MUSIC AUTO GENERATION

Kejia Wang **Yinan Gu** **Mengyang Zhang** **Jiarui Zhao**
University of Victoria
{kejiawang, yinangu, mengyangzhang, jiaruizhao}@uvic.ca

ABSTRACT

Song writing can be simpler and more efficient if it can be done automatically. Compose songs by importing multiple midi file songs, designed as a project that takes multiple midi sources as input and outputs a complete extended song. First, our project uses notation to numerically label the note in midi files so that computers can better read the files. Then analyze the characteristics and rules of song arrangement, and automatically create songs. Therefore, the project converts the sorted number string generated by the automatic authoring into the midi format and plays it. This requires using artificial intelligence techniques and training a database that can contain large amounts of music.

1. INTRODUCTION

Many music lovers tend to have enthusiasm about writing their own music but know little about music composing. Also, some song writers may find it time consuming to think of a complete song. Therefore, automatic song composing can reduce the effort of song writing in these situations. In our project, we want to achieve the goal of writing songs automatically using a given dataset.

In music writing, deconstruction and evaluation are the two most significant parts. To deconstruct, music elements such as pitch, rhythm need to be identified. The repetition and sequence of different parts of the music also needs to be recognized. To evaluate, it is important to judge the context, genres, and the composers' style of the songs.[1]

We need to identify what components of the sound source are crucial in transforming it into music nodes, such as the frequency and duration of each music node. Then utilize AI techniques to build a model and train a dataset for composing the song, so it will automatically analyze the most suitable next music note of the current note, and save the note list to a midi file which will be our new song.

As the example given by Figure 1, AI intelligent data analysis, smart furniture, driverless cars, etc. Nowadays, many Internet companies will analyze everyone's hobbies and habits through AI data analysis, and then use these data to push the product content that everyone is interested in. AI can continuously improve itself through learning and improve itself step by step.

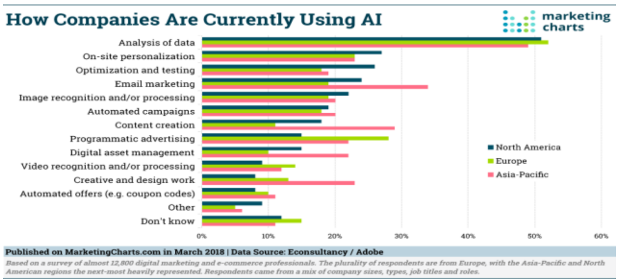

Figure 1. Example of products that use AI [2]

AI technology is now also used in automatic generation. One of the products uses RNN automatic generation technology, which can automatically generate ancient poems by loading a large amount of data and learning for a long time [5]. After loading a lot of data from ancient poems, the text cannot pass through the RNN directly [5]. It needs to be awakened and encoded by one-hot, and then it can be passed through the RNN after it is turned into a number [5]. In RNN, it predicts what words should be followed by each sentence by guessing [5]. After completing this step, wake up the decode through de-one-hot, turn the number into text, and finally output it as text that we can read.

Since ancient poems can be automatically generated by this method, I think a similar method can be used to make a software that automatically generates music. Many songs can be characterized into a certain kind of genre, and many song writers have their specific composition style and pattern. We can utilize this and AI techniques to automatically generate melodies. Our project is designed to allow AI to automatically create some new simple music by storing large amounts of data and learning by itself. We are planning to use Keras to build the LSTM network [2-6]. With the recent breakthroughs in data science, it is discovered that for most of the sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been found as the most effective way to solve them [7]. Getting plenty of input of music and encode them to the language computer could understand and use the same technology as RNN project to create new numbers and then decode them to real music [5].

2. DATASET

To compose a suitable song, the first task is creating a dataset. This dataset consists of different music, which will be used for training and testing throughout the project. Therefore, composing this dataset with as many different types of music as possible to ensure the richness of this dataset is very important. In order to facilitate the subse-

Attribution: K. Wang, Y. Gu, M. Zhang, and J. Zhao, "Music Auto Generation", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

quent processing of the dataset, the music format in the dataset should be a MIDI Music file. We find 110 different styles of music as our dataset.

We find 110 different styles of music as our dataset. The dataset is from the Music MIDI Collection on kaggle, which is provided by NEEL REDKAR and created by Fonzi and Kenzi [8]. Part of the data is shown below.

