A PROJECT REPORT

"Automatic Music Generation using Deep Learning"

Submitted in partial fulfillment of the requirements for the award of the degree of

Masters of Science in Big Data and Business Analytics

Submitted by

Pukhraj Suthar (IGSC19004)

Taral Sarvagod (IGSC19005)

INDIRA GLOBAL STUDY CENTER

INDIRA GLOBAL STUDY CENTER

Certificate

This is to certify that the first year project entitled on "Automatic Music Generation using Deep Learning" is submitted by IGSC students for the partial fulfillment of the requirements of the degree of masters from INDIRA GLOBAL STUDY CENTER ,PUNE is a bonafide work carried out during the academic year 2019-2020.

Dr.PunamBhoyar

(Director)IGSC

ACKNOWLEDGEMENT

We are pleased to present this Project report entitled Automatic Music Generation using Deep Learning. It is indeed a great pleasure and a moment of immense satisfaction for us to express our sense of profound gratitude and indebtedness towards our guide Mr.Sujit Deokar and Ms.Sejal Pitale whose enthusiasm are the source of inspiration for us. We are extremely thankful for the guidance and untiring attention, which they bestowed on us right from the beginning.

Their valuable and timely suggestions at crucial stages and above all the constant encouragement have made it possible for us to achieve this work. We would also like to give our sincere thanks to Dr.Punam Bhoyar Director- IGSC for necessary help and providing us the required facilities for completion of this project report.

We would like to thank the entire Teaching staffs who are directly or indirectly involved in the various data collection and software assistance to bring forward this project report. We express our deep sense of gratitude towards our parents for their sustained cooperation and wishes, which have been a prime source of inspiration to take this project work to its end without any hurdles. Last but not the least, we would like to thank all our IGSC staff for their co-operation and useful suggestion and all those who have directly or indirectly helped us in completion of this project work.

ABSTRACT

Nowadays, audio generation plays an important role in human-computer interactive applications. However, the audio generated by machine is far from nature sound, especially in expressiveness and complexity. Traditionally music was treated as an analogue signal and was generated manually. In recent years, music is conspicuous to technology which can generate a suite of music automatically without any human intervention. To accomplish this task, we need to overcome some technical challenges which are discussed descriptively in this paper. A brief introduction about music and its components is provided in the paper along with the citation and analysis of related work accomplished by different authors in this domain. Main objective of this paper is to propose an algorithm which can be used to generate musical notes using Recurrent Neural Networks (RNN), principally Long Short-Term Memory (LSTM) networks. A model is designed to execute this algorithm where data is represented with the help of musical instrument digital interface (MIDI) file format for easier access and better understanding. Preprocessing of data before feeding it into the model, revealing methods to read, process and prepare MIDI files for input are also discussed. We used an advanced arithmetic for generating music using Generative Adversarial Networks (GAN). The music is divided into tracks and the note segment of tracks is expressed as a piano-roll, through trained a gan model which

generator and discriminator continuous zero-sum game to generate a wonderful music integrality. In most cases, Although GAN excel in image generation, the model adopts a full-channel lateral deep convolution network structure according to the music data characteristics in this paper, generate music more in line with human hearing and aesthetics.

INDEX

INTRODUCTION

Music is an art that reflects the emotions of human life. Music can improve people's aesthetic ability, purify people's minds, and establish lofty ideals. We express our emotions through music and release many of our emotions. Music has been integrated into all aspects of our lives. With the provision of living standards, the demand for music is also increasing. Presently, deep learning has made remarkable progress in speech recognition and speech generation. Because the neural networks can simulate any continuous mapping function theoretically, to generate audio by means of it is a nature idea. Neural networks are being used to improve all aspects of our lives. In recent years, there have been a number of tutorials on how to generate text using neural networks but a lack of tutorials on how to create music.

In this we will go through two approaches which are how to create music using LSTM and recurrent neural network and Generative Adversarial Networks (GAN) in Python using the Keras library. The art of ordering tones or sound in succession, in combination is music. It is a temporal relationship to produce a composition of notes having continuity and unity. In other words, a musical note is any sound generated with the help of musical instruments or human voice. A musical note is a simple unit of music. Music and its notes have certain properties concerning its quality and performance. The sound input for training the Artificial Intelligence (AI) model can be monadic, having a single melodic line, or polyphonic, involving many sounds. The musical notes are exhibited by an octave or some interval of pitches. Pitch has a pitch class that refers to the relative position in an octave. Music details are essential for training the model and the model's complexity and output depends upon the nature of the input.

GAN (generative adversarial network) is an artificial intelligence algorithm for unsupervised learning. It was proposed by Good fellow et al. in 2014. GAN is constituted by two networks, generation network G and discriminator network D, G is responsible for generating target objects. D is responsible for discriminating the object generated by the generator from the real object. With the separate training of the two networks and play zero-sum games with each other in the adaptation training. The generator will generate objects that are very similar to the genuine objects, so that the discriminator network cannot distinguish between the generated objects and the real objects, thereby achieving GAN training. At present, the integration of AI and music has been studied at home and abroad, mainly in deep learning models such as GAN and LSTM. This is because music has the characteristics of typical time series data, which is in line with the advantages of these models.

