Arranging an Audio Track to other Genre using CycleGAN-based Deep Learning

Alex DongHyeon Seo
Hyecheol (Jerry) Jang
Stella Kim

Table of contents

Introduction / Motivation

<AWS Deep Composer>

- AWS Deep composer applies Deep Generative models to help users build their own creative music just with the keyboard notes they provide

<Music Arrangement to Different Genre>

- There are a lot of trials to arrange specific songs to different genre to provide listeners to have new musical experience

Objective / Related Work

- Project Objective

- Change the genre of a music track, given a set of user input containing song and desired genre
- Use CycleGAN–based model that is widely used in image-to-image translation and use MIDI file as an input data

- Related Work (Brunner et al.)

- Motivated from great success of using Deep Generative Models for style transfer for images, this paper approaches to apply generative models to transfer music style
- However, they used MIDI file with complicated data pre-processing work that would generate biased results

Symbolic Music Genre Transfer with CycleGAN

Gino Brunner, Yuyi Wang, Roger Wattenhofer and Sumu Zhao*
Department of Information Technology and Electrical Engineering
ETH Zürich
Switzerland

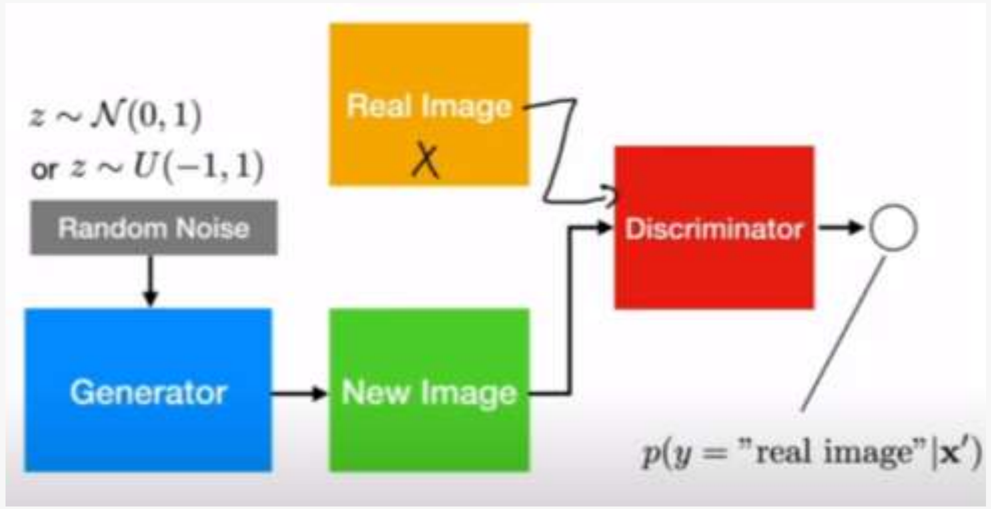
arXiv:1809.07575v1 [cs.SD] 20 Sep 2018

[illegible]

<Referenced CycleGAN-based model paper>

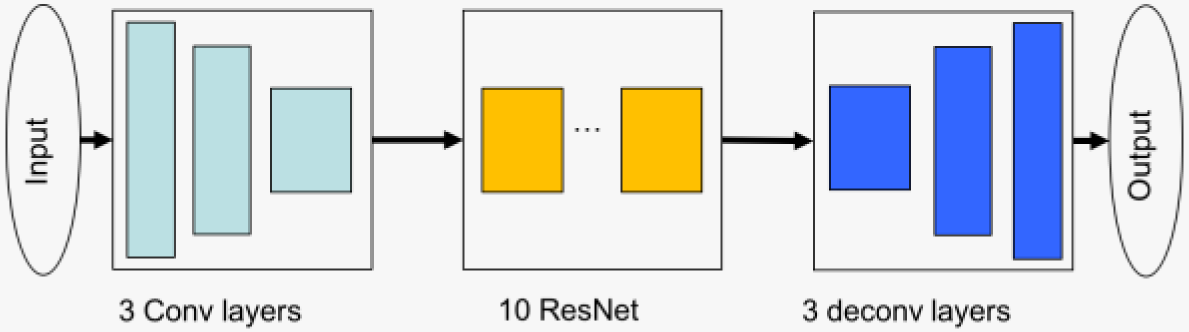
Method

- General GAN model consists of **Generating Model** and **Discriminating Model**

