
  - (2x16 → 4x32 → ... → 128x1024)

---

Engel et al., “GANSynth: Adversarial neural audio synthesis,” *ICLR* 2019
GANSynth: Why instantaneous frequency?
GANSynth: Pitch-conditioned generation

• Input to the generator
  • 256-dim random vector Z
  • 61-dim one-hot vector (MIDI 24-84) for pitch conditioning

• Auxiliary pitch classification loss for the discriminator (ACGAN [1])
  • In addition to the real/fake loss
  • Try to predict the pitch label

GANSynth: Possible future extensions

• Modify the network to generate variable-length audio

• From note-level synthesis 
  (i.e., condition on a one-hot pitch vector);

to phrase-level synthesis 
  (i.e., condition on a piano roll) [1,2]

  • The IF might be noisy during note transitions
  • May need to deal with the use of playing techniques [3]

PerformanceNet: Possible future extensions

- Building an “AI Performer” [1,2,3]

Outline

Section 1
Overview of music generation research

Section 2
Introduction to GANs

Section 3
Case studies of GAN-based systems

Section 4
Current limitations
Future research directions
**Limitations of GANs**

• A bit difficult to train (it's helpful to know some tips and tricks [1])
• Only learn $z \rightarrow X$ mapping, not the inverse ($X \rightarrow z$)
• Less explainable [2]
• Less clear how GANs can model text-like data or musical scores
• Unclear how GANs (and all other music composition models in general) can generate “new” music genres

---


The many other generative models

- Variational autoencoders (VAEs)
- Flow-based models
- Autoregressive models
- Attention mechanisms (transformers)
- Restricted Boltzmann machines (RBMs)
- Hidden Markov models (HMMs)
- ...and more!
Future research directions

• Better network architectures for musical data, including piano rolls, MIDI events, musical scores, and audio
• Learning to generate music with better structure and diversity
• Better interpretability and human control (e.g., [1])
• Standardized test dataset and evaluation metrics [2]
• Cross-modal generation, e.g., “music + lyrics” or “music + video”
• Interactive music generation (e.g., [3])

Future research directions

• Composition style transfer (musical genre) using the LPD dataset
• More work on performance style transfer
• Phrase-level audio generation instead of note-level synthesis [1]
• Multi-singer audio generation and style transfer
• Lyrics-free singing generation [2]
• EDM generation [3,4]

Future research directions

• The “MIR4generation” pipeline
  • Learning to generate music (that is expressive) from machine-transcribed data (i.e., learning to compose and perform at the same time)

Yeh et al., “Learning to generate Jazz and Pop piano music from audio via MIR techniques,” ISMIR-LBD 2019
Thank you!
Any questions?

Contact
Hao-Wen Dong (salu.hwdong@gmail.com)
Yi-Hsuan Yang (yang@citi.sinica.edu.tw)

Tutorial website
salu133445.github.io/ismir2019tutorial/