Project No.20

Development of a Machine Learning Application for Composing Piano Songs

Presented by

1. Poonnaphop Benjalert 58070503420
2. Kittipat Ariyaprayoon 58070503440

Advisor

Asst. Prof. Jumpol Polvichai, Ph.D.

"I've read and approved the content of this report"

…….

(……)

Advisor

Development of a Machine Learning Application for Composing Piano Songs

Mr. Poonnaphop Benjalert
Mr. Kittipat Ariyaprayoon

Project Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi
Academic Year 2018

Development of a Machine Learning Application for Composing Piano Songs

Mr. Poonnaphop Benjalert
Mr. Kittipat Ariyaprayoon

A Project Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi
Academic Year 2018

Project Committee

…… Advisor
(Asst. Prof. Jumpol Polvichai, Ph.D.)

…… Committee
(Asst. Prof. Sanan Srakaew)

…… Committee
(Asst. Prof. Dr.Suthathip Maneewongvatana, Ph.D.)

…… Committee
(Asst. Prof. Dr.Phond Phunchongharn, Ph.D.)

Project Title	Development of a Machine Learning Application for Composing Piano Songs
Project Credit	3 credits
Project Participant	Poonnaphob Benjalert Kittipat Ariyaprayoon
Advisor	Asst. Prof. Jumpol Polvichai , Ph.D.
Degree of Study	Bachelor's Degree
Department	Computer Engineering
Academic Year	2018

Abstract

"Inspiration" is the most important factor beside from pitch and rhythm which are normally considered to compose a music.To help finding the inspiration is done routinely by listening some other music. We want to create an application is planned to generate music selected by different kinds of moods in order to make inspiration for everyone by using a technique in artificial intelligence. The RNN (Recurrent Neural Network) is a deep learning technique is applied for composing the music in this application. This application will produces the music without lyrics typically performed by the piano. By using different varieties of pitch and rhythms; happy, sad, and calm are moods that users may feel after using the application. In addition, to test our application, a web application is implemented by selecting the designed mood, time duration and the first note at the start. To evaluate our work, a number of users are required to use this application to see if they enjoy or accept the accurate mood or not. Finally, in the end, the application would hopefully create some inspiration to everyone who love music.

หัวข้อโครงงาน	พัฒนาโปรแกรมการเรียนรู้ของเครื่องคอมพิวเตอร์เพื่อที่จะสามารถแต่งเพลงเปียโน
หน่วยกิตของโครงงาน	3 หน่วยกิต
จัดทำโดย	นาย ปุณณภพ เบญจเลิศ นาย กิตติพัฒน์ อริยประยูร
อาจารย์ที่ปรึกษา	ผศ.ดร.จุมพล พลวิชัย
ระดับการศึกษา	วิศวกรรมศาสตรบัณฑิต
ภาควิชา	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	2561

บทคัดย่อ

"แรงบันดาลใจ" คือตัวแปรสำคัญที่สุดนอกเหนือจากจังหวะและระดับเสียง ซึ่งเป็นสิ่งธรรมดาที่ต้องให้ความสำคัญในการที่จะแต่งเพลง. ซึ่งการที่จะหาแรงบันดาลใจได้นั้น สามารถหาได้จากการฟังเสียงดนตรี.เราจึงต้องการที่จะสร้างโปรแกรมที่สามารถแต่งเพลงโดยเลือกได้ตามอารมต่างๆเพื่อที่จะเป็นแรงบันดาลใจสำหรับทุกคน โดยใช้ปัญญาประดิษฐ์เข้ามาช่วย เครือข่ายประสาทเทียมแบบวนซ้ำคือการเรียนรู้เชิงลึกอย่างหนึ่ง ซึ่งเป็นหนึ่งในสาขาของการเรียนรู้ของคอมพิวเตอร์ (ปัญญาประดิษฐ์) ได้ถูกนำมาประยุกต์ใช้สำหรับการสร้างโปรแกรมแต่งเพลง. โปรแกรมนี้จะผลิตเพลงออกมาโดยมีแค่เสียงเปียโน ไม่ได้มีเนื้อร้อง โดยการใช้จังหวะและระดับเสียงที่แตกต่างกันจำนวนมาก. "เพลงแนวมีความสุข","เพลงเศร้า","เพลงผ่อนคลาย" คืออารมณ์ที่ผู้ใช้สามาถรรู้สึกได้หลังจากได้ใช้โปรแกรมแต่งเพลงนี้ นอกจากนี้, การทดสอบโปรแกรมของเรานั้นจะใช้ในรูปแบบของเว็บแอ๊พพลิเคชันในการทดสอบโดยการให้ผู้ใช้ได้เลือกอารมณ์, ความยาวนานของเพลง และ โน๊ตตัวแรงสำหรับเพลงนี้. ส่วนด้านการประเมินผลนั้น, เราจะนำเว็บแอ๊พพลิเคชันแต่งเพลงของเราไปให้ผู้ใช้ได้ใช้งานและเก็บผลของผู้ใช้ ว่าผู้ใช้เพลิดเพลิน และยอมรับเพลงว่าตรงตามอารมณ์ที่ผู้ใช้ได้เลือกไว้หรือไม่. และในท้ายที่สุดนี้ทางผู้จัดทำหวังว่า แอ๊พพลิเคชันนี้จะสามารถที่จะสร้างแรงบันดาลใจให้แก่ทุกคนที่รักในเสียงเพลง.

Acknowledgements

First, We would like to express our gratitude toward our advisor Asst. Prof. Jumpol Polvichai whom always there to help us. When we need an advice, had a question about our research or a problem in writing. He always support us even in times that he is busy, he still finds some time or alternative way to help us do our research such as TAs and Facebook comment. We are very grateful for what you've done for us Thank you.

Secondly, we owe our deep gratitude to all of our committees: Asst. Prof. Sanan Srakaew, Dr.Suthathip Maneewongvatana, and Dr. Phond Phunchongharn, whom interest in our project and guided us all along until the completion. By providing all the necessary information for developing a good system, give us advice and comment to make our project and ourselves to become better.

Last, we would like to thank our friends and family who, support us and help us in any other way besides the project, such as encouragement, guideline, and many other ways. We are very grateful toward all of them thank you.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Problem Statement and Approach

When composing music the person should have some background knowledge in the music field such as Rhythm, Pitch, Harmony, and other music theory and music components. Beside that other important thing when composing music is “inspiration”. It is an initial spark of any music composing and without it, there is no music.

Sometimes, when people have no inspiration and they need to seek for one. There are many method people use to find inspiration such as, watching TV, reading the novels, observing the world or by simply listening to other pieces of music.

We are interested in creating a machine learning application that people can use to compose pieces of music based on the mood or feeling. In addition, the application is built intended not only for anyone to have fun but for musicians that want to have a musical guideline inspiration.

1.2 Objective.

- To create an application for generating music selected by different kinds of moods in order to make inspiration for everyone by using a technique in artificial intelligence.
- To learn more about machine learning and apply for the real world problem.

1.3 Scope

- A simple web application that will be deployed only on localhost.
- We will focus on Recurrent Neural Network for our machine learning method.
- Create a dataset containing 1500 songs and divide into different kind of moods.
- The type of moods is happy, sad and calm.

1.4 Tasks and Schedule

1. Research on how to compose a piece of music.
    a. Most of the music component, music theory and other factors which make the song become melodic.
2. Research on RNN Model.
    a. Research on the model process and how to implement the model.
    b. Research for many types of RNN to use too such as Bi-directional RNN, LSTM that can benefit more.
3. Gather and clean music dataset and separate by mood.
    a. Gather the piano instrument song in the format of MIDI.
    b. Sort the song into 3 moods so that each mood contain 500 songs.
4. Design and develop a prototype model that can compose simple music.
    a. Plan the machine learning process and model.
    b. Develop a prototype.
    c. Test the prototype and get the feedback.
5. Design Web-application User interface.
6. Develop a RNN and train model using cleaned dataset.
    a. Create a hidden layer for the RNN model

   b. Train the model with the cleaned dataset
7. Create web-application for user interaction.
   a. Design the UI of the web application.
   b. Create core or back-end that is integrated with the train model.
   c. Create front-end based on the design.
8. Test the program with users and get feedback.
9. Conclude the project

From the tasks breakdown above we can separate our work into two semesters. Table 1.1 is a schedule for the first semester. The main focus is to study and develop a simple model to evaluate and prepare for the second semester.

Table 1.1: Semester 1 Schedule

Activity / Week	August				September				October				November				December			
1. Research on how people compose a music.																				
2. Research about RNN model.																				
3. Gather and clean data then separate by moods.																				
4. Design and develop a prototype model.																				
5. Design Web-application User interface																				

Table 1.2 is a schedule for the second semester. The main focus is to use the knowledge from the first semester to develop a model and create a web application for user interaction with the model. Another focus is to evaluate and conclude the project.

Table 1.2: Semester 2 Schedule

Activity / Week	January				February				March				April				May			
6. Create a finalize model to be use in the web application																				
7. Create web application for user interaction																				
8. Test program with users and get the feedbacks																				
9. Conclude the project																				

1.5 Deliverables for Term 1

- A cleaned music dataset divide by moods.
- A designing of Web-application UI.
- A prototype of the machine learning model that can compose songs.

1.6 Deliverables for Term 2

- Develop machine learning that can compose music base on mood.
- Create Web-application for user interaction.
- Test our program with the user and get the feedback.

Chapter 2
Background, Theory and Related Research

2.1 Music Theory

2.1.1 Pitch

Pitch is a high or low sound that people can hear in a piece of music. A pitch can be described using a frequency. From Figure 2.1, it shows that each note in the music will have a different frequency, therefore, each note contain a different pitch.