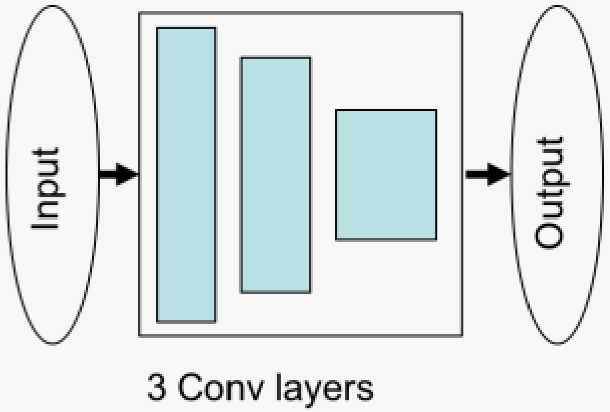


- Generator's goal is to fool discriminator
  by making a good fake output which looks like a real one

- Discriminator's goal is to check whether the given output is fake(generated by model) or real(from label)



<Generator for our model>



<Discriminator for our model>

Method

- CycleGAN

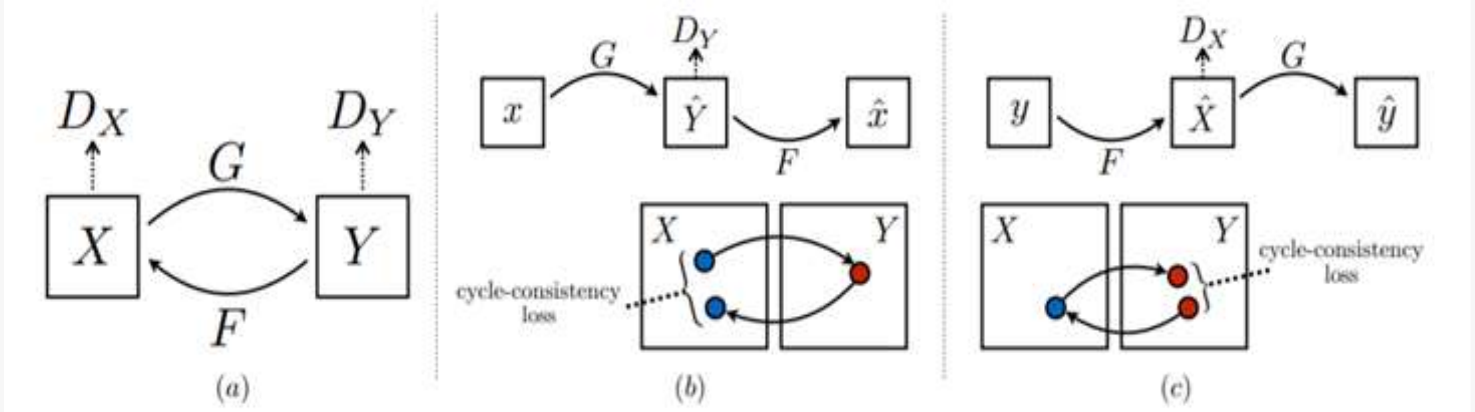


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

- CycleGAN consists of two GANs arranged in cyclic fashion and trains in unison

- Model checks the quality of transfer by using both XtoY and YtoX translator.

- Once we transfer from X to Y,
  we use YtoX translator to restore original x,
  then calculate the difference