The model to be trained is supposed to recall the former details and create a rational piece. Music is generated by playing notes of different frequencies and the linkages between the notes are preserved. One method of generating music by utilizing existing music is genetic algorithm. As stated in genetic algorithm can highlight the strong rhythm in each fragment and combine them into distinct pieces of music. But it has low efficiency because every iteration process of it has a delay. In addition, due to the lack of context, it is difficult to get the coherence and deep-seated rhythm information. So, to overcome the above problem we require a system that should be able to remember the previous note sequence and predict the next sequence and so on. Recurrent Neural Networks especially Long Short-term Memory a special RNN is used. This paper construes an algorithm (Neural Network) based on LSTM networks which can be used to generate music and melodies automatically without any human intervention. The key goal is to develop a model which can learn from a set of musical notes, analyze them and then generate a pristine set of musical notes. This task is a real challenge because the model must have capabilities to recall past details and structure of musical notes for future projection of learning sequence. The model needs to learn the original sequences adjacent to past one and transform it for the learning system.

OVREALL OBJECTIVES

Deep Learning is a field of Machine Learning which is inspired by a neural structure. Our team aim is to use neural networks extract the features automatically from the dataset and are capable of learning any non-linear function. The key steps in solving this problem are understanding the data and formulating it and defining the architecture to solve the problem.

Why LSTM –RNN for music generation?

One key challenge with modeling music is selecting the data representation. Possible representations are signal, transformed signal, MIDI, text, etc. A relevant issue is the end destination of the generated music content. The format destination could be a human user, in which case the output would need to be human readable, for instance a musical score. The final output format is therefore readable by a computer, which in this case is a MIDI file (musical instrument digital interface). The MIDI representation was selected because it offers a particularly rich representation in two senses: first it carries characteristics of the music in the metadata of the file, like time steps. Second it is a common digital representation which allowed access to freely and widely available data.

1) Another relevant factor is the level of supervision in the generation of the output. At one extreme is complete autonomy and automation with no human supervision.
2) It could be more interactive, with early stopping built into the model to supervise the music creation process.
3) The MIDI file format optimized for this dimension as well because it offers a complete end product that is machine readable without human intervention.
4) The level of autonomy is an interesting potential development for actual musicians who can interrupt the model in the middle of content generation. or more aesthetically pleasing musical compositions.

Why GAN (generative adversarial network) for music generation?

GAN has been widely researched and apply. Here's a brief list of GAN's advantage and disadvantages.

Advantage

1) GAN is a generative model that uses only back propagation compared to other generation models without the need for complex Markov chains.
2) GAN can produce a clearer, realistic sample closer to the real object.
3) GAN adopts an unsupervised learning style training, which can be widely used in unsupervised learning and semi-supervised learning.
4) Compared to the variation auto encoders GAN does not introduce any deterministic bias, and the variation method introduces deterministic bias because they optimize the lower bound of the log likelihood rather than the likelihood itself.
5) Compared with VAE, GAN has no variation lower bound. In other words, GAN is progressive, but VAE is biased.

Disadvantages

1) Training GAN needs to reach Nash equilibrium sometimes it can be done by gradient descent method or not, We have not found a good way to reach Nash equilibrium, so training GAN is unstable compared to VAE or PixelCNN, but I think in practice it is still more stable than training the Boltzmann machine.
2) GAN is not appropriate for processing discrete forms of data, such as text.
3) GAN has problems of unstable training, gradient disappearance, and mode collapse.

Variant

1) DCGAN: Deep Convolutional Generative Adversarial Networks greatly improves the stability of GAN training and the quality of the resulting results.
2) WGAN: Wasserstein GAN solve problems such as mode collapse, improve learning stability, and provide meaningful learning curves useful for debugging and hyperparametric searches.
3) LSGAN: Least Squares Generative Adversarial Networks Solving the vanishing gradients problem during the learning process by using a least squares loss function for the discriminator.

4) BEGAN: Boundary Equilibrium GAN training auto-encoder based Generative Adversarial Networks which Replace the similarity between the distribution by estimating the similarity between the distribution errors of the distribution to achieve fast and stable training.
5) LSTM-GAN: A conditional LSTM-GAN is optimized to generate discrete-valued sequences of music data by introducing a quantizes.

DATASETS

MIDI (Musical Instrument Digital Interface) is a protocol designed for recording and playing back music on digital synthesizers that is supported by many makes of personal computer sound cards. Originally intended to control one keyboard from another, it was quickly adopted for the personal computer. Rather than representing musical sound directly, it transmits information about how music is produced. The command set includes note-ons, note-offs, key velocity, pitch bend and other methods of controlling a synthesizer. The sound waves produced are those already stored in a wavetable in the receiving instrument or sound card.

Why MIDI file is used?

Here, we selected MIDI file format because firstly, it bears characteristics of a song in its metadata and secondly, it is commonly used, as considerable number of datasets are available. The MIDI format is highly suitable for this, because a MIDI signal is essentially a numeric set of code arranged in time. The module extracted from is used to perform preprocessing and post processing of “Note Matrix”, including the creation of MIDI file for generated notes by the model and importing MIDI files and tracks. The any set of MIDI files consisting of a single instrument would work for our purposes.
We downloaded and combined multiple classical music files of a digital piano from numerous resources.

