ECS7022P: Computational Creativity Project

Project Title: Generating Music with GANs
Student Name: Chao Ma
Student ID Number: 220133414

HTML Link to Colab Notebook [required]: ECS7022P_Project_Music Generation with GANs

Back up link on colab

HTML Link to System Outputs: Generative Model Outputs

Project Overview [10%]

Generating realistic and aesthetic pieces has been considered one of the most exciting tasks in AI. Implementing a generative model called generative adversarial networks (GANs) have made prominent progress in generating images, videos and text as the revolution of deep neural networks. Generating music or music-like sounds from a model or algorithm is referred to as Music Generation. Similarly, attempts of music generation with GANs have also been made. However, there are still challenging remains in this area.

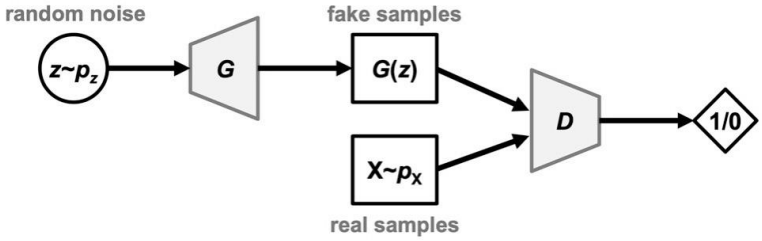
Essentially, the generative AI system aims to create multi-track music that has more than one instrument playing at a time using by building a neural music generator based on state-of-the-art GANs. The models can generate music either entirely on their own, or while accompanying a track that the user has already provided. Its potential for music generation has not been fully realized.

The core concept of GAN is to achieve adversarial learning by constructing two networks: the generator 'G' and the discriminator 'D'. The goal of the generator is to convert a random noise sampled from a prior distribution into a realistic artificial data, and the discriminator needs to identify real data from those generated by the generator. While training, both the generator and the discriminator will be updated simultaneously. To do this, we use training data from the Lakh Pianoroll Dataset to teach our model how to generate musical phrases that are found in pop songs. These phrases include tracks for bass, drums, guitar, piano, and strings.

Artificial Intelligence and writing music have always been my passion and I have explored these area for several years. In addition, music generation using deep learning techniques has been a topic of interest for the past two decades. Music proves to be a different challenge compared to images, which also motivates me to undertake this project.

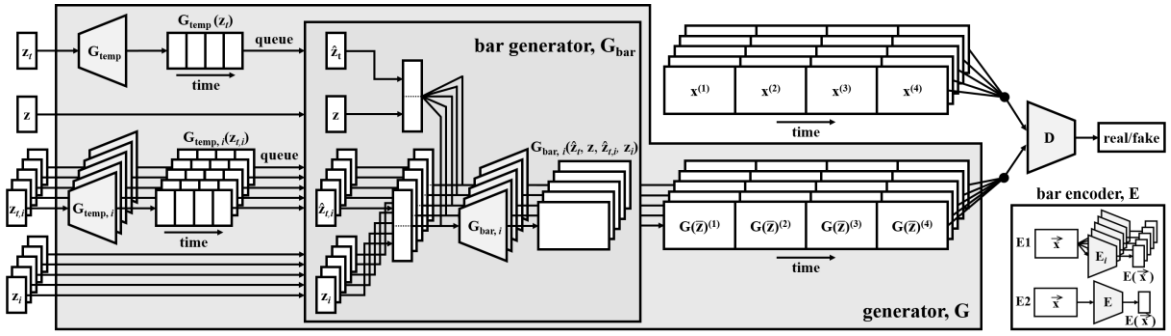
Generative Models [10%]

The type of the architecture is based on GANs, which is shown as:



The generator makes G(z) indistinguishable from real data for D, the discriminator tells G(z) as fake data from X being real ones.