Dx, Dy : Discriminator
G, F : Generator

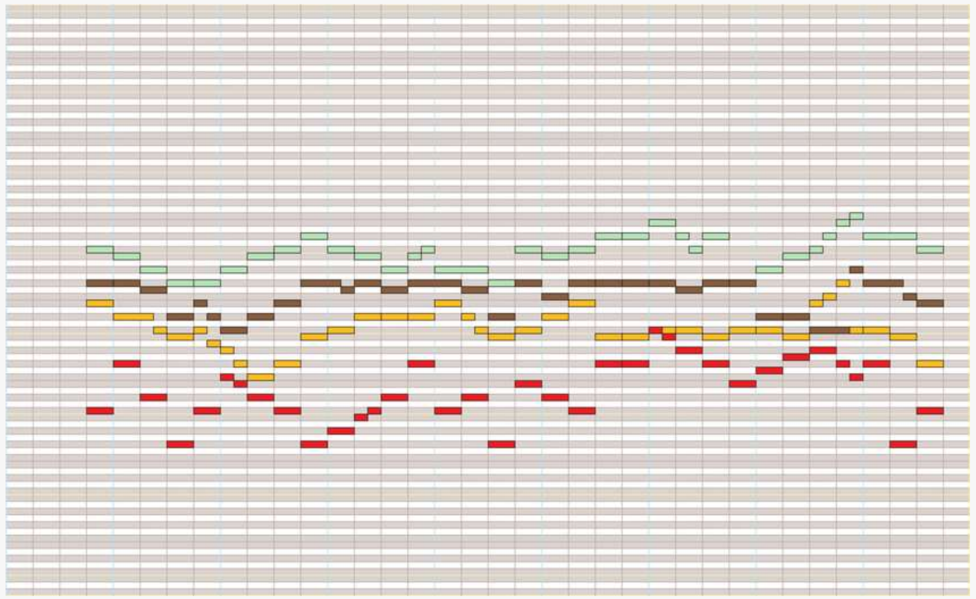
Experiments – Data Selection

<Data Experiment>

- Raw Audio File (MP3, WAV):
  Could not find anyway to put these files to the Machine Learning models

- Convert Audio File to MIDI:
  Audio files have independent audio from multiple instruments and computer could not isolate each instruments

- Raw MIDI file:
  MIDI is a standardized way to represent the musical instruments via computer

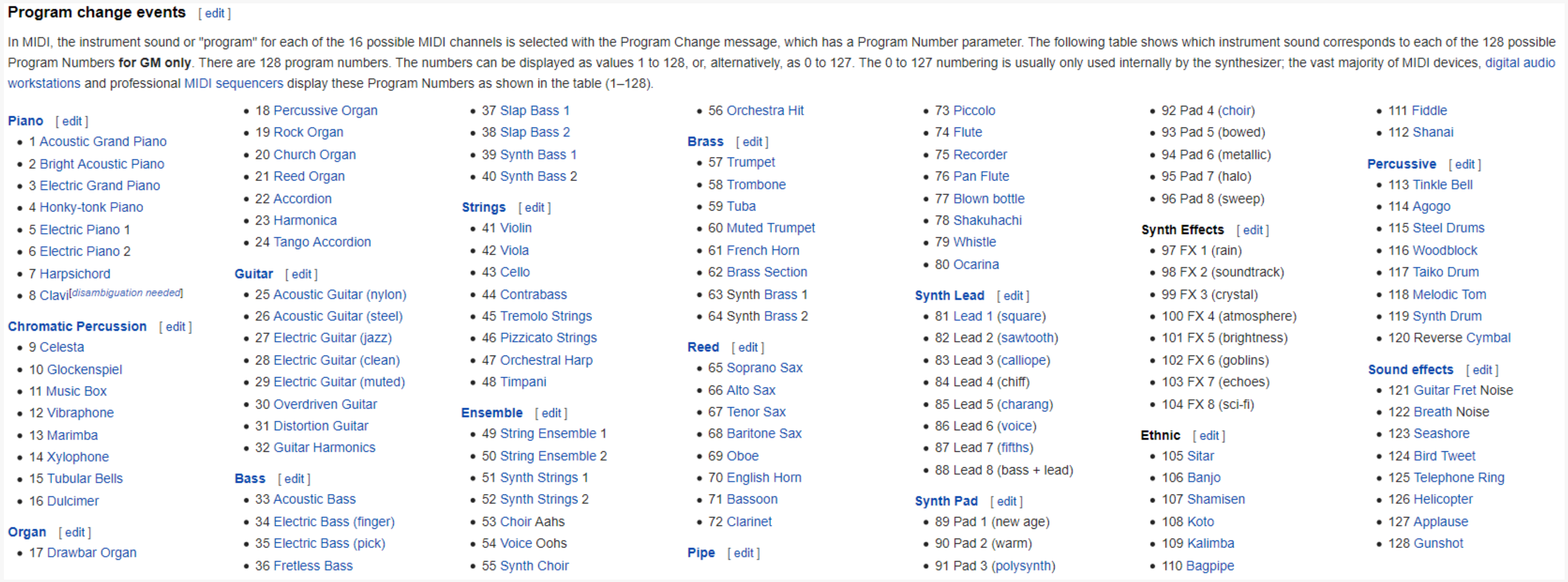


<Example of MIDI file>
MIDI: Electrical version of musical sheet

Decided to use **MIDI file** for our dataset

Experiments – Data Selection

- Advantage of using MIDI file is that MIDI specifies the different instruments independently
- There are 128 representations of different instruments