We use **MAESTRO** (MIDI and Audio Edited for Synchronous TRacks and Organization) from Magenta as the dataset. This dataset only contains piano instruments. We will take 100 musics randomly from around 1000 musics to make our training time faster.

What are the Constituent Elements of Music?

Music is essentially composed of Notes and Chords. Let me explain these terms from the perspective of the piano instrument:

- **Note**: The sound produced by a single key is called a note
- **Chords**: The sound produced by 2 or more keys simultaneously is called a chord. Generally, most chords contain at least 3 key sounds
- **Octave**: A repeated pattern is called an octave. Each octave contains 7 white and 5 black keys

The Lakh MIDI Dataset is distributed with a CC-BY 4.0 license; if you use this data in any capacity, The reference this dataset is here below as.

Colin Raffel. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching". PhD Thesis, 2016.

Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. "The Million Song Dataset". In Proceedings of the 12th International Society for Music Information Retrieval Conference, pages 591–596, 2011.

Clean MIDI subset - A subset of MIDI files with filenames which indicate their artist and title (with some inaccuracy), as used in a few of my papers.

What is a lakh?

A lakh is a unit of measure used in the Indian number system which signifies 100,000 (or, in the Indian convention, 1,00,000). Depending on how you count, the Lakh MIDI Dataset includes about 100,000 MIDI files. The name is a play on the Million Song Dataset, which includes metadata and features for 1,000,000 music recordings.

DEVELOPMENT TOOLS

Python

- **Keras**

  Keras is a high level API built on top of tensorflow such that it simplifies the interaction made with Tensorflow. Keras library is used for creating and training our LSTM model based network. This framework's APIs simplifies the whole code by provide high level APIs to Tensorflow.

- **Music21**

  A. Music21 is a Python based tool used for operating on music. It allows explaining various aspects of music. The toolkit gives musical notes of various journals. Additionally, it nodes can be created. Our use for Music21 includes extracting the dataset which is basically MIDI files then getting the objects of notes and chord to feed into network. Further, our neural network predicted output is converted into musical notations with the help of Music21.

- **Tkinter**

  Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

  Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

  - Import the *Tkinter* module.
  - Create the GUI application main window.
  - Add one or more of the above-mentioned widgets to the GUI application.
  - Enter the main event loop to take action against each event triggered by the user.

METHODOLOGY

We have decided to use two approaches for music generation.

- LSTM (Long Short Term Memory)
- GAN (Generative Adversarial Networks)

At a high level, we feed MIDI files of music, mostly consists of some piano soundtracks. We train our LSTM and GAN networks with the midi files which is used to generate music later on.

Approach 1: Using Long Short Term Memory (LSTM) Model

Long Short Term Memory (LSTM) Model popularly known as LSTM, is a variant of Recurrent Neural Network that is capable of capturing the long term dependencies in the input sequence.
LSTM has a wide range of applications in Sequence-to-Sequence modeling tasks like Speech Recognition, Text Summarization, Video Classification, and so on.
The key goal is to develop a model which can learn from a set of musical notes, analyze them and then generate a pristine set of musical notes.
The preparation of input and output sequences. At each time step, an amplitude value is fed into the Long Short Term Memory cell – it then computes the hidden vector and passes it on to the next time steps.

Input format

To implement the neural network we must first understand the input format to the network.

- We input multiple midi files which splits into two types of objects, Chords and Notes.
- A note object obtained from Music21 contains information about three things of music, offset, pitch values and octave.

- Pitch: The degree of sound which determines its extent (highness or lowness) depends, which is basically the frequency of the sound. It is represented with the letters from A to G. Octave: It is the interval between one musical pitch.

Another thing is the intervals between notes, as the notes can have varying intervals. Notes can occur in any form where there can be swift series for a time period followed by a pause where no note is being played. MIDI files read using Music21 provide the interval between two successive notes. We get to see it's generally 0.5 therefore we can ignore such small offset for our experiment and it will not have much effect on the melodies of the music

B. Preparing the data

We know the format of input which is basically the MIDI files of different music which is read by Music21 to generate objects of notes and chords. This data of chords and notes is then fed into our LSTM[x4] network. Each midi is parsed through the Music21 converter. On parsing MIDI files we get a stream object which consists of all notes and chords. The pitch value is encoded into string notation and appended. We encode id of all notes in a chord separating them by a dot in a single string. Output of the network can be easily decoded into notes and chords due to such encoding.

Since our network much better with integer based value, therefore we convert this categorical data to integer-based values using one hot encoding.