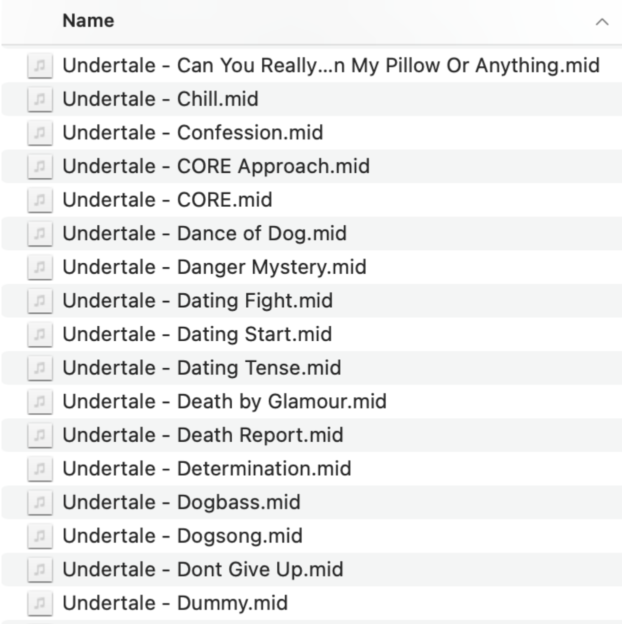

Figure 2. Part of dataset

We can understand it as every piece of music is made up of different notes [9].

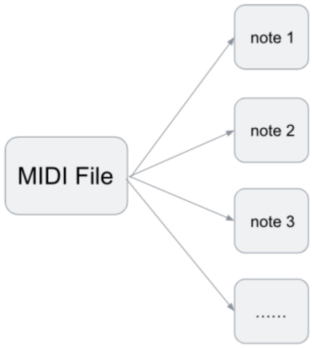

Figure 3. Composition of music

For example, for the music Alphys.mid, it consists of the following notes:

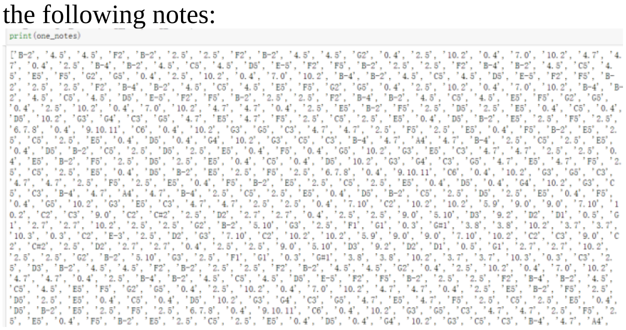

Figure 4. Notes of Undertale - Alphys.mid.

Each character in the above figure (such as '4.5', 'C5', 'F2'), we can consider it as a note. Many notes make up a piece of music.

3. PROCESS

Sometimes we may hum some melody and it is random, you may even detect that you are familiar with some pieces of the fashion music. By taking some pitch into the machine to see how it creates a complete and new melody based on your music piece. Also, you can cut some pieces of the fashion songs and input them into the machine to see how the algorithm would process it and what kind of music it will produce.

In our project, we make part of the training set the input of the model and because the notes are randomly picked from the training set, it can be viewed as a new piece of music.

3.1 Convert Data Sets format

After creating the dataset, since our input is a piece of music in MIDI format, and the computer cannot read the MIDI format directly, we need to train the dataset into a usable format--convert the music file in MIDI format into a "note" array [10]. This data processing is necessary. Convert the music datasets that people usually listen to into a format that the computer can understand. After that, each piece of music is composed of many notes.

The specific implementation method is implemented using the mu-sic21 library in python. Therefore, we define the get_notes function to turn all mid files in the dataset into an array called all_notes. The specific implementation process is as follows:

If there is an instrument part, take the first instrument part.

If there is no instrument part, take note directly.

[1] If it is a Note type, like

take the note.

[2] If it is a Chord type, like

take the serial number of the note.

Call get_notes and use the music21 library to turn all the MIDI files in the folder into an array of notes, but in practice this process is relatively slow. Therefore, we consider that the converted note array can be saved for the first time for subsequent use. We define functions for saving (save_data) and reading (get_data). Calling these three functions successfully converts the mid file into a note array and saves the note array.