Program change events [edit]

In MIDI, the instrument sound or "program" for each of the 16 possible MIDI channels is selected with the Program Change message, which has a Program Number parameter. The following table shows which instrument sound corresponds to each of the 128 possible Program Numbers **for GM only**. There are 128 program numbers. The numbers can be displayed as values 1 to 128, or, alternatively, as 0 to 127. The 0 to 127 numbering is usually only used internally by the synthesizer; the vast majority of MIDI devices, digital audio workstations and professional MIDI sequencers display these Program Numbers as shown in the table (1–128).

Piano [edit]
- 1 Acoustic Grand Piano
- 2 Bright Acoustic Piano
- 3 Electric Grand Piano
- 4 Honky-tonk Piano
- 5 Electric Piano 1
- 6 Electric Piano 2
- 7 Harpsichord
- 8 Clavi[disambiguation needed]

Chromatic Percussion [edit]
- 9 Celesta
- 10 Glockenspiel
- 11 Music Box
- 12 Vibraphone
- 13 Marimba
- 14 Xylophone
- 15 Tubular Bells
- 16 Dulcimer

Organ [edit]
- 17 Drawbar Organ
- 18 Percussive Organ
- 19 Rock Organ
- 20 Church Organ
- 21 Reed Organ
- 22 Accordion
- 23 Harmonica
- 24 Tango Accordion

Guitar [edit]
- 25 Acoustic Guitar (nylon)
- 26 Acoustic Guitar (steel)
- 27 Electric Guitar (jazz)
- 28 Electric Guitar (clean)
- 29 Electric Guitar (muted)
- 30 Overdriven Guitar
- 31 Distortion Guitar
- 32 Guitar Harmonics

Bass [edit]
- 33 Acoustic Bass
- 34 Electric Bass (finger)
- 35 Electric Bass (pick)
- 36 Fretless Bass
- 37 Slap Bass 1
- 38 Slap Bass 2
- 39 Synth Bass 1
- 40 Synth Bass 2

Strings [edit]
- 41 Violin
- 42 Viola
- 43 Cello
- 44 Contrabass
- 45 Tremolo Strings
- 46 Pizzicato Strings
- 47 Orchestral Harp
- 48 Timpani

Ensemble [edit]
- 49 String Ensemble 1
- 50 String Ensemble 2
- 51 Synth Strings 1
- 52 Synth Strings 2
- 53 Choir Aahs
- 54 Voice Oohs
- 55 Synth Choir
- 56 Orchestra Hit

Brass [edit]
- 57 Trumpet
- 58 Trombone
- 59 Tuba
- 60 Muted Trumpet
- 61 French Horn
- 62 Brass Section
- 63 Synth Brass 1
- 64 Synth Brass 2

Reed [edit]
- 65 Soprano Sax
- 66 Alto Sax
- 67 Tenor Sax
- 68 Baritone Sax
- 69 Oboe
- 70 English Horn
- 71 Bassoon
- 72 Clarinet

Pipe [edit]
- 73 Piccolo
- 74 Flute
- 75 Recorder
- 76 Pan Flute
- 77 Blown bottle
- 78 Shakuhachi
- 79 Whistle
- 80 Ocarina

Synth Lead [edit]
- 81 Lead 1 (square)
- 82 Lead 2 (sawtooth)
- 83 Lead 3 (calliope)
- 84 Lead 4 (chiff)
- 85 Lead 5 (charang)
- 86 Lead 6 (voice)
- 87 Lead 7 (fifths)
- 88 Lead 8 (bass + lead)

Synth Pad [edit]
- 89 Pad 1 (new age)
- 90 Pad 2 (warm)
- 91 Pad 3 (polysynth)
- 92 Pad 4 (choir)
- 93 Pad 5 (bowed)
- 94 Pad 6 (metallic)
- 95 Pad 7 (halo)
- 96 Pad 8 (sweep)

Synth Effects [edit]
- 97 FX 1 (rain)
- 98 FX 2 (soundtrack)
- 99 FX 3 (crystal)
- 100 FX 4 (atmosphere)
- 101 FX 5 (brightness)
- 102 FX 6 (goblins)
- 103 FX 7 (echoes)
- 104 FX 8 (sci-fi)