In this project, models for symbolic multi-track music generation under the framework of generative adversarial networks (GANs) are proposed. The models are trained with Lakh Pianoroll Dataset (LPD), a new multi-track piano-roll dataset, in an unsupervised approach. The system of the proposed model for multi-track sequential data generation is shown as:

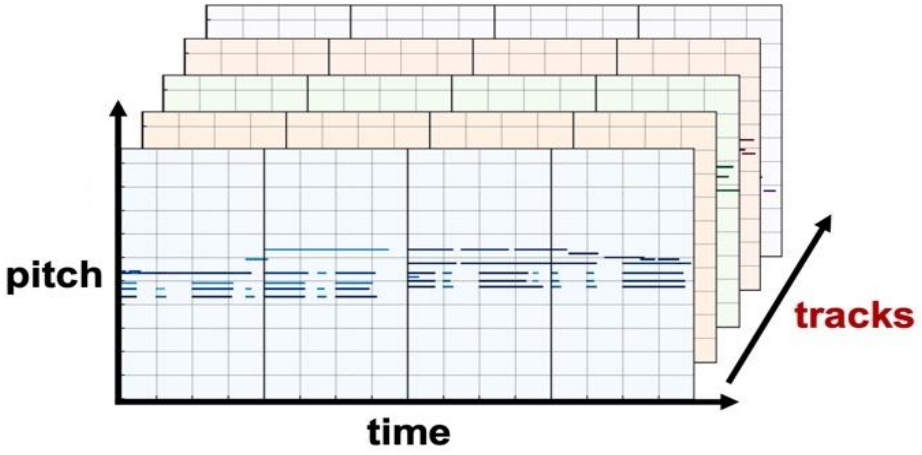


Trained the proposed models on a dataset of over one hundred thousand bars of rock music and applied them to generate piano-rolls of five tracks: bass, drums, guitar, piano and strings. The discriminator is trained to distinguish real data from those generated by the generator, whereas the generator is trained to fool the discriminator. The training procedure can be formally modelled as a two-player minimax game between the generator G and the discriminator D. In order to achieve such conditional generation with high-dimensional conditions, an additional encoder E is trained to map y to the space of z. Firstly, train the composer model for generation from scratch to gain insights. Secondly, the generated piano-rolls evolve as the training process unfolds. Finally, acquire musically meaningful phrases to train the temporal model.

The results show that G indeed becomes better as the training process proceeds. The input to this system, denoted as z, is composed of four parts: an inter-track time-independent random vectors z, an intra-track time-independent random vectors z_i, an inter-track time-dependent random vectors z_t and an intra-track time-dependent random vectors $z_{i,t}$. The output series (of latent vectors), together with the time-independent random vectors, z and z_i, are concatenated and fed to the bar generator G_{bar}, which then generates piano-rolls sequentially.

While some existing work used lead sheets or music in ABC format as the training data, this project aims at learning directly from MIDI files.

The training data is collected from Lakh Pianoroll Dataset (LPD), a new multitrack pianoroll dataset which is a collection of 174,154 multitrack pianorolls derived from the Lakh MIDI Dataset (LMD).The multitrack pianorolls in LPD are stored in a special format for efficient I/O and to save space. In this project, the data is loaded with Pypianoroll which is an open-source Python library for working with piano rolls. It provides essential tools for handling multitrack piano rolls, including efficient I/O as well as manipulation, visualization and evaluation tools [3]. During the data cleansing and pre-processing procedure, those MIDI files are converted into piano-roll with symbolic timing, which can be represented as flows:



By default, samples will be generated alongside the training. It is possible to disable this behaviour by setting the save_samples_steps to zero. The generated pianorolls will be stored in .npz format to save space and processing time.

Process [15%]

The project is an integration and extension of the proposed multitrack and temporal models, takes as input four different types of random vectors:

- an inter-track time-independent random vector (z)
- an inter-track time-dependent random vector (zt)
- M intra-track time-independent random vector (zi)
- M intra-track time-dependent random vectors (zi, t)

Prepare training data, the training data is collected from Lakh Pianoroll Dataset(LPD), a new multitrack pianoroll dataset.

Train a new model, set up a new experiment with default settings and modify the configuration for experimental setting.

It is possible to use pretrained models by downloading the pretrained models manually: *Pretrained_models.tar.gz*

For track i (i =1… M),the shared temporal structure generator Gtemp and the private temporal structure generator Gtemp, I take the time-dependent random vectors, zt and zi, t, respectively, as their inputs, and each of them outputs a series of latent vectors containing inter-track and intra-track, respectively, temporal information.

The output series (of latent vectors), together with the time-independent random vectors, z and zi, are concatenated and fed to the bar generator Gbar, which then generates pianorolls sequentially.

The model can generate music either from scratch, or accompanying a track given by user. By choosing the instrument from GUI, or uploading a given track, users can interact with the system.