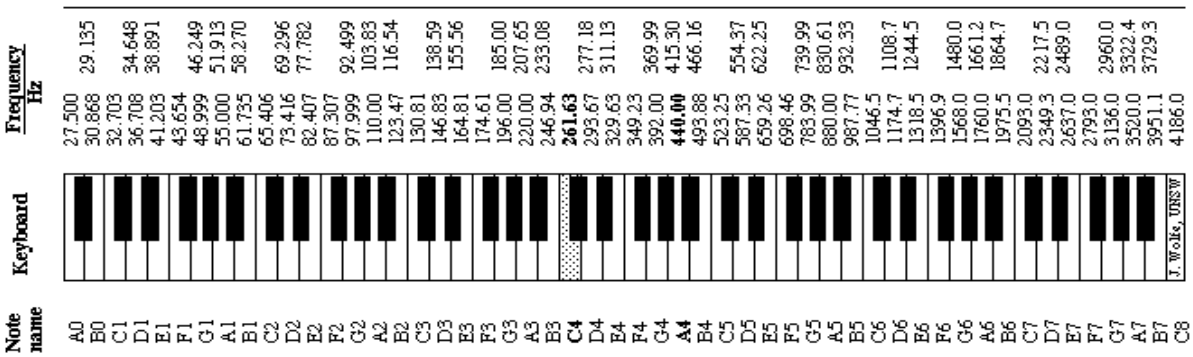

Figure 2.1: Frequency of each note [5].

2.1.2 Melody

Melody is the “singable” flow of sound. It is the tune, or musical line of notes, that our brains hear as one unit. A melody is an organized sequence of pitches. There is two kinds of melody, conjunct (smooth and easy to sing), disjunct (large leap and jump to many pitches).

2.1.3 Rhythm

Rhythm is one of the elements in music. It is a time-based element of the music, the rhythm is referred to as how long of the music events occur. The music events can be a single note or a sequential note. Rhythm is when the music event start and end. There are several aspects in rhythm such as Metre, Beat, and Tempo.

2.2 Musical Instrument Digital Interface (MIDI)

MIDI is a technology for a musician that allows electronic instrument and other digital musical tools to communicate with each other. MIDI file does not make the sound but It contains a list of messages like “note on”, ”note off” or ”pitch” that tell electronic device, which note should be played. MIDI can define into 3 paths: MIDI messages, The physical transports for MIDI and file formats of MIDI

2.2.1 MIDI messages

The MIDI Messages is the most important part of MIDI. The MIDI Messages are described the note message that tells MIDI device what note to play and it contains velocity that tells the MIDI device how loud of note should be played, the pitch that tells the MIDI device how long or short of note will be. There are many programs that can read the message in a MIDI file and translate it for MIDI device what instrument to play as you can see in Figure 2.2

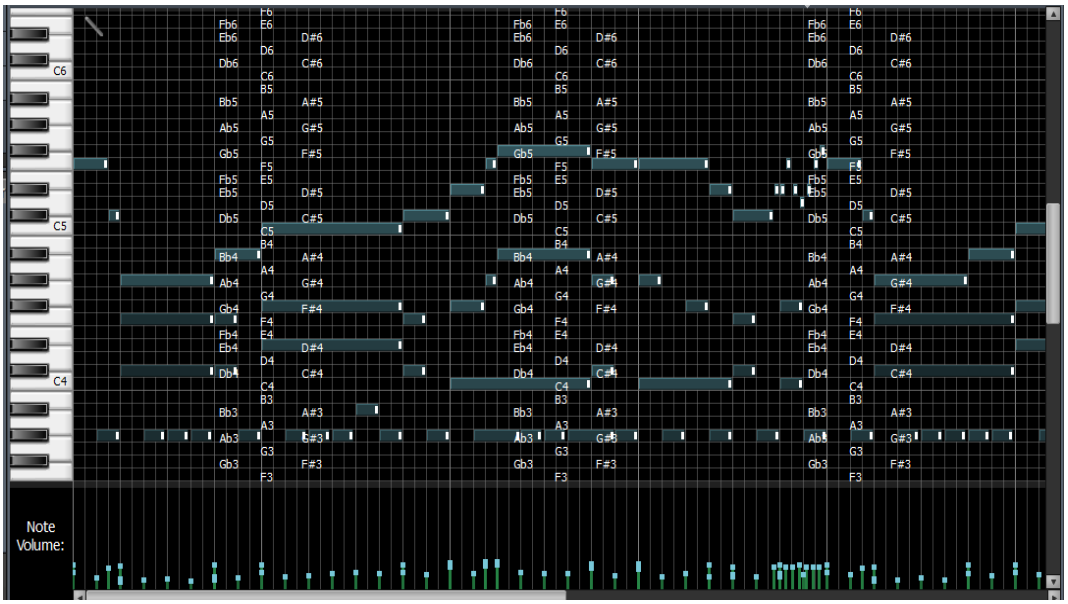

Figure 2.2: MIDI interface.

2.2.2 The physical transports for MIDI

There are many connector & cables that used to transport MIDI data between devices (USB, Bluetooth, FireWire, etc.).

2.2.3 File Formats of MIDI

MIDI is the made up of the standard MIDI file(*.mid). All of the computer platforms can play a MIDI file. And anyone can make MIDI file by using free software(Garageband, FL Studio, etc.).

2.3 Music Moods Classification

Music moods classification is a method to categorize music on the feeling generated by a song. In the most existing method, the moods of a song are divided according to psychologist Robert Thayer's traditional model of mood as seen in Figure 2.3.

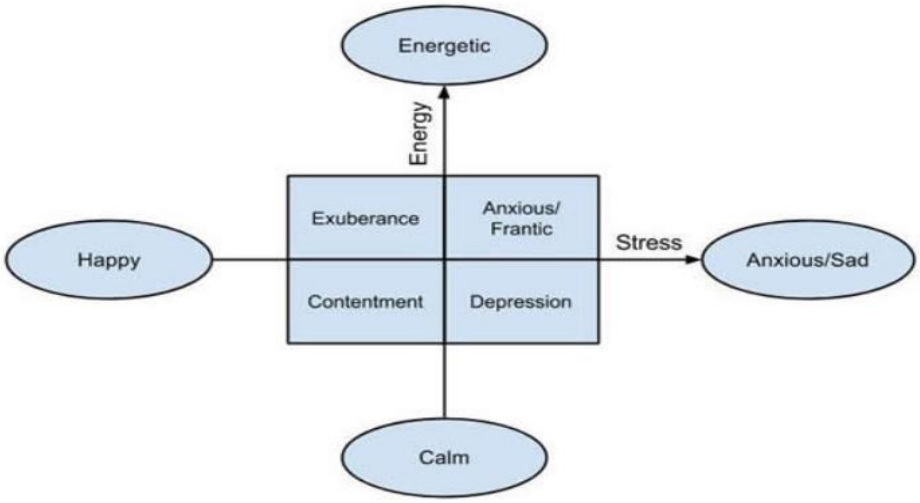

Figure 2.3: Thayer's mood model [2].

The model divided moods along with two lines, energy and stress. In the stress line from happy to sad and intersect with the energy line from calm to energetic. When breaking a song down into a music component such as rhythm, pitch, intensity, and timbre can allow matching the song with a mood represent from Robert Thayer's model.

The rhythm of the song is associated with the energy line, the faster the rhythm of the higher energy, and the slower rhythm the lower of the energy. The intensity or the volume also associated with energy, high intensity means a song that feels angry while low intensity suggests a softer song. High overall pitch can be indicated as a happiness and light mood songs, while lower pitch implied more sad and serious tone. Timbre or the tone quality is associated with the harmonic of a song. A song that has a simple harmonic can be indicated as a "darker" timbre and tend to soothe human emotions. From the music component and moods can be created into the correlation as seen in the table below.

Table 2.1: Moods classified according to musical components [2].

Mood	Intensity	Timbre	Pitch	Rhythm
Happy	Medium	Medium	Very High	Very High
Exuberant	High	Medium	High	High
Energetic	Very High	Medium	Medium	High
Frantic	High	Very High	Low	Very High
Anxious/Sad	Medium	Very Low	Very Low	Low
Depression	Low	Low	Low	Low
Calm	Very Low	Very Low	Medium	Very Low
Contentment	Low	Low	High	Low

From Table 2.1 shows all eight of Thayer's mood classifications with the different musical components ranking from very low to very high. Moods like happy, exuberant and energetic all have a higher amount of intensity, rhythm and pitch than moods like calm, sad and depressed.

Table 2.2: Mean value for each mood based on music components [2].

Mood of Western songs	Mean Intensity	Mean Timbre	Mean Pitch	Mean Rhythm
Happy	0.2055	0.4418	967.47	209.01
Exuberent	0.317	0.4265	611.94	177.7
Energetic	0.4564	0.319	381.65	163.14
Frantic	0.2827	0.6376	239.78	189.03
Sad	0.2245	0.1572	95.654	137.23
Depression	0.1177	0.228	212.65	122.65
Calm	0.0658	0.1049	383.49	72.23
Contentment	0.1482	0.2114	756.65	101.73

Table 2.2 shows the numeric value from the qualitative value in table 2.1. Each value is a mean value of each mood based on music components. The value gathers from the experiment done by the research team. Pitch in the table is an amount of frequency in Hz. Rhythm in the table is given as the number of beat per minute.

2.4 Variational Autoencoder (VAE)

To understand the VAE first, we need to understand the standard autoencoder. An autoencoder is an unsupervised machine learning algorithm. Autoencoder consists of two network-connected together, one is the encoder network, the other is the decoder network. The encoder is a network that takes an input then produce the representation of the input that contains enough information to differentiate it but with a smaller size. Then by using the decoder network, it will reconstruct the original input using the representation by the encoder as seen in figure 2.4 below.