Ethnic [edit]
- 105 Sitar
- 106 Banjo
- 107 Shamisen
- 108 Koto
- 109 Kalimba
- 110 Bagpipe
- 111 Fiddle
- 112 Shanai

Percussive [edit]
- 113 Tinkle Bell
- 114 Agogo
- 115 Steel Drums
- 116 Woodblock
- 117 Taiko Drum
- 118 Melodic Tom
- 119 Synth Drum
- 120 Reverse Cymbal

Sound effects [edit]
- 121 Guitar Fret Noise
- 122 Breath Noise
- 123 Seashore
- 124 Bird Tweet
- 125 Telephone Ring
- 126 Helicopter
- 127 Applause
- 128 Gunshot

Experiments – Code Analysis

<Benchmarking Study>

- Brunner et al. paper's code were reviewed – Uses TensorFlow 1.4
  https://github.com/sumuzhao/CycleGAN-Music-Style-Transfer

Problems

- Uses outdated version of TF
- Some functions that were used in this code were removed from TF
- Most of the codes contained hard-coded directory connection
- Unnecessary data preprocessing produced biased result

→

Solutions

- Fixed to use latest version of TF 1.x
- Fixed the code to iterate all defined dataset at once while pre-processing
- Fixed to use command line argument while training and testing the model

Experiments – Generalized Model

- The Brunner et al.’s model filtered MIDI file such that
  1. Having time signature change
  2. First beat does not start at time 0
  3. Time signature is not 4/4

- The music that does not filter is very specific, which does not represent the songs in the genre

- Our goal for here is to remove those filters and make generalized model based on the Brunner et al.’s work.

Results – Generalized Model

<Original Pop Song> <Transferred to Classical>

<Original Classical> <Transferred to Pop Song>

- Original Song does not contain all sounds as the code only uses one track (the piano), and drop the others (Only accept 2D array[tone, time] for training)

- Input data does not represent all the sounds of a song
- Causing the model unable to learn the characteristic of each genre

Experiments – Merged to one Channel

- Seeing the Results and the analysis for generalized model trial,
  We at least **need to merge tracks to solve dimension issue** or
  need to fix code to utilize 3D tensor for input
- There exists blank slices
  Remove trailing and ending whitespace from the MIDI file before slicing

- Model 1: epoch = 100, Learning Rate = 0.0002, ngf = 64
- Model 2: epoch = 200, Learning Rate = 0.0001, ngf = 64
- ~~Model 3: epoch = 100, Learning Rate = 0.0002, ngf = 128~~
- ~~Model 4: epoch = 200, Learning Rate = 0.0001, ngf = 128~~

- Note that learning rate linearly diminish after 50% of epoch

Result – Merged to one Channel

<Original Pop Song>

<Model 1> <Model 2>

<Transferred to Classical>

<Original Classical>

<Model 1> <Model 2>

<Transferred to Pop>

- At least it sounds like a song
- It also contains the melody from the original track
- But hard to tell if the transferred song belongs to classical or pop

- Need further verification on the reason why
  1) Maybe because lots of pop songs are already motivated from classical songs
  2) By removing the instrument information, it ruins the characteristics of each genre

- For further studies,
  Try with other genres: Jazz and Rock

Discussion / Conclusion

- There is no perfect way to pass full audio information to deep learning model.
  Using MIDI file is not generalized approach to represent music.
  (We do not commonly use MIDI files to listen to the music)

- While the ResNet and U-Net are reckoned to be the state-of-the-arts convolution filters for vision tasks, there's no "good" convolution filters for audio tasks to the best of our knowledge

- Using Spectrogram is another way to deal with audio tasks, but it can only be used for classification as there is no way to map spectrogram back to the audio.

- Further Topics
  Modify the model structure to get 3D tensor as input to represent all the instruments.
  Try with other genres

References

1) Brunner, G., Wang, Y., Wattenhofer, R., & Zhao, S. (2018, September 20). Symbolic Music Genre Transfer with CycleGAN. Retrieved from https://arxiv.org/abs/1809.07575
2) Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2018, November 15). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. Retrieved from https://arxiv.org/abs/1703.10593
3) https://www.midi.org/specifications-old/item/gm-level-1-sound-set

Thank You!