In addition to defining the function that converts the mid file into a note array, we should also define a function that converts the note array into a MIDI Music file [11], which is used to convert the note array generated by the model into music that people can hear.

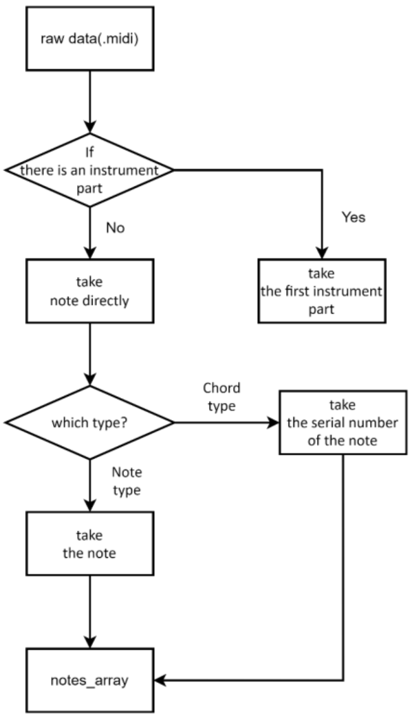

Figure 5. Process of converting raw data to note array

3.2 Analysis Data Sets

Since the conversion of the MIDI Music file into a note array may be slow each time, saving the note array after the first conversion is a good way to solve this problem. The process principle is to read a song by taking a string with a specified length. Each string is stored in X for analysis, and stored one digit after each string for Y. As shown in the figure, [1, 15, 8, 9, 10, 1, 1] is X as the length is 7, the one dight after X is 9, then stored 9 in Y. For code, we take string length for 100.

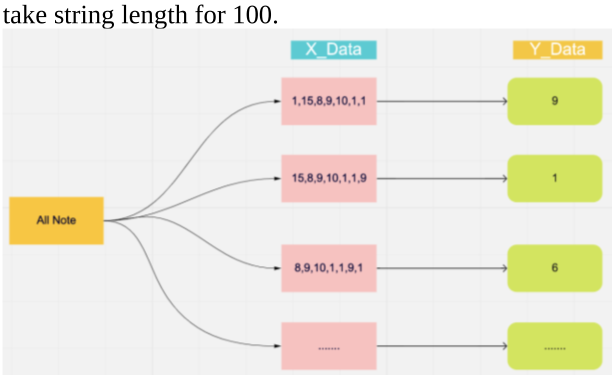

Figure 6. Process principle of constructing a dataset

Next, since the LSTM network cannot handle the note array directly, in order to put the data into the LSTM network for training, we need to number the notes [12]. Since each note needs to be one-hot encoded, the notes need to be numbered. When each note has an id number, it is easy to one-hot encode each note [13]. First, we number each note, such as "A5" is numbered 0, "F5" is numbered 4. When each note has an id (such as 0 for "A5" and 4 for "F5"), each note can be one-hot encoded. At the same time, the note needs to be converted into an id while reading the data. That is to say, the last in the data is not the note but the id. Specifically, we convert the note to id while fetching data. So, In the end, the data in X_train and Y_train is not a NOTE but an ID. One-hot encoding, here use Keras tools. X_one_hot and Y_one_hot are the final LSTM data.

3.3 Modelling

A feed-forward network can not store the past information [14][15][16], which will make it difficult to detect where the period should be and the computer would fail to learn the relationship between the inputs and ends up composing bad music.

Also, due to the problem of vanishing gradients [17] in standard Recurrent Neural Networks (RNNs), we decide to choose Long Short -Term Memory (LSTM), which is a special type of RNNs. It can resolve the Long-Term Dependencies [18]. This would make the produced music more sweet for humans.

For modeling, we will import Keras which is a high-level neural networks API in Python to build the LSTM networks.

We will also use the toolkit Music21[19] in Python to help us create Note and Chord objects and for extraction. We first want to create the new chord music to play the notes generated according to the input notes.

However, due to the strict shape the model requires, we unfortunately have to use the random notes picked from the training set as the input to generate new music.

Note is an object that contains the information of sound's frequency, octave and offset. An offset can refer to the beats which imply the location in the piece. A chord is a set of notes that are played at the same time.

To predict the next note or chord, we need to collect all the notes and chords in the training dataset and ideally make them as large as possible.