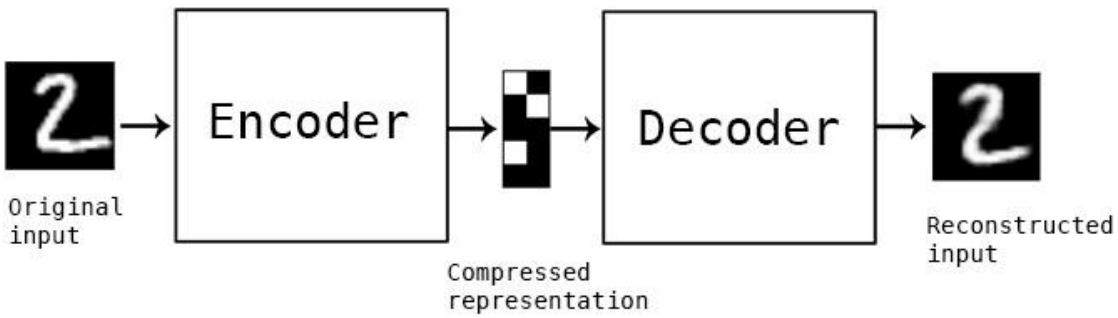

Figure 2.4: Standard Autoencoder[23]

The variational autoencoder much likes the standard one, but there is one fundamentally different in the latent space of VAEs. The latent space of VAE is a probability distribution, unlike the standard autoencoder as seen in Figure 2.5. The left picture is the result of normal encoding by the standard autoencoder. The right picture is the result of VAE encoding.

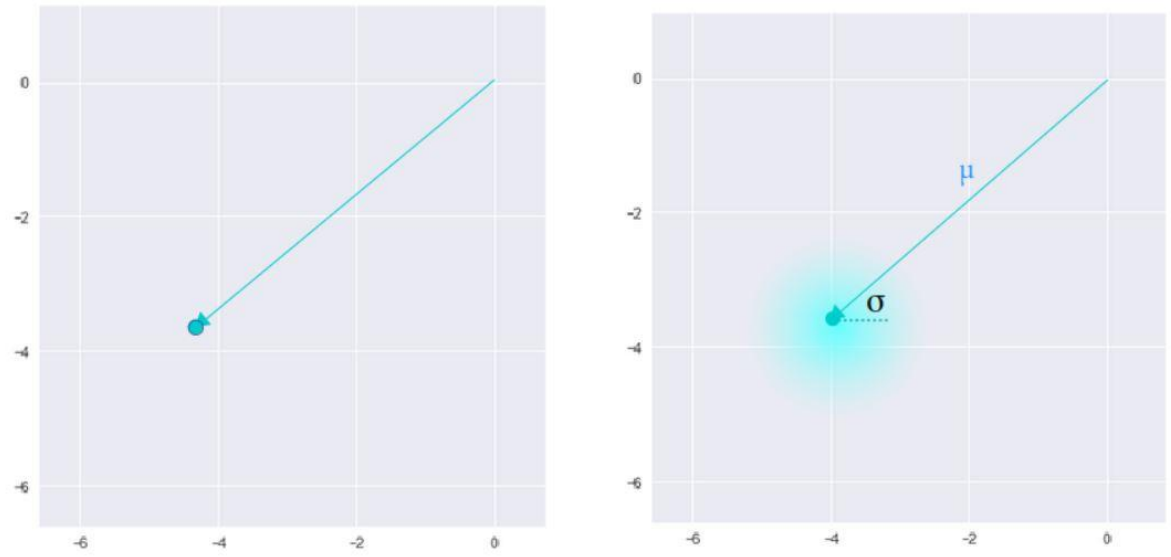

Figure 2.5: Different between autoencoder and VAE [21].

The VAE achieve this by using the mean vector (μ), and the standard variation vector (σ). The mean vector is used to determine where the input of the encoder is center. The standard variation is used to control the area of which the encoding can be varied. Thus the encoding can be generated randomly from any point within the circle. Because of this property, it allows the VAE to decode more vary range of input.

2.5 K-mean clustering

K-mean clustering is one of the unsupervised machine learning algorithms. K-mean is used to group the set of unlabeled data together, with the number of the group represented by the number of the cluster or K. The amount of cluster or K is defined by a user, while there is no method to determine the exact value of K but there is a way to help to choose the K. Each data will be put to a group which is similar. In each group will also consist of one centroid, which is represented of the center of the cluster. The way to determined which cluster to assign data is by finding the nearest cluster.

2.6 Recurrent Neural Network (RNN)

RNN is a class of artificial neural network but the difference is there is a connection between each state to form a memory called “Internal memory”. Unlike the feedforward neural network, RNN can use the internal memory to process the input that is sequential data.

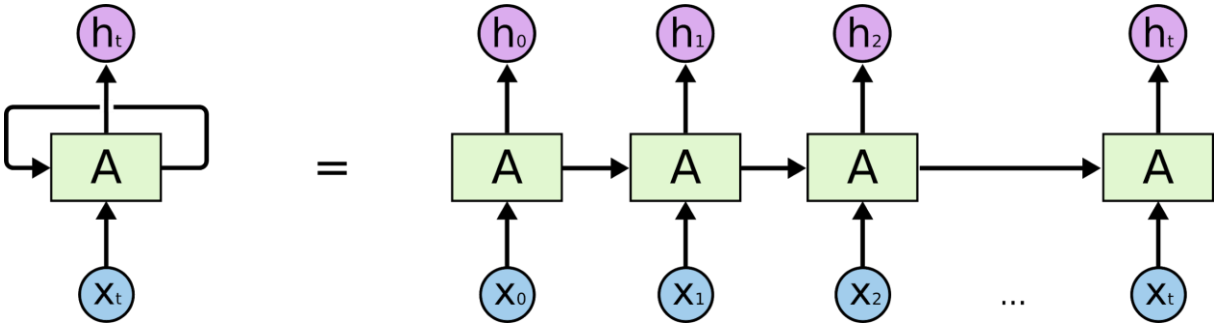

Figure 2.6: An example of RNN [7].

From Figure 2.6 the method that RNN use to create the internal memory is to create a loop within it. From figure A is activation, get the input Xt in the state of time, and output value to the hidden layer. The loop allows the information to be pass into the next state, therefore the first input will affect the output in the last state.

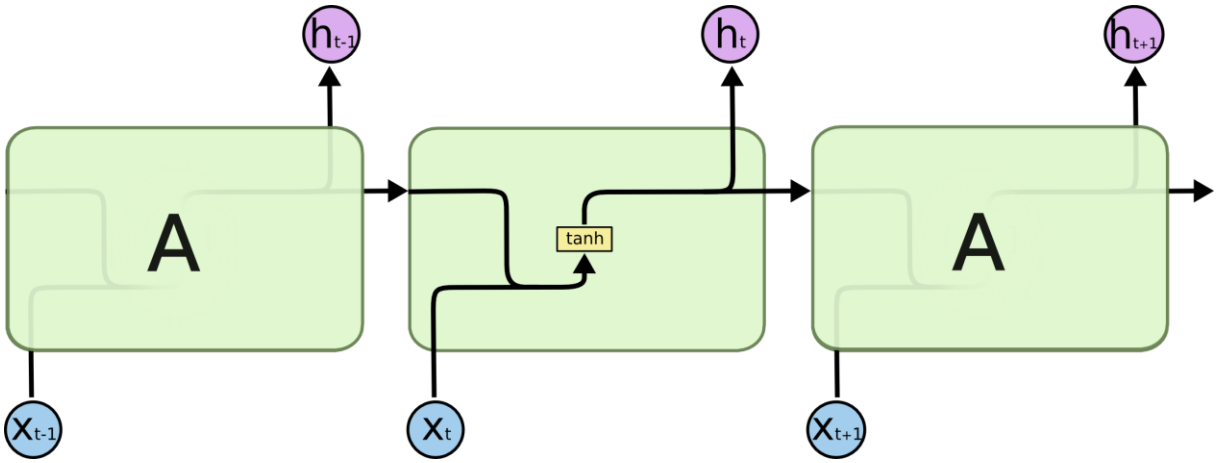

Figure 2.7: A standard RNN method [7].

Inside the RNN layer, the methods are different for each type as you can see from Figure 2.7 it is the standard RNN method. It takes the output value from the previous state and then combines with the input of the present state. After combining the result will go to the non-linear function, in this case, is Tanh. Then the output from this state will be used in the later state. The figure above can be transformed into formula below.

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

h(t) is a hidden layer or output value of this state.

ϕ is an activation function either sigmoid function or tanh.

W is the weight of the input.

x(t) is a present state input.

U is a hidden state matrix.

h(t-1) is a hidden layer or output value of the last state.

Because of this method, many consider RNN to use whenever they need the context from the past such as speech recognition, video analysis, and music composition.

2.6.1 Long Short-Term Memory(LSTM) Network

Similar to RNN the Long Short-Term Memory network is also a recurrent neural network but with different architecture design as seen in Figure 2.8. LSTM is specifically used to avoid the long-term dependency problem or the vanishing gradient.

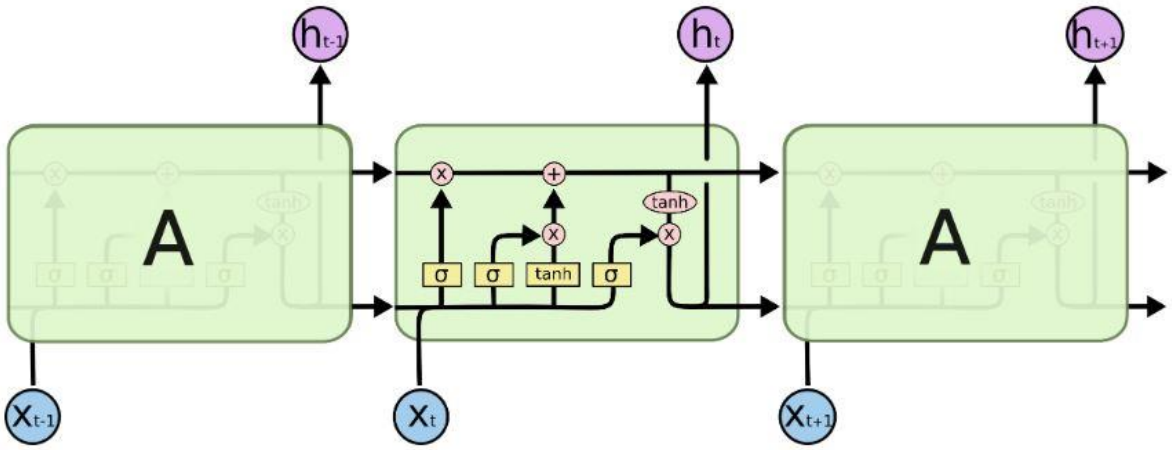

Figure 2.8: LSTM architecture design [7].