C. Network Architecture

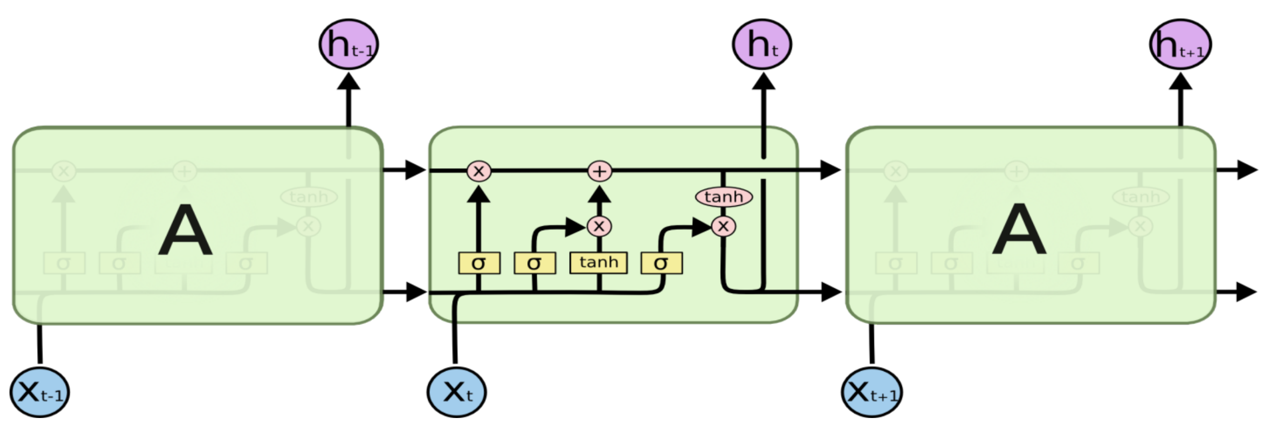


Fig.1. LSTM neural network [image sourece: Deniz Yuret, Jan 6, 2017]

Our model consists of these layers.

- LSTM: A Recurrent Neural Network A traditional human brain learns by persisting things. We do not start learning by scratch every time instead we learning things above what we have already learned. A traditional neural network cannot do this. For example, to understand the events happening at every point in a movie. Recurrent Network are networks with loop in them.

- Dropout layers To reduce overfitting in neural network we are using Dropout technique.

- Dense layers is nothing but a fully connected neural layer connecting each input node to output node.

- The Activation layer.

D. Generating Music

We will reuse the code we used for training the model to generate the music. The only difference is that instead of loading the notes and chords objects we will load the model with weights file. We will setup the network the same way as we did before.

For our music, the beginning point is selected randomly from the list but in any case if one wants to control the starting point then a function can be created to replace the current random function.

You can choose any number of notes in generated music. We chose 1000 notes. Which roughly generates 4 minutes of music We provide the sequence to the network for every note we want to generate. Selecting higher number of notes will require more time to generate the music but will lead to longer music length.

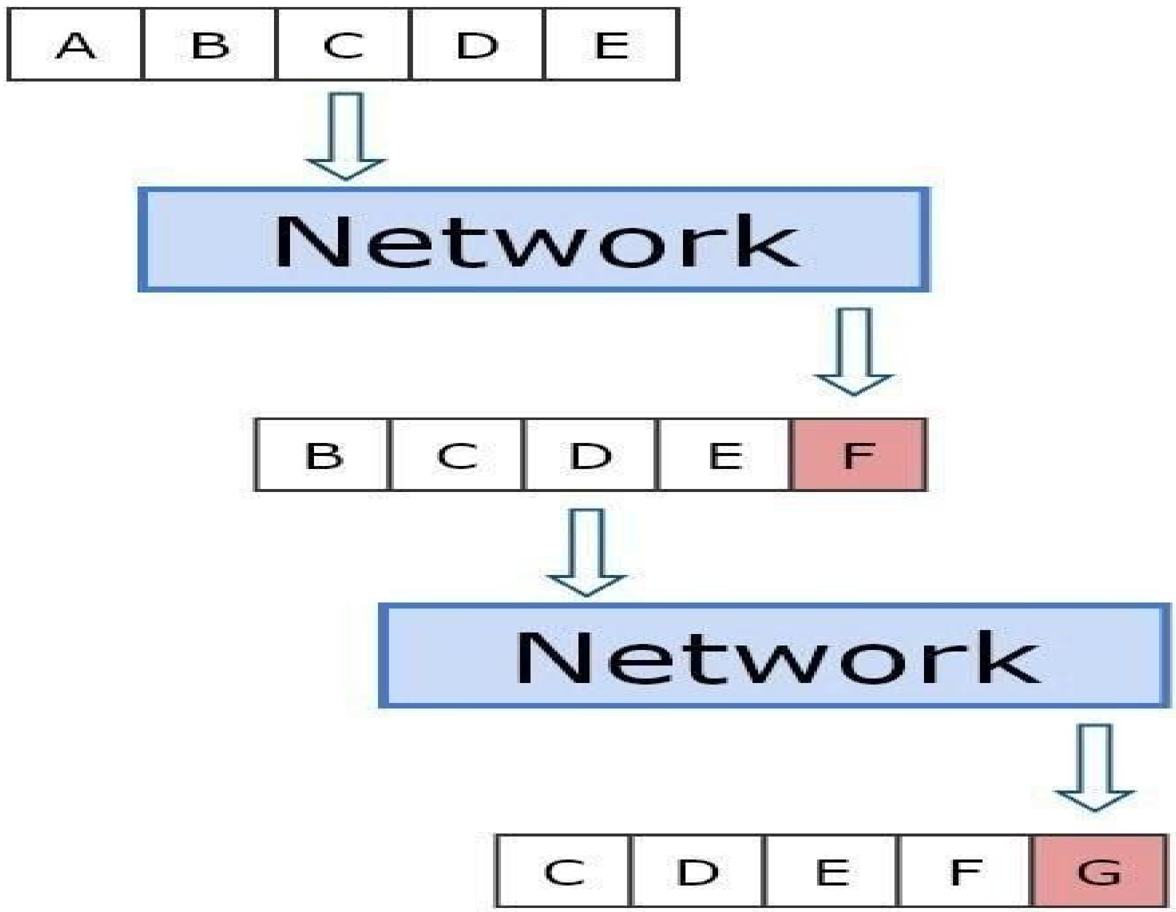


Fig 2. Network selecting notes to generate music. (Image source: Proceedings of the 12th INDIACom; INDIACom-2018; IEEE Conference ID: 42835 2018 5th International Conference on "Computing for Sustainable Global Development", 14th - 16th March, 2018)