We have tried collecting the information separately to an array. The first step we tried is to make each of the music file a stream object by the function *converter.parse(filename)*, then we can do the extraction of notes and chords.

We first try to make the input to the model a sequential list of notes and chords. The model we are now building is to predict the following 1000 notes and use them to generate the music file.

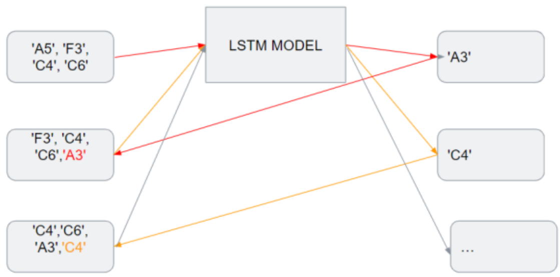

Figure 7. How to get the predictions

As the example in figure 7 shows, we use the first 4 notes to predict the next note according to the largest possibility, then the new note will be used to update the list of notes to predict the next note. The first note in the old list, in the example, ‘A5’ will be deleted and the new list would be {‘F3’,’C4’,’C6’,’A3’} and so on and so forth.

For building the model, it will have four different types of layers:

1. LSTM layers
2. Dropout layers
3. Dense layers
4. The Activation layers

A LSTM layer will make sure the model returns a sequence with the input sequence. A dropout layer is going to avoid overfitting of data. A dense layer is used to connect all the input nodes to the output nodes. The activation layer is used to give the function of calculation of the output.

In this project we tried with 6 layers: 1 input layer, 2 LSTM layers, 2 dropout layers, 1 dense layer:

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100, 441)]        0
_________________________________________________________________
lstm (LSTM)                  (None, 100, 512)          1953792
_________________________________________________________________
dropout (Dropout)            (None, 100, 512)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 256)               787456
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 441)               113337
=================================================================
Total params: 2,854,585
Trainable params: 2,854,585
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. Model Structure

3.4 Training

We take all the data from the database into the designed Long Short-Term Memory network and set some threshold values to modify the parameters of the models so that it can work well on the input values.

For example, we set the learning rate in Adam to be 0.001. The input and output should be sequence of integers [20]. We tried running the code in personal computer and it takes a lot of time without GPU configuration:

```
Epoch 1/100
23/23 [==============================] - 1586s 69s/step - loss: 5.0601 - accuracy: 0.0246
Epoch 2/100
23/23 [==============================] - 1526s 66s/step - loss: 4.7448 - accuracy: 0.0217
Epoch 3/100
23/23 [==============================] - 1474s 64s/step - loss: 4.7186 - accuracy: 0.0230
Epoch 4/100
23/23 [==============================] - 1341s 58s/step - loss: 4.7055 - accuracy: 0.0232
Epoch 5/100
16/23 [==================>..........] - ETA: 7:31 - loss: 4.6983 - accuracy: 0.0239

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-40-1cc17c6dbb07> in <module>
      8 )
      9 callbacks_list = [checkpoint]
---> 10 model.fit(X_one_hot, Y_one_hot, epochs=100, batch_size=2048,callbacks=callbacks_list)
```

Figure 9. Training Process

During the training step, we learnt from online resources that we can shut down the training process when it is running without losing any previous work by using the checkpoints. It will help us save the weights of the network before we stop the training manually.

When training the model, we need to have a function to calculate the loss at each time through one training. Considering that our output is going to be a sequence of notes and chords where each of them is going to be mapped to a single class and be encoded by one-hot, then one way to do it is the ***categorical cross entropy*** [21].

3.5 Music generation

After the training of the model, we will finally have a machine used for producing music based on a single piece of music. We will convert the MIDI input into a sequence of sequence of notes, and then we will give each note its integer id and then the sequence of notes will be transformed into a sequence of integers. Later this sequence of integers will be used to generating new sequence of integers, and further transform into notes and output as a new MIDI file. After the model is trained, we can try to feed it with a sequence, and finally get an array of encoded notes and chords. We need to decode them and append them to a stream in music21 object and then transform to the MIDI file. [22]

Because most chords are evolved from several tonic chords. And all chords have different arrangements and different numbers of constituent notes. This software will automatically use different chords to generate new songs according to the number of chords that the user selects. At first, we considered the method of producing the music in all single note form first, then add chords to some specific positions that fits the rhythm. So, we wanted to solve the problem of adding chords to a given single note melody.