The core idea of LSTM is the cell state, the horizontal line that goes between each state. The cell state is like a bridge to carry out information from the first state throughout the model. Besides the ability to carry information, LSTM also able to manipulate with the cell state by adding gates that interact with the cell state. Each gate can alter the information on cell state like forgetting some of the data or to determine which information need to remember.

2.7 Generative Model

The generative model is one of an unsupervised learning model in order to learn the training data and try to generate a new one from the same distribution. The generative models aim is to learn the data distribution in the training set in order to generate the data with some variation. Therefore a good model can give similar data distribution with the real data distribution. The sample of the generative model can be seen in figure 2.9.

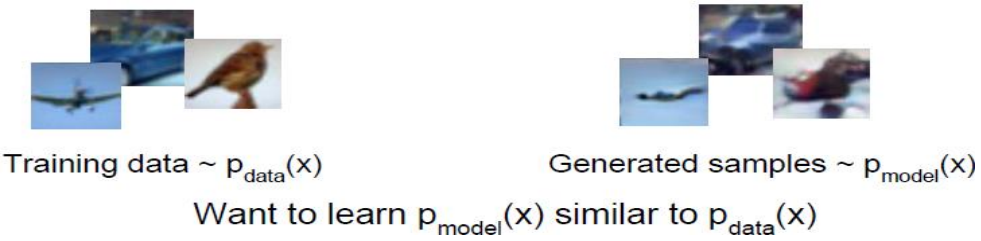

Figure 2.9: The sample of the generative model result [31].

2.8 Perplexity

Perplexity is the measurement of the model probability in term of the distribution between each class in the sample. The formula of the perplexity of a probability distribute is shown below.

$$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)} \tag{1.2}$$

H(p) is the entropy of the distribution p(x)
x is a random variable over all possible events.

Thus the perplexity value is, in fact, the exponential of the entropy. Therefore the perplexity value can be defined as “the weighted average number of choices a random variable has” [28]. Then, the lower the perplexity the better the quality of the generative model.

2.9 Model Development tools

2.9.1 Keras

Keras is an open source neural network library written in python. It uses Tensorflow as one of the backends using both CPU and GPU. It's designed for a fast experiment with Deep Neural Network. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Keras library commonly uses in building neural networks such as network layers, activation function, optimizers and etc.

2.9.2 TensorFlow

Tensorflow is an open source library for numerical computation and large-scale machine learning. It uses Python to provide a convenient front-end API for building applications with the framework while executing those applications in C++. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks and being the backend for other libraries.

2.9.3 Music21

Music21 is a Python-based toolkit for computer-aided musicology and tinkering with MIDI file. It can be used for both encoding and decoding MIDI file, edit music notation, generate music examples, and etc.

2.10 Web Application

2.10.1 Flask

Flask is a microframework for Python based on Werkzeug, Jinja 2. It is free and open source since 2010. It is suitable for developing the Restful API because it comes with excellent documentation and features including lightweight server, good templating and security feature.

2.10.2 Bootstrap

Bootstrap is a powerful front-end framework that worked faster and easier for a web developer. It includes HTML and CSS based on design templates for the user interface. Bootstrap is free and opens sourced framework since 2010 and it was known as Twitter Blueprint. It can create flexible and responsive web layouts with fewer efforts

2.10.3 Vue

Vue is a framework used for creating the user interface. The core library of Vue is focused on the view layer while it’s able to integrate with other libraries or the existing projects.

2.11 Literature Review

2.11.1 Using Machine Learning to Create New Melodies

This program created by Brannon Dorsey teams. They used an idea of machine learning to create an Artificial Intelligence program to generate a melody by previous melodies. The previous melodies that they used are video game called Final Fantasy as their dataset. In this program, they used midi-RNN that is a tool that allows a machine to generate a melody with machine learning by using a basic of RNN and LSTM. Those models are designed to work with sequential data. In this program, a user can view the model training error in real time, so if user see a result of training model and get worse(this is called overfitting) the user can stop the training process

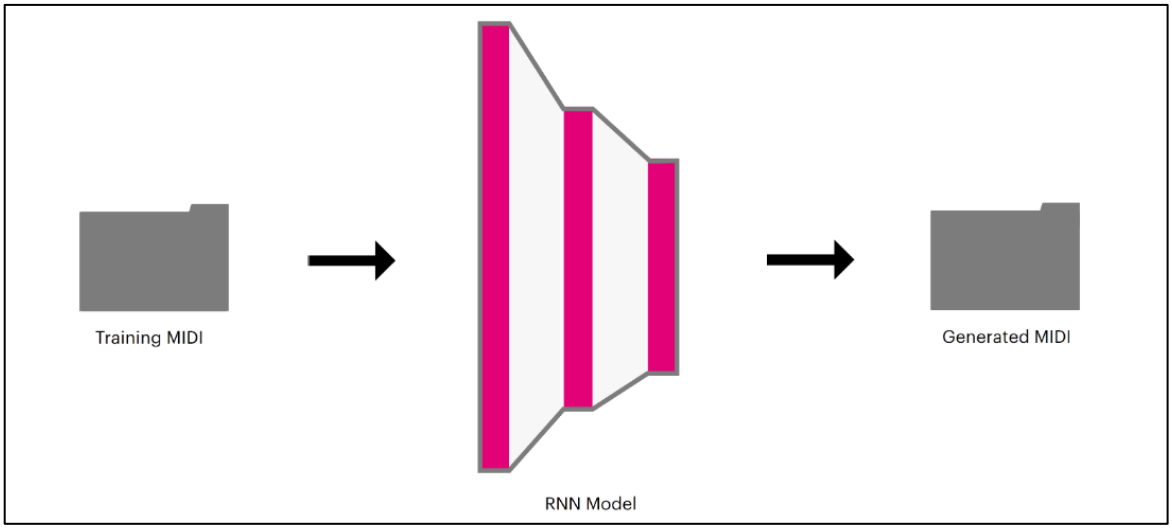

Figure 2.10: midi-RNN Training Process [3]

From Figure 2.10, it is a training process. The machine learning will train an RNN model by using MIDI file in a training MIDI folder, then uses a model to generate a new MIDI file of melody.

2.11.2 Jukedeck

Jukedeck is a startup that uses Artificial Intelligence to compose music. It uses neural networks as a deep learning method. The development team used a big set of musical data to analyze, then the neural network learns to write original music. Jukedeck's compositions cost $1 for individuals or small businesses but are free as long as the company is credited. There is a web application for people to register and play with the model to generate music. In the web application after login user can choose a music genre, instrument, music mood, duration, climax and Beat Per Minute as seen in Figure 2.11.

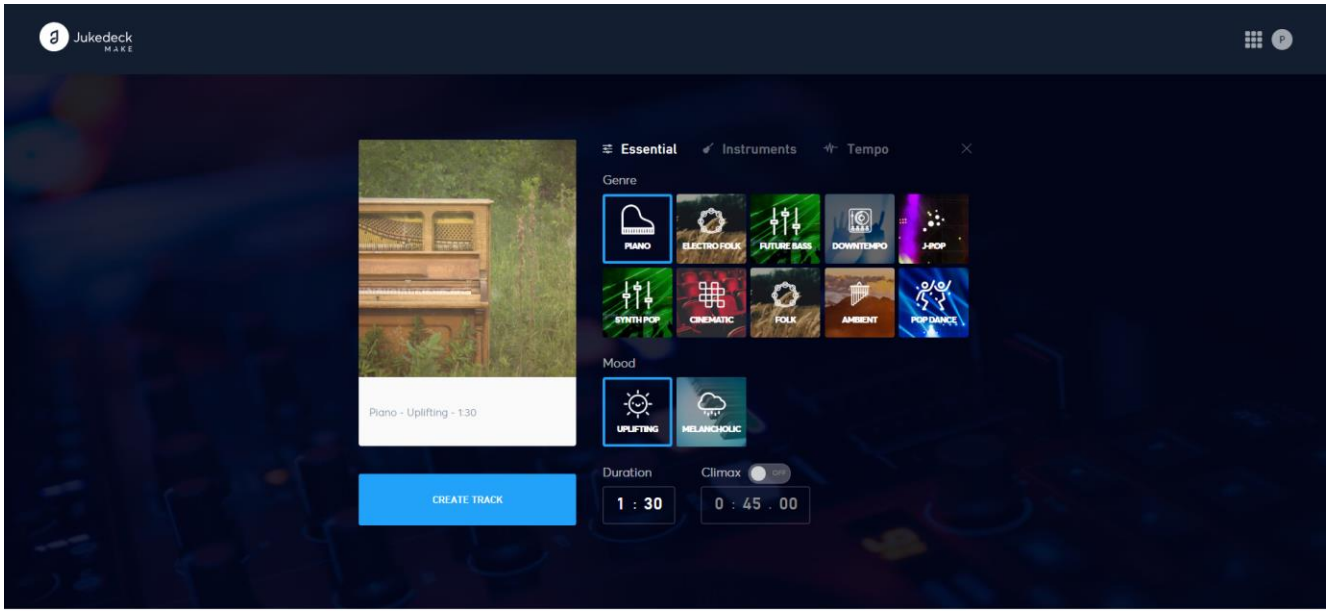

Figure 2.11: Jukedeck Create track user interface [19].

After the model complete generating the music, it will display on the web application in the individual user library. The user can download the complete music, mark as a favorite and edit a song within the library.