A output file is generated by the network which can be played on different applications like VLC music player. There are some weird notes which can be seen in the output sheet. This is because of neural network incapable of making perfect melodies.

Approach 2: Using generative adversarial network (GAN) Model

GAN (generative adversarial network) is an artificial intelligence algorithm for unsupervised learning. It was proposed by Goodfellow et al. in 2014. GAN is constituted by two networks, generation network G and discriminator network D, G is responsible for generating target objects. D is responsible for discriminating the object generated by the generator from the real object. With the separate training of the two networks and play zero-sum games with each other in the adaptation training. The generator will generate objects that are very similar to the genuine objects, so that the discriminator network cannot distinguish between the generated objects and the real objects, thereby achieving GAN training.
We propose a recurrent neural network architecture, C-RNN-GAN (Continuous RNN-GAN), that is trained with adversarial training to model the whole joint probability of a sequence, and to be able to generate sequences of data. Our system is demonstrated by training it on sequences of classical music in midi-format, and evaluated using metrics such as scale consistency and tone range.

Objective function equations:

$$L_G = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(\boldsymbol{z}^{(i)})))$$

$$L_D = \frac{1}{m}\sum_{i=1}^{m}\left[-\log D(\boldsymbol{x}^{(i)}) - (\log(1 - D(G(\boldsymbol{z}^{(i)}))))\right]$$

(where z (i) is a sequence of uniform random vectors in [0, 1]k , and x (i) is a sequence from the training data. k is the dimensionality of the data in the random sequence.) The input to each cell in G is a random vector, concatenated with the output of previous cell. Feeding the output from the previous cell is common practice when training RNNs as language models, and has also been used in music composition. The discriminator consists of a bidirectional recurrent net, allowing it to take context in both directions into account for its decisions. In this work, the recurrent network used is the Long short-term memory (LSTM). It has an internal structure with gates that help with the vanishing gradient problem, and to learn longer dependencies.

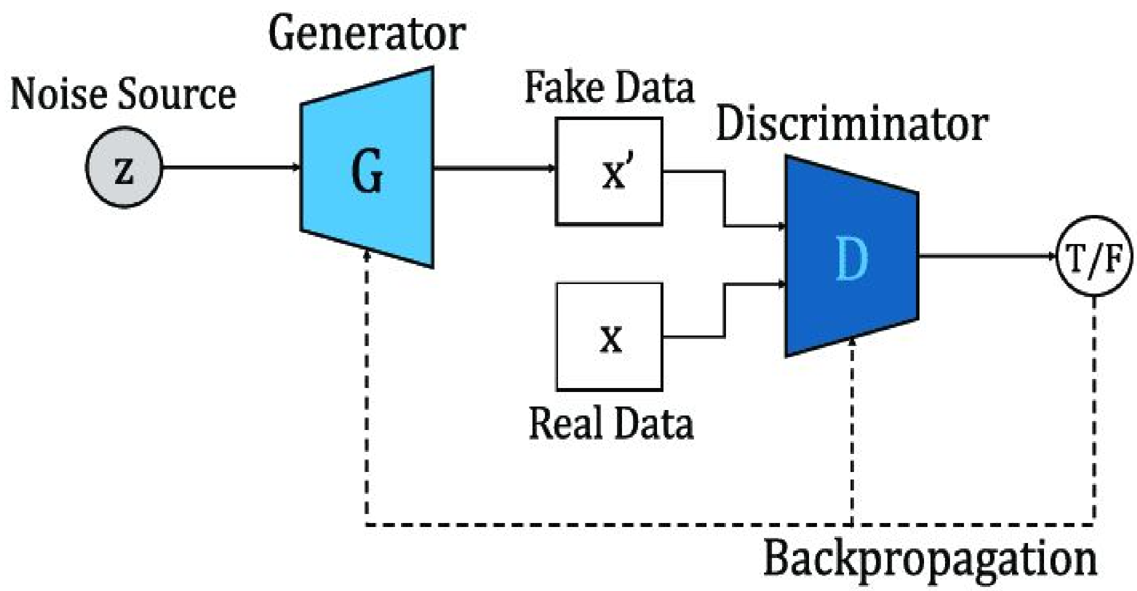


Figure 5. GAN train process.