The matched subset of the Lakh MIDI dataset [4], after cleansing
• Pop/rock, 4/4-time signature, C key
• Five tracks: bass, drums, guitar, piano, strings (others)
• Get 4-bar phrases by structural feature-based segmentation

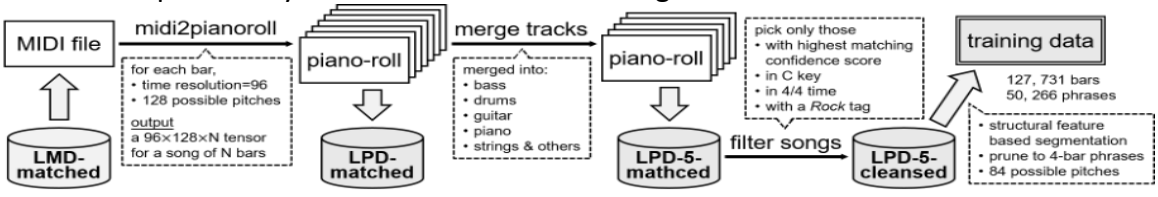
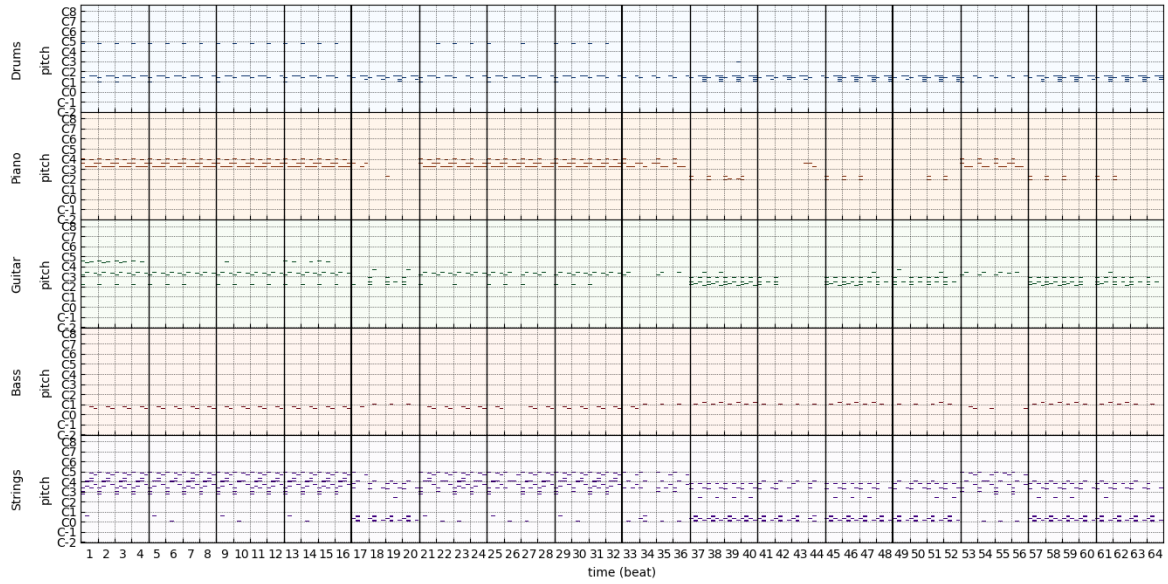


Figure 2. Flowchart of the data cleansing and pre-processing procedure

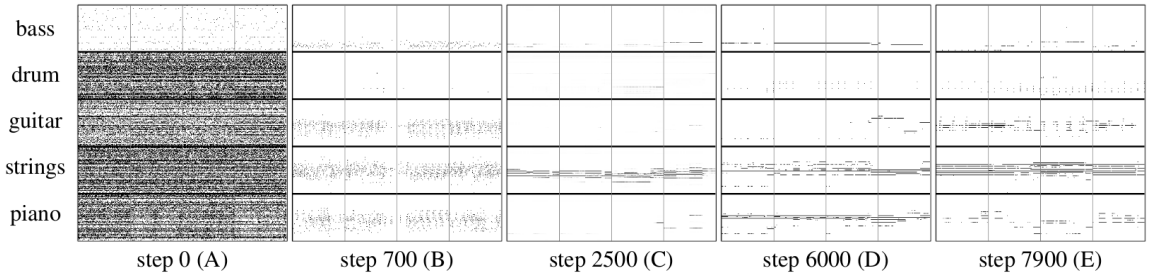
Example Outputs [10%]

Pianoroll Visualization:



By default, samples will be generated alongside the training. Users can disable this behaviour by setting 'save_samples_steps' to zero in the configuration file (config.yaml). The generated will be stored in the following three formats by default.