Chapter 3
Design and Methodology

3.1 Architecture Diagram

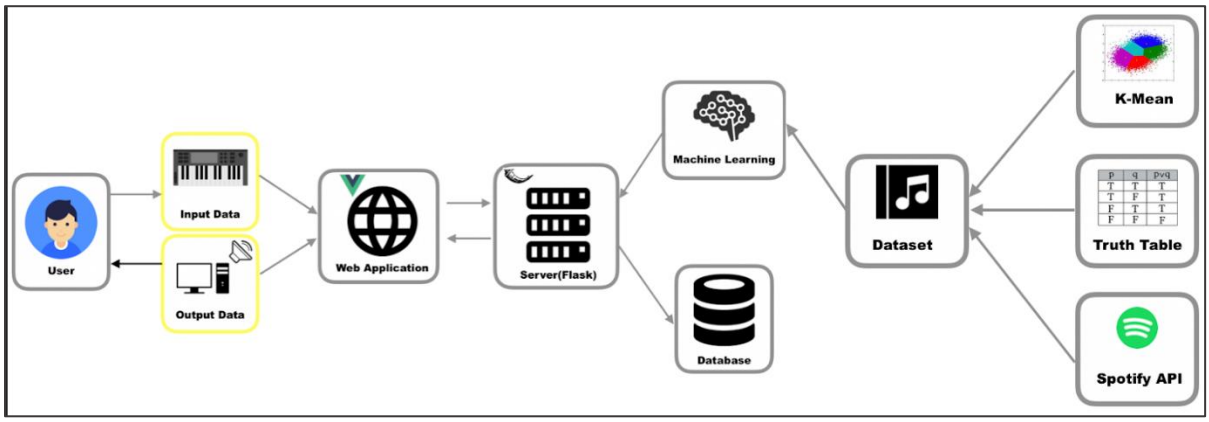

Figure 3.1: Architecture diagram.

From Figure 3.1. we can separate into 4 major part; input, process, dataset, and output.

Input:

- User select moods of the song (Happy, Sad, Calm) on the web application.
- User select duration of the song between 10 sec - 15 min on the web application.
- User plays the input note by using the piano or keyboard to be the input for the web application.
- User sends the feedback for the song on the web application.

Process:

- A server receives notes, mood, and duration that the user selected from the web application.
- A server generates a song by using a machine learning model according to the mood that the user selected.
- A server returns a song generated by model back to the web application.
- A server receives feedback from a user via the web application.
- A server sends the feedback from the user to the database.

Dataset:

- The dataset of happy, sad, and relax model are separated using k-mean, truth table, and Spotify API methods.

Output:

- Display a Song based on mood and duration that the user selected via the web application.
- Allow a user to play, pause and download a generated song via the web application.

3.2 Use case diagram and Use case narratives.

3.2.1 Use case diagram

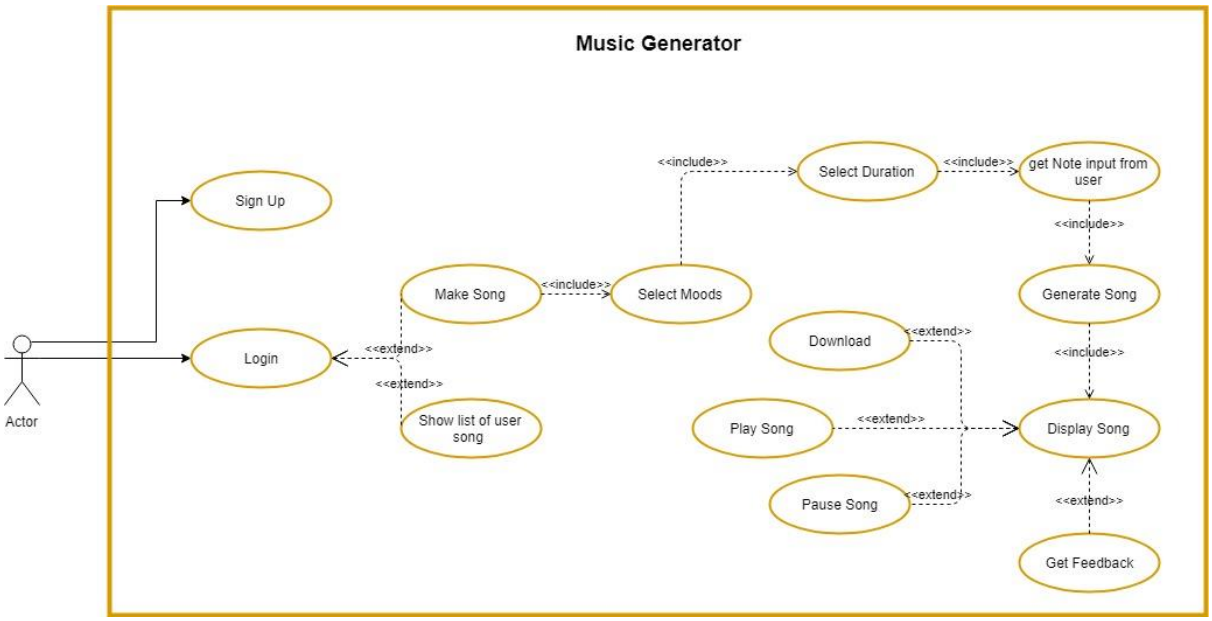

Figure 3.2: Use case diagram.

Figure 3.2 shows the use case diagram of the program. Before the user can make the song, a user needs to log in first. However, if the user doesn't have an account then they need to register beforehand. After that, the user can choose 2 option; Make a song or Show list of user previous songs. When user select make a song, the user needs to select the mood, duration and playing some note as the input for the program to generate the song, then the program will generate the song and will display it for the user. The user then can play a displayed song or pause it, also he/she will be able to download the song as well. After that user can send feedback of a generated song.

3.2.2 Use case narratives

Title: Generate song.
Actor: User
Post-Condition: User get a song
Main success scenario:

1. User select mood, duration of a song
2. User press the record button to play some note to be the input for the program to generate the song
3. User press the render button
4. The system generates a song
5. The system displays the song on the user interface

Alternate success scenario:

1A User select duration of more than 15 minutes.
2A Ask user for the new duration.
3A Return to main success scenario step 2.
2B User didn't select a mood or duration.
3B Ask the user to select mood and duration.

4B Return to main success scenario step 2.
3C User didn't record the notes for the input song
4C Ask the user to record the notes for the input song
5C Return to main success scenario step 2.

Title: Login
Actor: User
Post-Condition: Login
Main success scenario:

1. User fills username and password.
2. User login successfully.
3. system show user library song page.

Alternate success scenario:

1A User fill invalid username or password
2A Ask user for the correct username or password
3A Return to main success scenario step 2.

Title: Send feedback.
Actor: User
Pre-Condition: User-generated a **s**ong.
Main success scenario:

1. The user selects the feedback of a song.
2. The system receives feedback from the user.
3. The system displays the notification on the user interface.

3.3 Database schematics and data dictionary

3.3.1 ER-Diagram

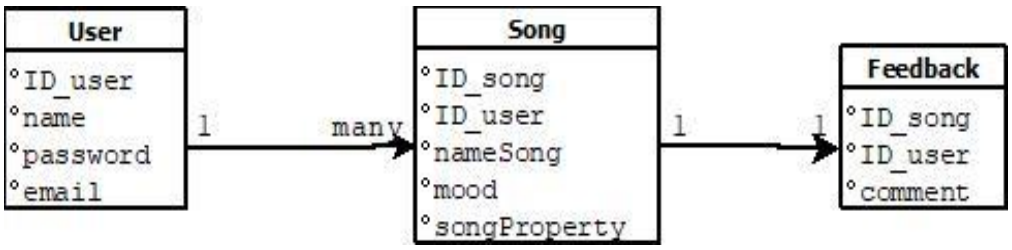

Figure 3.3: ER diagram of the web application.

3.3.2 Data Dictionary

Table 3.1: User table data dictionary.

Field Name	Data Type	Description	Example
ID_user	Integer	Unique number ID for all user	1111
Name	Text	Username for all user	iamgide
password	Text	Password for all user	T12345
email	Text	Email for all user	aaa@mail.com

Table 3.2: Song table data dictionary.

Field Name	Data Type	Description	Example
ID_song	Integer	Unique number ID for all song	1111
ID_user	Integer	Unique number ID for all user	1111
nameSong	Text	Song's name for all song	Leave me
mood	Text	Mood of song for all song	Happy
songProperty	Array	Song property for all song	[A4,0.0,0.0,3.2]

Table 3.3: Feedback table data dictionary.

Field Name	Data Type	Description	Example
ID_song	Integer	Unique number ID for all song	1111
ID_user	Integer	Unique number ID for all user	1111
Comment	Text	Password for all user	Not bad at all

3.4 Protocol Design

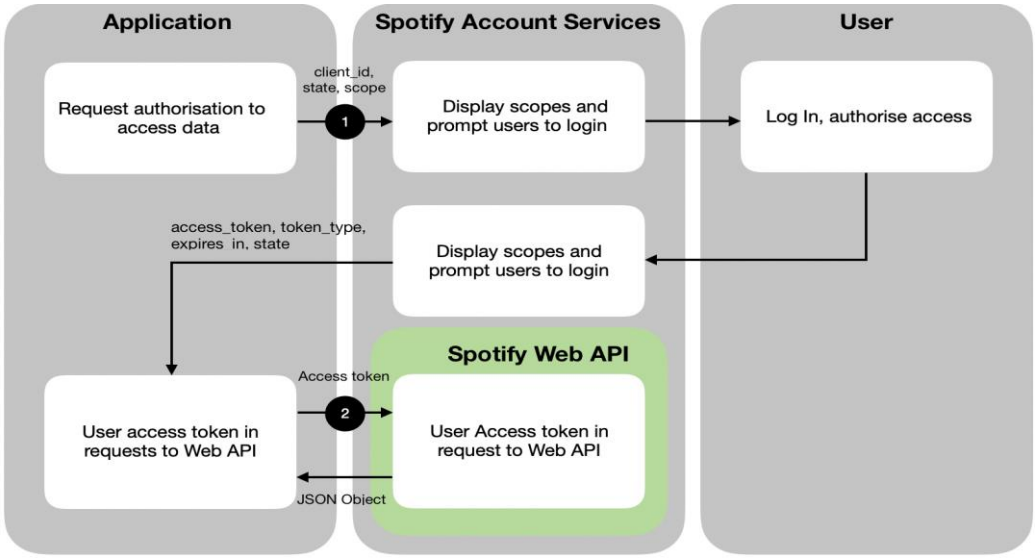

Figure 3.4: Protocol Spotify API Diagram.