GAN Model
GAN train process follows as pseudo code:

```
##start
    generator = Generator_build()
    discriminator = Discriminator_build ()
    combiner = discriminator (generator(noise_z))
    for epoch in range(epochs):
    # Train Discriminator # Generate a half batch of new music
    gen_musics=generator.predict(noise)
    # Train the discriminator discriminator.train(gen_musics)
    # Train Generator combiner.train()
##end
```

Data Set

The mentioned above dataset from The Lakh MIDI Dataset1 , which is a collection of 176,581 unique MIDI files, 45,129 of which have been matched and aligned to entries in the Million Song Dataset. These MIDI files are a complete song, including musical instrument combinations of multiple tracks, such as piano, guitar, bass, drum etc. Each track contains some attributes of the instrument, including instrument name and note sequence. The sequence of notes consists of beat, pitch, velocity, and duration.

Data Process

Due to the algorithm design of the paper, we are required to convert the prepared midi file into a piano roller blind and further transform it into the input data format we need to train.

First, considering that we are expected to generate some Pop music. Convert the MIDI file to piano-roll, and spilt it to five tracks: piano, guitar, bass, drum, others. Popular music is mainly composed of the above instruments, and other less common instruments are classified as others.

Second, Consideration of training model, we need to slice and standardization the midi track file. We use the 4 bars as the window unit and cut it longitudinally. If you don't have enough data, you can split the window in smaller steps. And, we separated a 4 bar and 4/4 beats into 192 copies. Since the pitch of most instruments doesn’t cover 0-127, we shorten it to 0-84. Finally, the data we get after standardization such as [N * 5(track) * 192(time step) * 84(pitch)]. In particular, we will clean and filter the dirty and nonstandard data.

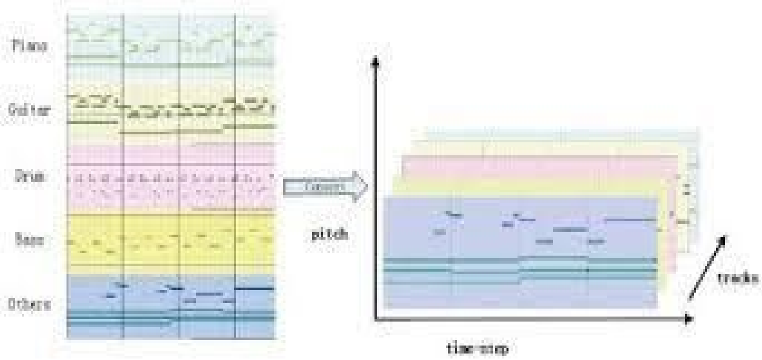


Figure 6. Convert complete piano-roll to train's input data. **Image sources:(INSPEC Accession Number:** 18994176
DOI: 10.1109/ELTECH.2019.8839521)

GAN Train

The model mentioned in the paper contains two networks: generator model G, discriminator model D. This paper uses CNN's network structure for training. The network hierarchy diagram is shown below.

After establishing the G-network and D-network models, we started to train the network GAN and there is some tips.

- When we train the discriminator, hold the generator values constant; and when we train the generator, hold the discriminator constant. Each should train against a static adversary.
- By the same token, pretraining the discriminator against dataset before we start training the generator will establish a clearer gradient.

- Each side of the GAN can overpower the other. If the discriminator model is overtrained, it will return values so close to 0 or 1 that the generator will struggle to read the gradient.
- Otherwise, it will persistently exploit weaknesses in the discriminator that lead to false negatives.
- This may be mitigated by the nets' respective learning rates. GAN model training is a quite time-consuming process, so we can consider using parallel GPU for training under conditions, otherwise, we can only do other things while training.

During the experiment, we found that we can perform as many iterations as possible, but in the training process, we can also try to generate some sample for verification. After all, in the music generation algorithm, it is not necessary to get a colossal model but a model that sounds better.

Evaluate

Music is a subject with a particularly subjective nature. At present, it is not possible to judge the quality of music without hearing. So, we use two sets of programs to evaluate our experimental results.

The first set of evaluation systems is based on statistically objective evaluations. We invited some experts in the music field to evaluate and analyze some high quality music and got some indicators from the midi data level. It is mainly divided into two categories; one is the similarity between the musical instrument orbital note sequence and the chord, because we believe that harmony is an important component of people's sense of hearing. The other is the interval relationship between notes and notes, which are well documented in both psychology and biophysics.