To do this, we tried to figure out how to find a chord that is suitable for a single note. Also, where in the melody needs to add chords. In addition, considering there are many possibilities of chord choices for one single note, what is the most suitable choice for a specific melody.

As a result, we mainly aim to use the following two kinds of three-key piano chords.

Common major piano chords include:
C major (C). C - E - G
C# major (C#). C# - E# - G#
D major (D). D - F# - A
Eb major (Eb). Eb - G - Bb
E major (E). E - G# - B
F major (F). F - A - C
F# major (F#). F# - A# - C#
G major (G). G - B - D
Ab major (Ab). Ab - C - Eb
A major (A). A - C# - E
Bb major (Bb). Bb - D - F
B major (B). B - D# - F#
Common minor piano chords include:
C minor (Cm). C - Eb - G
C# minor (C#m). C# - E - G#
D minor (Dm). D - F -A
Eb minor (Ebm). Eb - Gb - Bb
E minor (Em). E - G - B
F minor (Fm). F - Ab - C
F# minor (F#m). F# - A - C#
G minor (Gm). G - Bb - D
Ab minor (Abm). Ab - Cb - Eb
A minor (Am). A - C - E
Bb minor (Bbm). Bb - Db - F
B minor (Bm). B - D - F#
[23]

However, during model building and training, we found that it was not necessary to add chords individually as an extra step after forming the music in note form. So, we decided to abandon our original approach and ended up storing data without discriminating between chords and monophonic. Then let the LSTM process find the best match for the next pitch, which can be a chord or a single note. Because when we use the computer to read and mark the notes, we store the musical notes in the address, compare and mark the addresses of the musical notes, and analyze the arrangement law. So, if the user takes chord music as input, the output will be chord music. Conversely, if the user uses monosyllabic music as input, the output will be monosyllabic music.

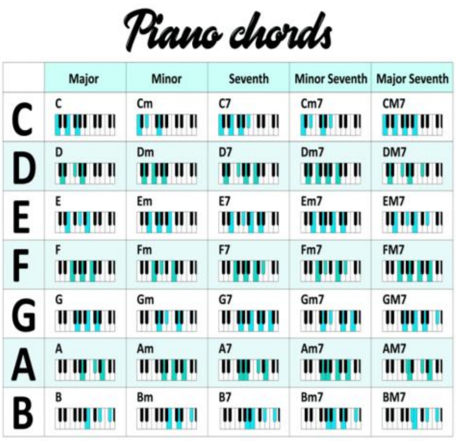

Figure 10. Piano Chards table [24]

The primary part of the music generation process is using the LSTM model to predict the next note with some given notes. We use a sequence containing certain number of music notes as input, put it into the LSTM model, and analyze the next note that suit the most. Adjust the sequence with the new note appended and feed the model again. Continue with the process and finally we can get a full new list of notes that is autogenerated, this can be further output to a MIDI file. For example, it we feed the sequence 'A5', 'F3', 'C4', 'C6', and the model gives 'A3' as next node, then we change the sequence to 'F3', 'C4', 'C6', 'A3', removing the front end note and adding the new note to the end, then feed this again to the model and get 'C4' as new note. The predicted notes will be in the form of integer id. If the dataset contains chords, then it may have note combination such as '2.5.9' which is a three-note chord.

4. EXPERIMENTAL RESULTS

This project aims to automatically generate pieces of music, we would have liked to input speech and extend the melody into a more complete song, but the human ear and brain are very complex. Due to the binaural effect, people can tell whether the sound is coming from the left or the right. At the same time, with the presence of the pinna, one can distinguish whether the sound is coming from above or below, from the front or the back. Not to mention that the process by which the brain analyzes audio is a series of Fourier-like transforms [25]. So, people who can listen to music and realize the notes in the music by their ears are amazing. But if people can do it, so can machines. There are much software on the market that can perform computer recognition of musical scores. For example, "Ajay pro ai, adobe audition cs6, logic pro x" We try to use the same vocal to make a score, music recognition, they can use m1's neural network engine to separate vocals, drums, accompaniment, and other songs, and generate a score. [26] [27] [28]. And it's very accurate. But we got stuck in the process of building the model and matching the pattern. There is no doubt that it is difficult to make a computer comparable to the human brain because the human brain is a more complex system and structure than a computer. Also, when the human voice is humming as input, there are many unstable factors, similar to the noise and noise of the environment, the pitch and timbre of the human voice, and so on. At present, we do not have a good way to control these factors so that our software can recognize human voices stably and correctly, and because of time constraints. So, we turned in the other direction, skipping vocal recognition into musical scores, i.e. composing new melodies based on a given dataset of MIDI files. We also attempted to analyze the MIDI files in the dataset using the monophonic pitch extraction method. We later found it more convenient to use the music21 library, but the process of trying it was an invaluable experience for understanding the key components of a music file.