From figure 3.4, shows a diagram of how to get data from Spotify web API. This diagram consists of 3 parts; Application, Spotify account services, and User. At first, a user needs to send the request for getting a token from Spotify account services. After the user successfully login, Spotify account services will send a token to an application. Then the application will be able to use the access token to request data from the Spotify.

3.5 User Interface design

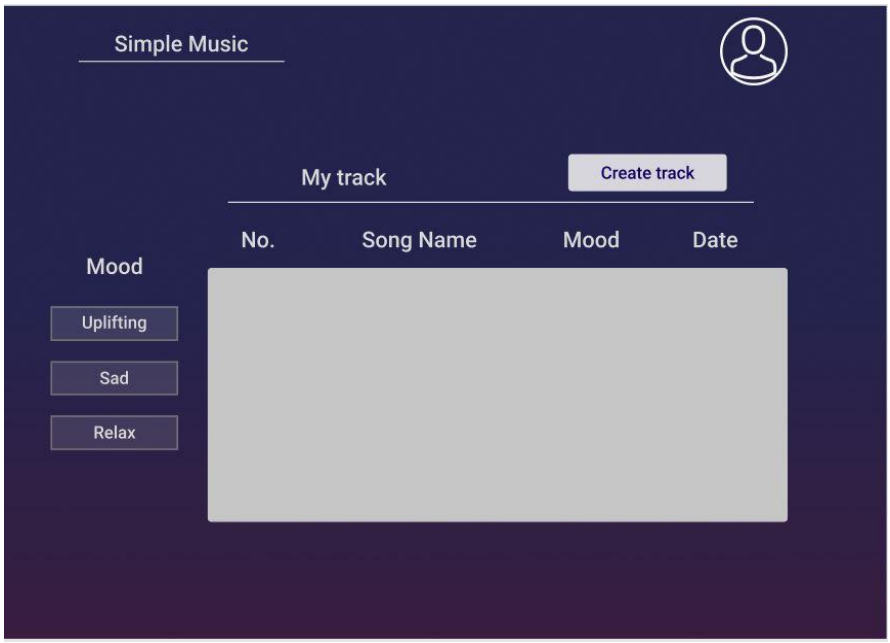

Figure 3.5: User interface design of the user library page.

From figure 3.5 show the user interface page for the library page of the user. This page will contain the previous songs that the user used to generate. The song will contain information about the song name, mood, and generating date. This is the first page from when user login. The user is then able to select create track button in order to go to the generate song page.

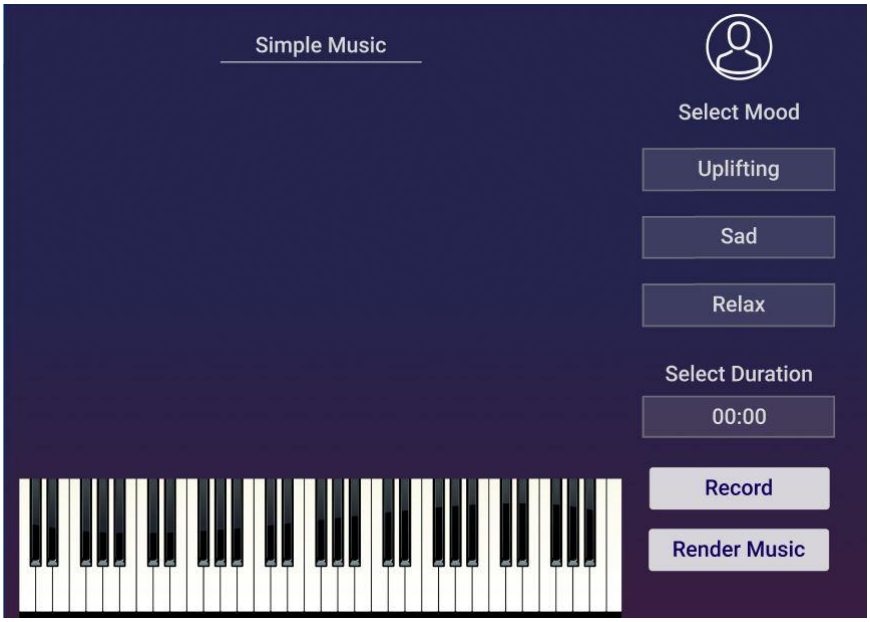

Figure 3.6: User interface design of the generate song page.

From figure 3.6 show the user interfaces for the generate song page of the web application. The user can select moods by clicking the button with the mood name on it. In the duration box, the user can select a number of duration by typing in it. Then the user can select the record button to start recording the note input by playing on the piano. After the user selects mood, duration, and input some note then the user can select render music button which will lead to the song page.

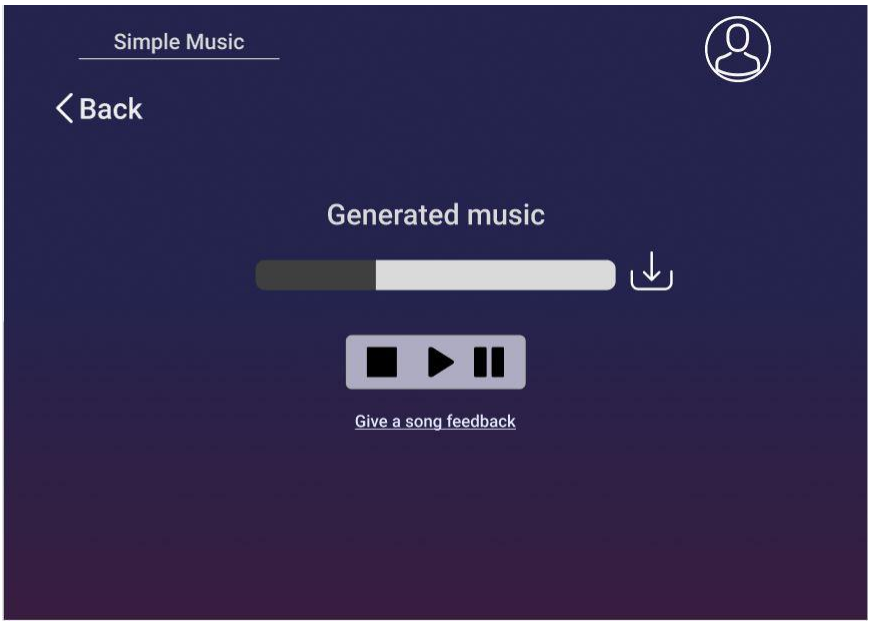

Figure 3.7: User interface design of the song page.

From figure 3.7 show the second page of the web application which is to display the music from the model. In this page, the user can play, pause or stop the music with the button display in the middle. The user can download the song by click on the download button display on the right of the music bar. Below the song control panel will be the link to the feedback page.

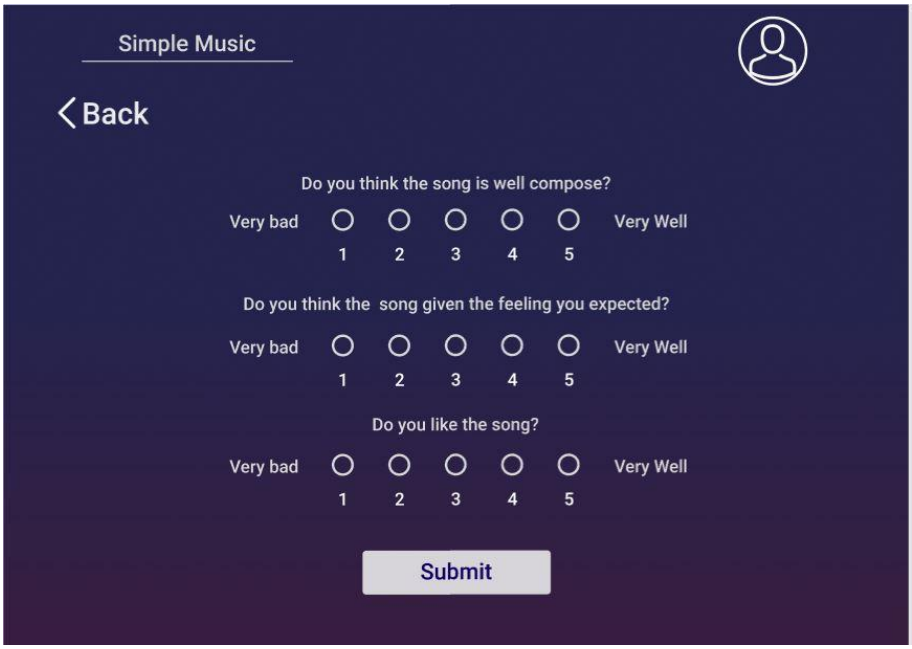

Figure 3.8: User interface design for feedback page.

From figure 3.8 show the user interface of the getting feedback page. In this page will display three questions correspond to the song that the model generated. The first question is to ask the quality of the song. The second question is to ask for the mood of the song. The last one is to ask if the user like or enjoy the song. All of the three questions will have the radio button below for the user to choose the score. At the bottom of the page will be the submit button for the user to finish and submit the feedback.