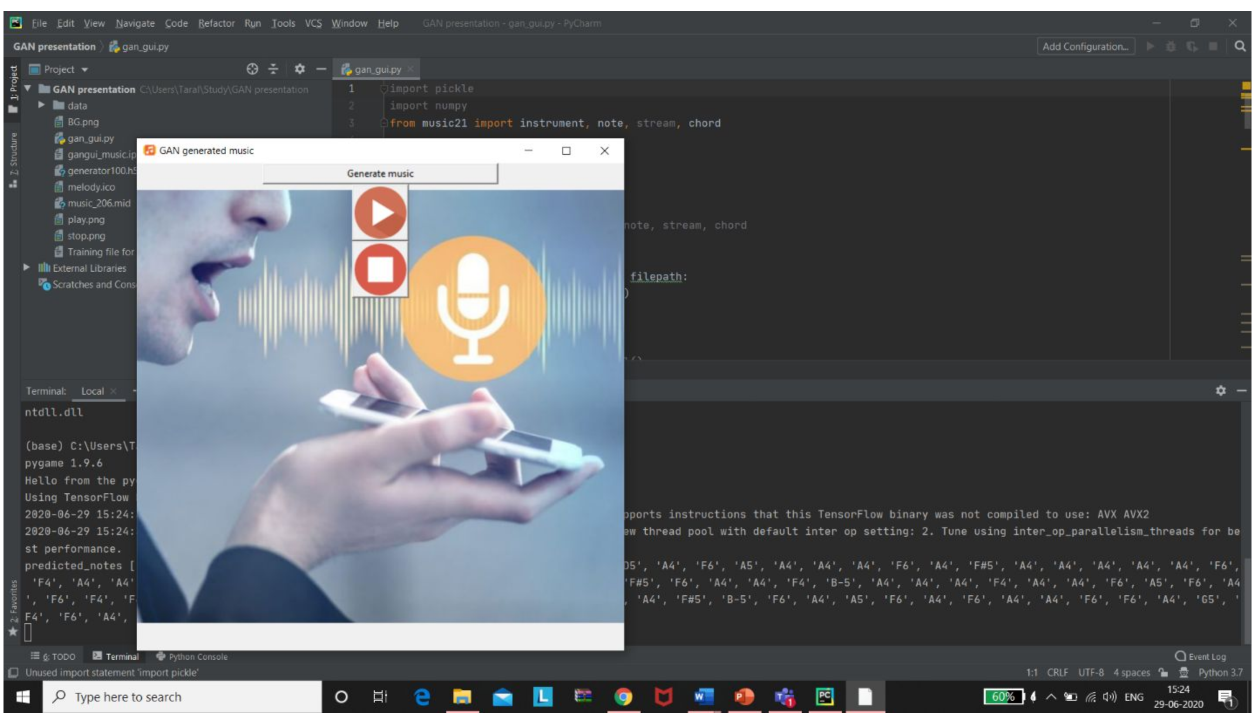
The second set of the evaluation system is to conduct a sample survey. We randomly selected a number of different styles of songs, mixed with some human-made music that people don't often hear, and invited our friends and colleagues to rate all the songs, although some people can tell which ones It is made by humans and which is machine generated, but they still make a high evaluation of machine generated music. A few people think that the music generated by the machine brings those surprises and shocks.

SCHEDULE

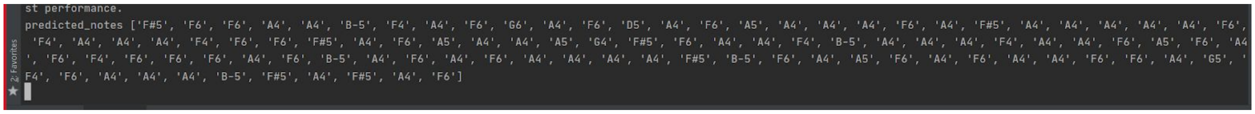
Task Name	From	To	Duration
Project Planning Proposal	02/04/2020	09/04/2020	1 week
Requirements Gathering and Specification	09/04/2020	23/04/2020	2 week
Design and Coding	23/04/2020	07/05/2020	2 weeks
Testing and Debugging	07/05/2020	15/05/2020	1 week
Implementation	15/05/2020	23/05/2020	1 week
Final Submission	24/05/2020	25/052020	1 days

SCREENSHOT OF PROPOSE WORK

Main window:

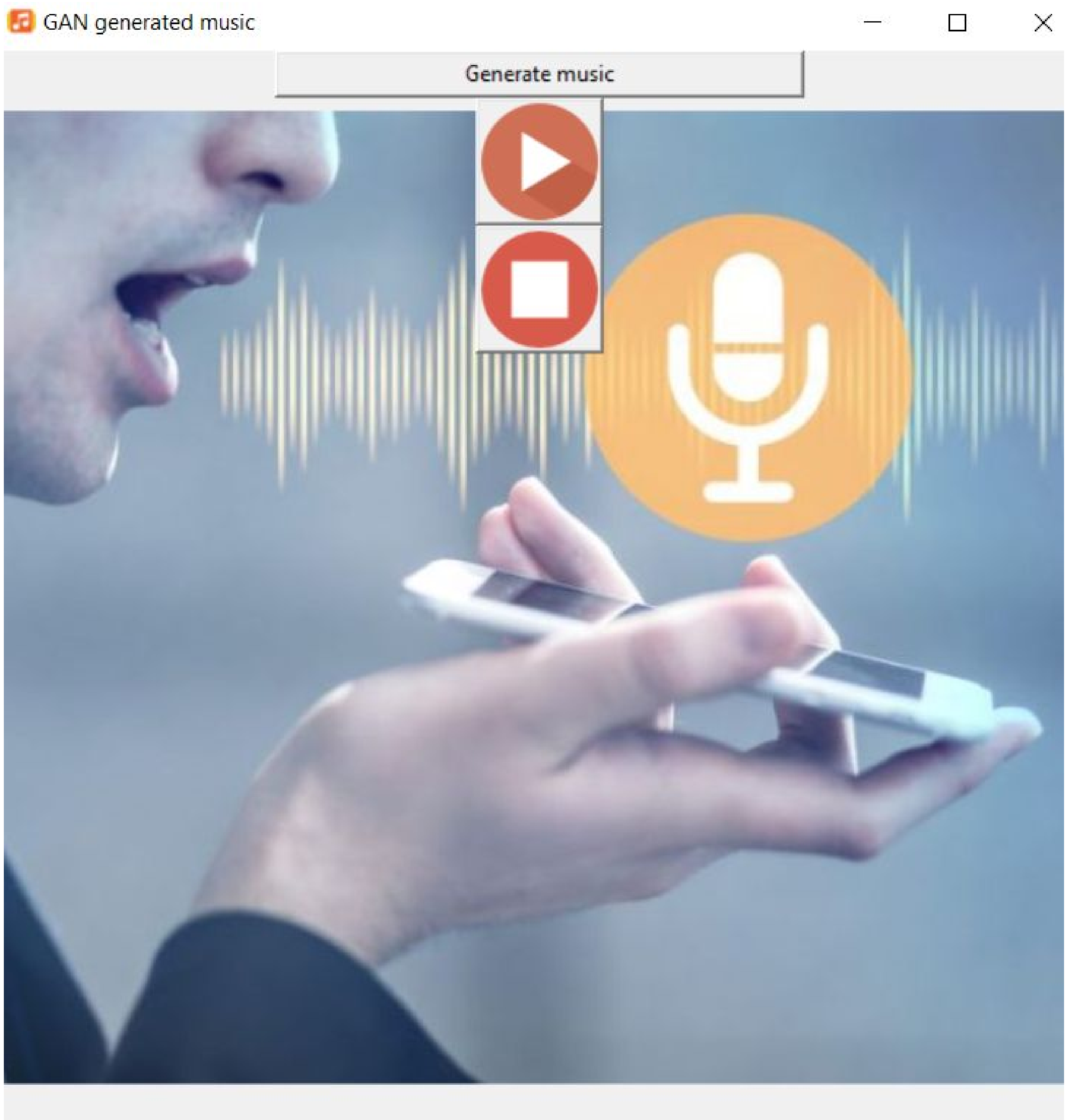


Notes Generated:



```
st performance.
predicted_notes ['F#5', 'F6', 'F6', 'A4', 'A4', 'B-5', 'F4', 'A4', 'F6', 'G6', 'A4', 'F6', 'D5', 'A4', 'F6', 'A5', 'A4', 'A4', 'A4', 'F6', 'A4', 'F#5', 'A4', 'A4', 'A4', 'A4', 'A4', 'F6',
 'F4', 'A4', 'A4', 'A4', 'F4', 'F6', 'F6', 'F#5', 'A4', 'F6', 'A5', 'A4', 'A4', 'A5', 'G4', 'F#5', 'F6', 'A4', 'A4', 'F4', 'B-5', 'A4', 'A4', 'A4', 'F4', 'A4', 'A4', 'F6', 'A5', 'F6', 'A4
', 'F6', 'F4', 'F6', 'F6', 'F6', 'A4', 'F6', 'B-5', 'A4', 'F6', 'A4', 'F6', 'A4', 'A4', 'A4', 'A4', 'F#5', 'B-5', 'F6', 'A4', 'A5', 'F6', 'A4', 'F6', 'A4', 'A4', 'F6', 'F6', 'A4', 'G5', '
F4', 'F6', 'A4', 'A4', 'A4', 'B-5', 'F#5', 'A4', 'F#5', 'A4', 'F6']
```

GUI:



LIMITATIONS

Through our LSTM network and 352 classes, we were able to achieve remarkable results. However, it can be improved in various areas.

First, we didn't consider the interval between notes to keep our network simple but to have more satisfying results. To achieve this we will have to add more classes for each and every duration and one extra class to represent to intervals.

Second, our network need to know how to handle unknown notes. Currently, our network would fail if encountered with a note that it is unaware of. A possible solution for this could be finding the note similar to unknown one.

More instruments can be added to the dataset to generate different types of music currently we tested it with only single instrument.

CONCLUSIONS

Conclusion for LSTM

We used simple LSTM based network to automate music generation. Results may not be perfect but are very good which shows that the neural network can be used to create music and has potential to produce higher complex musical extracts. LSTM proved to be a good model for capturing long-timescale dependencies. By providing musical note objects to our network, it was able to learn a musical style which was then used to generate the music.

Conclusion for GAN

We propose a generation model for generating note sequences under the GAN framework. We use the deep convolution neural network and optimize it according to the musical note characteristics, this optimization algorithm enables the convolution network to concentrate on learning music features and make experiments faster. At the same time, we also get some knowledge of ordinary sense to speed up the training of discriminators. Experimental data and subjective user evaluations show that proposed model can generate music like sequence generation. Although the experimental results are below the human level, the model is theoretically mature and has ideal properties. We hope that the subsequent research can be further Improve it.

FUTURE SCOPE

Future work can be done on the variants of LSTM and ensemble models which will require high powered GPUs

Efficiency improvements: training could be accelerated greatly by divising better methods for coordinating G and D or determining better distributions to sample z from during training. Further improvement can be done in post processing of the audio data. As, the audio data synthesis can be further improved by using techniques like phase shift, frequency modulation and a lot. This will increase the tone quality and melody in data.

In future efforts can also be taken to build various GAN models along with the audio data to generate music. Importance should be give to the final music generated.