The project manages to automatically generate music and output a list of notes computed using an LSTM model since the model only puts notes into the computation, where notes have no other obvious characteristics such as amplitude or duration. We use the stream object from music 21 to put the results together, then use the write function to convert the results to MIDI sound output. The sound output we have is in the form of consecutive single notes and chords with the same amplitude and duration. While it loses some of its musical characters, the result doesn't sound too weird or harsh. The combination of notes is harmonious, which is the main achievement of LSTM technology, and it has the potential to turn into better, more complete melodies.

We also want to output musical notation (staves). If we also solve the way to add duration property, the notes can have time signature, and the note list can be constructed as a musical score [29]. If it has a duration property, it can be constructed as a musical score, making it easier to play on instruments like the piano. But our output can also be used for inspiration or to add a duration attribute later to complete music composition.

5. CONCLUSION

In this project, we have tried many ways to achieve our initial goal: get humming pitch and use it as a basis to create new music. The basic idea behind it is to use some deep learning structure to generate music according to some music. So, we focus on it more in our project. The model we built is far from perfect and it also neglect the beats. There are still a lot of parameters remained in the Keras to use to improve the model.

During modelling and training, it's a big challenge to get the data to fit the shape the machine needs. We tried to input some extra music that is not in our training dataset, and it kept crashing due to some dimension-unmatched issues. We still need to learn more about the LSTM model and see how it works for more details.

The dataset we used is not typical. If we want to generate the music that of all kinds, we would like to store all the notes and chords that exist in the world, which is immense work, and the model would need to be more exquisite to utilize those large storage.

Actually, we can bring it down a bit. for example, if we want to generate the classical music, then, we may use the dataset that is composed with all typical classical music, then we will have a set of notes and chords that is most likely in classical music, then the model trained on it would be more accurate.

We propose a task of automatically generating music with LSTM machine learning. We learned how to use Keras to build an LSTM model and then automatically generate music. However, due to limited knowledge and limited computer capabilities, we can only generate simple, short pieces of music, but not more complex, popular music. In the process of reviewing the literature, we found that someone has developed a computer neural network that generates lyrics through artificial intelligence [30]. We believe there will be more mature AI-generated music in the near future, and we will continue to learn about it.

6. PROJECT TIMELINE AND ROLES

February.20 Analyze voice source and characterize it. Generate a way to extract the music piece into a music sheet.
March.15 Complete code for database analysis
March.25 Complete code for composing music
April.5 Finish debugging and prepare for presentation

Team Member Name	Contributed By
Yinan Gu	LSTM Modelling and training
Kejia Wang	Dataset management
Mengyang Zhang	LSTM note prediction
Jiarui Zhao	Conversion of formats

7. REFERENCES

[1] "Musical Analysis Writing Guide 2012 edition," *Music Industry College*. [Online] Available: http://mic.org.au/wp-content/uploads/2012/01/MUSICAL-ANALYSIS-WRITING-GUIDE-2012-edition2.pdf . [Accessed: 08-Feb-2022].

[2] "Marketers are using AI primarily for data analysis," *Marketing Charts*, 01-Mar-2018. [Online]. Available: https://www.marketingcharts.com/customer-centric/analytics-automated-and-martech-82540 . [Accessed: 08-Feb-2022].

[3] Keras-Team, "Keras-Team/Keras: Deep Learning for Humans," *GitHub*. [Online]. Available: https://github.com/keras-team/keras . [Accessed: 08-Feb-2022].

[4] Skuldur, "Skuldur/classical-piano-composer," *GitHub*. [Online]. Available: https://github.com/Skuldur/Classical-Piano-Composer . [Accessed: 08-Feb-2022].

[5] "Yufeng's blog," *Online CSC Community*. [Online]. Available: https://www.cnblogs.com/xiaohuiduan/p/14330637.html . [Accessed: 08-Feb-2022].