3.6 Detail of project design

3.6.1 Mood classification

The objective for mood classification is to classify the mood in our dataset before using the data to train the generating model. In order to train the model to generate songs based on mood, we need to specify the mood of each song first before we can properly use it. Therefore we try to classify the data using these methods to make it more reliable than using only human ears.

Although we might be able to find the data that already been classified, however, it is difficult to find data with the timeshift and velocity of the actual human playing. We want our model to be able to generate the song that is melodic and human-like. But most of the song that records from human playing don't contain the mood. Therefore we want to create the mood classification method to classify these data

3.6.1.1 Mood Classification using the ground truth table

```
for each midi file in dataset
    if file has instrument part
        add instrument
    else
        add element
```

Figure 3.9: Pseudocode for reading midi files.

The first part is to loads each midi file from the containing folder. In a midi file, it contains a data instrument, tempo, key, notes, and chord, etc. As you can see in Figure 3.9 before gathering the element such as notes, chords, we need to specify the instrument first. After knowing the instrument then we can collect other elements into the object.

```
for each element in midi file
    if element is first tempo
        get tempo
    else
        break
    if element is note
        get pitch
    if element is chord
        for each note in chord
            get pitch
```

Figure 3.10: Pseudocode for collecting a tempo and pitches.

After getting all an element from a midi file, the next part is to get the value from each element as seen in Figure 3.10. The first step is to get the tempo or rhythm contain in the element. Another step is to check if the tempo is the first or not. In each song, there might be more than one tempo and it might make the classification difficult, therefore if it detects the second tempo it will stop the process for this song. Next step is to get the pitch from notes and chords. However, when getting pitches from chord we need to specify each note in the chord and then get the pitch of each note instead.

```
if pitch not equal to zero
    calculate pitch mean
```

Figure 3.11: calculate the mean value of the pitch.

After gathering all pitches and rhythm, we have to check if the pitch is equal to zero as seen in Figure 3.11. Sometime midi file can contain an instrument with no notes or chords whatsoever. Therefore checking if the pitch isn't equal to zero is to avoid error occurred when calculating for the mean value of the pitch.

3.6.1.2 Mood Classification using VAE and K-Mean Clustering

The concept of this method is to use VAE reduces the number of a variable in music and use those feature to clustering a group of moods in music. We want to cluster music into a group of mood using some of their dominant features for clustering. However such features are notes, velocity, tempo, and etc. which can be overwhelming. Therefore In order to cluster a mood, we need to reduce some of their features before clustering. Thus using the VAE would reduce those features into smaller representation.

3.6.1.2.1 Prepare Data

The method that used for prepare data is very similar to the first part of mood classification using the ground truth table. But in this method, we will not only collect a song note and chord. We will also collect the song name, the tempo of each note/chord, and the velocity of each note/chord as seen in figure 3.12. We need to loop through each element in order to collect all the information we need and store it in a list.

```
for each element in midi file
    if(element is tempo)
        get tempo
    if(element is note/chord)
        get note/chord
    if(element has velocity)
        get velocity
```

Figure 3.12: Pseudocode for getting note/chord, tempo, and velocity.

After collecting all note, we need to set each note with the specific number in order to use this data in the machine learning model. Therefore to change a note to a number, we will use the dictionary function that will set each note with specific values as seen in Table 3.4. Thus all of the note will be able to represent as an integer.

Table 3.4: Example of Note dictionary.

Key	value
A0	0
A1	1
B0	2
B1	3
B#2	4
C1	5

The next step is to normalize every element we gather, so the model will be able to process the data better. Therefore we need to normalize the note, tempo of each note, and the velocity of each note into a number between 0 and 1.

```
for note in eachsong
    set note = note / total number of notes
```

Figure 3.13: Pseudocode for Normalize Note.

After we assign every note into an integer using the dictionary function. we will normalize each note into a number between 0 to 1 by divide note's integer by a total number of unique note and chord as seen in figure 3.13.

```
for velocity in each notes
    set velocity = velocity / 128
```

Figure 3.14: Pseudocode for Normalize velocity of each note.

The velocity of each note that we collect from a preparing data process is a number in range 0 to 127. From figure 3, we will normalize velocity by dividing it an integer 128 as seen in figure 3.14. Therefore all of the velocity of the notes will be between number 0 and 1.

```
set maximumTempo = 0
for note in each song
    for tempo from each notes
        if tempo is more than maximumTempo
            set maximumTempo = current tempo

for tempo from each notes
    set tempo = current tempo / maximumTempo
```

Figure 3.15: Pseudocode for Normalize tempo.

To normalize the tempo, it will be more complicated than the velocity and notes. From figure 3.15 we need to find the maximum tempo of all the songs first. After that, we will use the maximum tempo as the measurement. By dividing all the tempo with the max tempo, thus making all the tempo become a number between 0 and 1.

```
for note in eachSong
    for length of shortest song
        add note, velocity, tempo to data_input array
    append data_input into network_input list
```

Figure 3.16: Pseudocode for Determine network input.

After we completed normalizing a note, velocity, and tempo, we will determine a network input by connecting those three together. Next, we will sort by using the first note and append with the first a velocity and last append with the current tempo. Then we will append the second note, second velocity, and second tempo respectively and so on. We will determine the sequence length of the input be the length of the shortest song as seen in figure 3.16.

3.6.1.2.2 Model Design

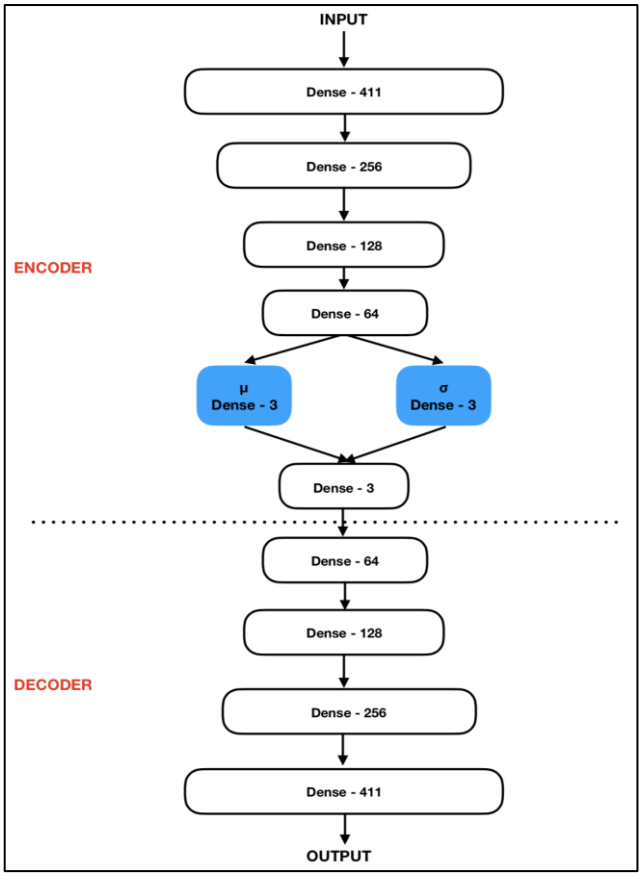

Figure 3.17: VAE Model Architecture.

From figure 3.17, it shows that the model consists of 2 parts; encoder and decoder. Part Encoder, it includes 5 fully-connected layers input layer contain 411 nodes and continuously decrease. Output layer of the encoder will contain only 3 nodes. In the decoder part, the layer in this part looks like a layer in the encoder part but the amount of node is continuously increased. The input and the output data from encoder and decoder parts are the same because we want to find the smaller representation which can represent the input data.

After we finish training the VAE model, we will use only the encoder model to predict the result. The output from the prediction will be 3 features as we design in the encoding part of the model. After that, we will use the 3 features as a criterion for the K-mean clustering. First, we will find the inertia to help us choose the number of clusters. Then we would be able to cluster the music into the same group of features or in this case the mood.

3.6.1.3 Classification Mood by using data from Spotify

The concept of this model is to use data from the Spotify to create a classification model to classify the mood of the song. We will use the feature from the track that we request from the Spotify web API like keys, Tempo, and Time signature. The feature

we use must be able to extract from the midi file as well. After acquiring the information we will label the song mood by using the keyword of the playlist such as, happy, sad, and relax. Then we will train the model and use it to classify our midi file.

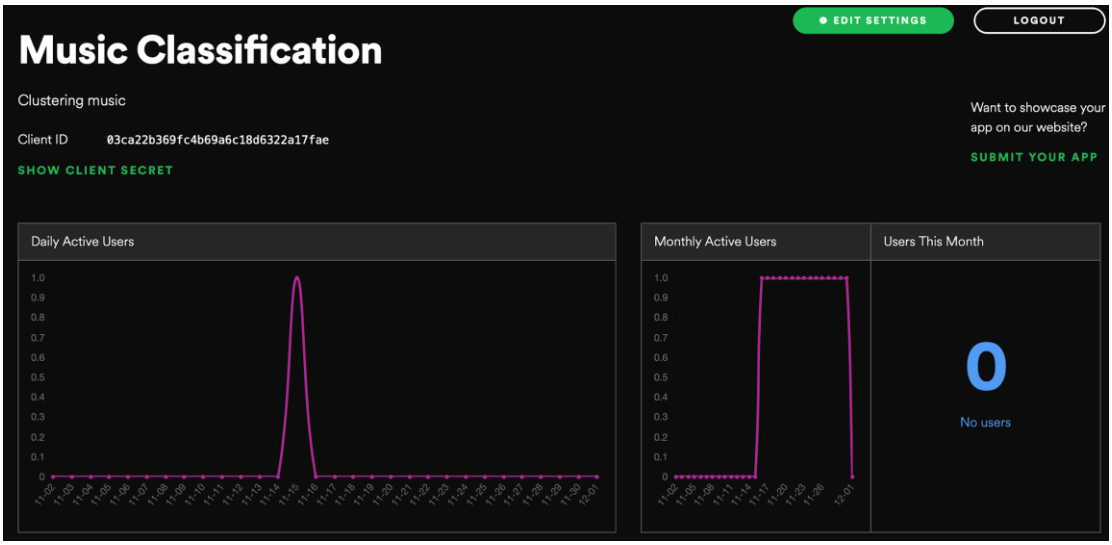

Figure 3.18: Spotify Web API [26].