The evolution of the generated pianorolls as a function of update steps:



- npy: raw numpy arrays
- png: image files
- npz: multitrack pianoroll files that can be loaded by the Pypianoroll package

The generated pianorolls are stored in .npz format to save space and processing time. Use the following code to write them into MIDI files:

```
from pypianoroll import Multitrack
m = Multitrack('./test.npz')
m.write('./test.mid')
```

Some sample results can be found in ./exp/ directory. More samples can be downloaded from the following links.

- sample_results.tar.gz (54.7 MB): sample inference and interpolation results
- training_samples.tar.gz (18.7 MB): sample generated results at different steps

Evaluation [20%]

Based on the outputs of the generated system, the proposed model can gain insights into the learning process and has a few desirable properties.

Based on the creative autonomy of the generated system, this model is a novel generative model for multi-track sequence generation under the framework of GANs.

Based on the higher-level issue in computational creativity, emphasis in on developing and deploying AI systems that people would agree to exhibit behaviours associated with creativity.

Based on deep learning metrics, to evaluate the models, design several metrics that can be calculated from both the training(real) data and the generated data. By comparing the values computed from the real and generated data, the performance of the generator can be evaluated, there are four intra-track and one inter-track metrics [4]:

- EB: ratio of empty bars (in %).
- UPC: number of used pitch classes per bar (from 1 to 12).
- QN: ratio of "qualified" notes (in %).
- NIS: ratio of notes in the C scale (in %).
- TD: or tonal distance.

Based on the ethical stance on generative AI, in terms of the training data, much of the data scraped for the training to generative models was not made available for this purpose. For some artists, it's fine for companies to use this data, but others, this is a problem. Especially as they are fearful of disruption in their industry. In terms of the job satisfaction and disruption, for many artists, their career, well-being and self-worth are influenced by their creative practice, for example, they feel being uniquely creative in some way. All of this is disrupted with the introduction of new tools, especially generative tools, if they automated some elements of this practise. Just because we can automate a task, however, it doesn't mean we should. In terms of the output diversity and fakes, generative models compress a vast amount information by learning stereotypes over images, txt and audio. Using their output can lead to further emphasis of harmful stereotypes. What's more, deepfakes are much easier to produce now, it seems the severe issues in authenticity coming soon. In addition, there are other ways of using generative AI technologies to cause offence. In terms of the environmental concerns, training models takes a lot of energy and money and huge number of predictions on pre-trained models. Besides, training and fine-tuning on-laptop and on-device may become commonplace. This generative AI systems takes large datasets and some time during and after training.

Value Added [20%]

In this project, several neural models are trained rather than using pre-trained ones. For example, the generator and discriminator models.

This is a project on music generation. In a nutshell, it aims to generate polyphonic music of multiple tracks (instruments). The proposed models are able to generate music either from scratch, or by accompanying a track given a priori by the user. This model consists of two parts: a multitrack model and a temporal model. The multitrack model is responsible for the multitrack interdependency, while the temporal model handles the temporal dependency [4]. Proposed three multitrack models according to three common composition approaches. For the temporal model, proposed one for generation from scratch and the other for accompanying a track given a priori by the user.

Casual Creator is an interactive system that encourages the fast, confident, and pleasurable exploration of a possibility place, resulting in the creation or discovery of surprising new artifacts that bring feelings of pride, ownership and creativity to the users that make them. In this project, a casual creator is created

In terms of the higher-level philosophical issues, emphasis in on developing and deploying AI systems that people would agree to exhibit behaviours associated with creativity. Machines usually take the creative lead and we evaluate in terms of public perception of creativity. Ultimately, it is thrilling to be in the presence of such an interesting creator as the system it's completely independent of users, and it teaches users new things and inspires them. For the GAN model to be taken seriously as an artist, it needs to join the debate about what creativity means, as an essentially contested concept and societal driving force. Based on its own creative endeavours and enable it to critique the thoughts of others, extend the models to human-AI cooperative music generation: given a specific track composed by human, we can generate four additional tracks to accompany it. In addition, add dialogue systems to propose, prove and disprove hypotheses about the nature of creativity and generally provoke discussion and though around the topic.