[6] "Understanding LSTM networks," *Understanding LSTM Networks -- colah's blog*. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ . [Accessed: 08-Feb-2022].

[7] Pranj52, "Essentials of Deep Learning : Introduction to Long Short Term Memory" *Analytics Yidhya* [Online]. Available: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/?utm_source=blog&utm_medium=how-to-perform-automatic-music-generation. [Accessed: 18-Apr-2022].

[8] REDKAR, N., 2022. Music MIDI Collection. [online] Kaggle.com. Available at: <https://www.kaggle.com/datasets/programgeek01/anime-music-midi> [Accessed 18 April 2022].

[9] Hewahi, N., AlSaigal, S. and AlJanahi, S., 2019. Generation of music pieces using machine learning: long short-term memory neural networks approach. Arab Journal of Basic and Applied Sciences, 26(1), pp.397-413.

[10] I. Krause and R. Sebastian II, "Capstone Project: Real Time Musical Audio Detection, Conversion & Transcription,".

[11] N. Itou and K. Nishimoto, "A voice-to-MIDI system for singing melodies with lyrics," in Proceedings of the International Conference on Advances in Computer Entertainment Technology, 2007, .

[12] N. Kotecha and P. Young, "Generating music using an LSTM network," arXiv Preprint arXiv:1804.07300, 2018.

[13] M. Nadeem, A. Tagle and S. Sitsabesan, "Let's make some music," in - 2019 International Conference on Electronics, Information, and Communication (ICEIC), 2019, . DOI: 10.23919/ELINFOCOM.2019.8706447.

[14] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, pp. 48, 2002.

[15] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, *(4)*, pp. 27-31, 1994.

[16] Briot J P, Hadjeres G, Pachet F D. Deep learning techniques for music generation--a survey[J]. arXiv preprint arXiv:1709.01620, 2017.

[17] S. Hochreiter *et al*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.,* vol. 9, *(8),* pp. 1735-1780, 1997.

[19] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," 2010.

[20] Seger C. An investigation of categorical variable en-cod-ing techniques in machine learning: binary ver-sus one-hot and feature hashing[J]. 2018.

[21] Gordon-Rodriguez E, Loaiza-Ganem G, Pleiss G, et al. Uses and abuses of the cross-entropy loss: case studies in modern deep learning[J]. 2020.

[22] "User's Guide, Chapter 4: Lists, Streams (I) and Output," *music21* [Online]. Available: https://web.mit.edu/music21/doc/usersGuide/usersGuide_04_stream1.html. [Accessed: 18-Apr-2022].

[23] "Piano Chords for Beginners," School of Rock. [Online]. Available: https://www.schoolofrock.com/resources/keyboard/piano-chords-for-beginners. [Accessed: 20-Mar-2022].

[24] J. C., "13 basic piano chords for beginners (easy)," *Music Grotto*, 04-Jan-2022. [Online]. Available: https://www.musicgrotto.com/basic-piano-chords/. [Accessed: 20-Mar-2022].

[25] N. Lenssen, "An introduction to fourier analysis with applications to music." [Online]. Available: https://scholarship.claremont.edu/cgi/viewcontent.cgi?httpsredir=1&article=1142&context=jhm. [Accessed: 19-Apr-2022].

[26] C. Cleyn, "Logic pro X tutorial - everything you need to ... - youtube." [Online]. Available: https://www.youtube.com/watch?v=nZ8UtZNOrfY. [Accessed: 19-Apr-2022].

[27] M. Russell, "Adobe Audition CC Tutorial for beginners - youtube." [Online]. Available: https://www.youtube.com/watch?v=en3qUvJG42s. [Accessed: 19-Apr-2022].

[28] T. G. Official, "Algoriddim Djay Pro Ai Tutorial: Overview - timmyg - youtube." [Online]. Available: https://www.youtube.com/watch?v=uqjMte4UECo. [Accessed: 19-Apr-2022].

[29] "How to Write Sheet Music," WikiHow [Online]. Available: https://www.wikihow.com/Write-Sheet-Music. [Accessed: 18-Apr-2022].

[30] J. Briot, G. Hadjeres and F. Pachet, "Deep learning techniques for music generation--a survey," arXiv Preprint arXiv:1709.01620, 2017.