Colab Notebook Code [15%]

Regarding the evaluation:

```
def eval(self, batch, output_type=0, quiet=False, save_fig=False, fig_dir='./'):
    """
    Evaluate one batch of bars according to eval_map and eval_pair
    Args:
        batch (tensor): The input tensor.
        output_type (int): 0 for scalar (mean of list), 1 for list
        quiet (bool): if true, print the values
        save_fig (bool): if true, plot figures and save them under 'fig_dir'
        fig_dir (str): dir to store images
    Returns:
        score_matrix: result of eval map
        score_pair_matrix: result of eval pair
    """
    batch =  np.reshape(batch,(-1, 96, 84, 5))
    num_batch = len(batch)
    score_matrix = np.zeros((self.metrics_num, self.track_num, num_batch)) * np.nan
    score_pair_matrix = np.zeros((self.pair_num, num_batch)) * np.nan

    for idx in range(num_batch):
        bar = batch[idx]

        # compute eval map
        for t in range(self.track_num):
            if(self.eval_map[0, t]):
                bar_act = self.metric_is_empty_bar(batch[idx, :, :, t])
                score_matrix[0, t, idx] = bar_act

            if(self.eval_map[1, t] and not bar_act):
                score_matrix[1, t, idx] = self.metric_num_pitch_used(batch[idx, :, :, t])

            if(self.eval_map[2, t] and not bar_act):
                score_matrix[2, t, idx]= self.metric_qualified_note_ratio(batch[idx, :, :, t])

            if(self.eval_map[3, t] and not bar_act):
                score_matrix[3, t, idx]= self.metric_polyphonic_ratio(batch[idx, :, :, t])

            if(self.eval_map[4, t] and not bar_act):
                score_matrix[4, t, idx]= self.metric_in_scale(self.to_chroma(batch[idx, :, :, t]))

            if(self.eval_map[5, t] and not bar_act):
                score_matrix[5, t, idx]= self.metric_drum_pattern(batch[idx, :, :, t])

            if(self.eval_map[6, t] and not bar_act):
                score_matrix[6, t, idx]= self.metric_num_pitch_used(self.to_chroma(batch[idx, :, :, t]))

        # compute eval pair
        for p in range(self.pair_num):
            pair = self.inter_pair[p]
            score_pair_matrix[p, idx] = self.metrics_harmonicity(self.to_chroma(batch[idx, :, :, pair[0]]),
                                                                 self.to_chroma(batch[idx, :, :, pair[1]]))
    score_matrix_mean = np.nanmean(score_matrix, axis=2)
    score_pair_matrix_mean = np.nanmean(score_pair_matrix, axis=1)

    if not quiet:
        print('# Data Size:', batch.shape, '       # num of Metrics:', np.sum(self.eval_map))
        self.print_metrics_mat(score_matrix_mean)
        self.print_metrics_pair(score_pair_matrix_mean)

    # save figures and save info as npy files
    if save_fig:
```

Regarding the combination of systems:

```
[ ]
# @title Helper functions which enable us to generate GUI widgets.
import ipywidgets

def get_button(desc, on_click):
  button = ipywidgets.widgets.Button(description=desc)
  button.on_click(on_click)
  return button

def get_slider(desc, min, val, max):
  return ipywidgets.widgets.IntSlider(value=val, min=min, max=max, step=1,
    description=desc, disabled=False, continuous_update=False,
    orientation='horizontal', readout=True, readout_format='d')

def get_text_box(desc, value="", default="Blank"):
  return ipywidgets.widgets.Text(value=value, placeholder=default, description=desc, disabled=False)

def get_dropdown_list(desc, options, val=None):
  v = options[0] if val == None else val
  return ipywidgets.widgets.Dropdown(options=options, value=v, description=desc, disabled=False)

def get_label(value):
  return ipywidgets.widgets.Label(value=value)

def get_check_box(description, value):
  return ipywidgets.widgets.Checkbox(value=value, description=description, disabled=False, indent=False)
```

Reference:

1. Generating Music Using Deep Learning | by Isaac Tham | Towards Data Science. (n.d.). Retrieved March 15, 2023, from https://towardsdatascience.com/generating-music-using-deep-learning-cb5843a9d55e

2. Lakh Pianoroll Dataset | A collection of 174,154 multitrack pianorolls. (n.d.). Retrieved April 16, 2023, from https://salu133445.github.io/lakh-pianoroll-dataset/

3. Pypianoroll — Pypianoroll documentation. (n.d.). Retrieved April 19, 2023, from https://salu133445.github.io/pypianoroll/

4. Hao-Wen Dong,* Wen-Yi Hsiao,* Li-Chia Yang and Yi-Hsuan Yang, "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment," *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018. (*equal contribution)