The first step before we can get data from Spotify web API is to set the client ID in Spotify Web API as seen in figure 3.18. After that, we will use those client ID and client secret code for authenticating requests. Without the client ID, we won't be able to access or request any information from Spotify.


```
"track": {
  "duration": 255.34898,
  "sample_md5": "",
  "offset_seconds": 0,
  "window_seconds": 0,
  "analysis_sample_rate": 22050,
  "analysis_channels": 1,
  "end_of_fade_in": 0,
  "start_of_fade_out": 251.73333,
  "loudness": -11.84,
  "tempo": 98.002,
  "tempo_confidence": 0.423,
  "time_signature": 4,
  "time_signature_confidence": 1,
  "key": 5,
  "key_confidence": 0.36,
  "mode": 0,
  "mode_confidence": 0.414,
```


Figure 3.19: JSON data from Spotify web API [27].

After we set up the client ID we can use it to make a request to Spotify web API to get the data we need. For example, we search for a playlist with the keyword "Happy" and send the request to Spotify web API and it will return a JSON object as seen in figure 3.19. We will use data in JSON object; keys, tempo and time signature of those songs to train the model for classification the mood of a song.

3.6.2 Model generate songs

3.6.2.1 Pre-process

The first thing we need to do is to load the midi file, similar to the first part of mood classification using the ground truth table. We need to get the instrument, then we can get the element of that instrument such as note and chord.

```
create note/chord list
for each element in the midi file
    check if the current element is note
        add note into note/chord list
    check if the current element is chord
        add chord into note/chord list
```

Figure 3.20: Pseudocode for getting note and chord.

After-acquired the element from the last part, we need to loop through the element and gather notes and chords. From figure 3.20 we will loop through each element and check if it's a note or a chord. Then we will add both note and chord into one single list, in the order that it is played.

The next process is to assign each note with a specific value in order to use in one-hot encoding. To achieve this we will use the function call Dictionary in order to assign each note and chord with a numeric value. Therefore all the note and chord in the note list will be able to determine using the integer number similar to the part in mood classification using VAE and k-mean clustering.

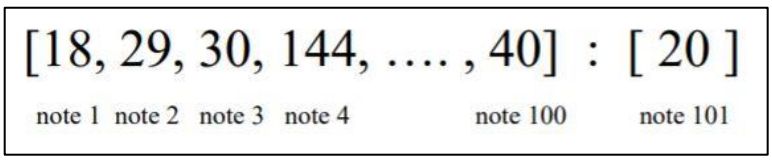

Figure 3.21: Labelling the input and output prediction.

To determine the input and output prediction we will label the note sequence to be the input and the next note to be the output prediction. The input will be the sequence of notes or chord from the list. The length of the sequence is 100 notes or chord. The output prediction will be the next note or chord that come after the particular sequence. Then we will move the sequence by one, for example, the first sequence starts from note 1-100 then the next sequence will be noted 2-101 as seen in figure 3.21. We then continue loop through this process until it's the end of the note list.

$$
\begin{aligned}
[1] &= [1, 0, 0, 0, \dots, 0] \\
[2] &= [0, 1, 0, 0, \dots, 0] \\
[3] &= [0, 0, 1, 0, \dots, 0] \\
&\vdots \qquad\qquad \vdots \\
[n] &= [0, 0, 0, 0, \dots, 1]
\end{aligned}
$$

Figure 3.22: Example of one-hot encoding.

After we get the output prediction we will need to make it into a vector of one-hot encoding as seen in figure 3.22. We need to use one-hot encoding because we will use “softmax” activation function and “categorical-cross entropy” as a loss function to classify the multiclass output. Beside one-hot encoding, we then divide each value of note with the amount of all notes and chords in order to make the value between 0-1. Thus making the input data become normalize.

3.6.2.2 Model design

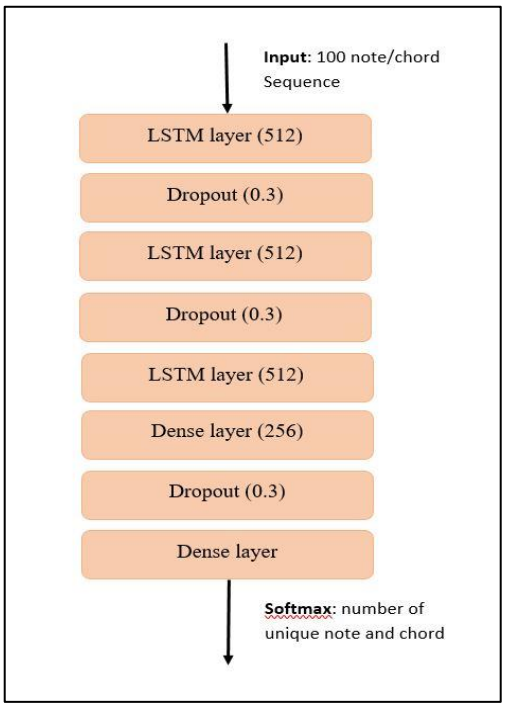

Figure 3.23: Model architecture.

From figure 3.23 the model consists of three LSTM layers and two fully connected layers with the last layer contain the amount of node equal to the number of unique note and chord. The number of unique note and chord is gathered when using

the Dict function and assign each unique note and chord value. The output of the model will be the prediction of each note and chord class due to the softmax activation function. The dropout layers help to reduce the overfit of the model and in this case, is to make the model not copy the dataset song completely.

3.6.2.3 Generating a song

When we want to generate a song using the model, we need to loop through the model and get each note prediction as seen in figure 3.24. First, we need to select a random sequence of note with the length of 100 or as same as we train the model. Then when using the model to predict for the output we need to convert back to the note using the Dic function. Because of the output from the model is in a numeric value specified when we use the Dict function in the first place. After that, we then proceed to add the note prediction into the output list. Before complete the process, we need to add the prediction output into the input sequence as well and remove the first note from the sequence. The reason we do this so that the model will use the output as an input and continue to predict the rest of the song. When completing all the task, the model then loops back and continue the process to get the nest prediction until it gets all 500 notes or the number of note we want.

```
create the randomly sequence
create prediction_output list
for 500 notes prediction
    predict the output using the random sequence
    use the dict function to turn predition back to note/chord
    add predict note to prediction_output list
    add the predict note into the random sequence
    remove the first note in the random sequence
```

Figure 3.24: Pseudocode for generating a song.

After complete gather, all the output prediction next is to covert the output into the midi file. From figure 3.25 we need to set the offset to be 0 first. The offset is a specific value which tells when the note or chord is played. After that, we will loop through all the prediction output list. Then we need to determine if the prediction is chord or note because the process to add the note and chord is a little bit different. Then we will set the offset and instrument for the current note or chord and add it to the output list. Before we can loop back we need to add the offset value, so all the note won't start at the same duration. Last when complete the loop, we will use the output list and use the music21 function to convert it back to the midi file.

```
set offset_count to 0
create output_note list
for number of prediction output
    if the prediction is chord
        set the offset of current chord to offset_count
        set the current chord instrument to piano
        add chord into the output_note list
    else is note
        set the offset of current note to offset_count
        set the current note instrument to piano
        add note into the output_note list
    offset_count + 0.5
convert to midi file using output_note list
```

Figure 3.25: Pseudocode for convert the output to the midi file

3.6.3 Model generate songs with velocity

The concept of the model is the same as the last one but adding the velocity into each note or chord. Thus adding the velocity into the generating should make the outcome become more varied since the same note can sound different based on its velocity. In order to achieve this, we then need two sets of sequence, one is note/chord sequence, while the other is its velocity of note/chord respectively.

3.6.3.1 Pre-processing

The concept of the pre-processing data does not differ from the last model. After loading, a midi file and read it the difference is when we add note/chod into the list we need to get its velocity too as seen in figure 3.26. We will add velocity into the different list, so after complete the loop, we will get two lists with the same amount of data.

```
create note/chord list
create velocity list
for each element in the midi file
    check if the current element is note
        add note into note/chord list
        add note velocity to velocity list
    check if the current element is chord
        add chord into note/chord list
        add chord velocity to velocity list
```

Figure 3.26: Pseudocode for getting a note, chord, and velocity.

The process to use Dictionary function for the note/chord is the same but the difference is when labeling the input and output. Because we have the list of note/chord and the list of velocity, therefore, we need to label two lists. But before label the velocity list we need to normalize the value first. By dividing each velocity with 128 because the velocity range that extracts from the midi file is 0-127. Also, we need to normalize the note/chord by dividing with the number of unique note and chord, as same as last model.

After complete labeling and normalize the data, we then need to stack the two input and output together. Therefore the input will be a sequence of note with its velocity, and the output will be a note with the velocity as seen in figure 3.27. All the value will be between 0-1 due to the normalization from the last step.

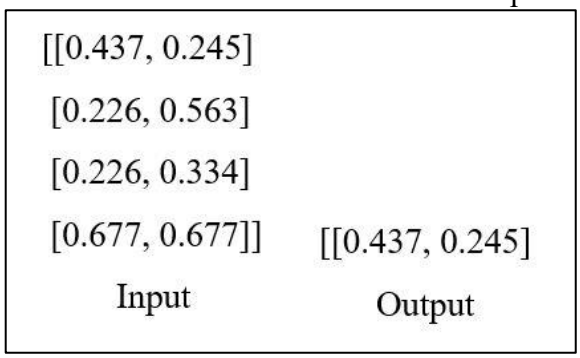

Figure 3.27: The input and output of the model.

Next, is to sort all the output and remove the duplicate data out in order to use Dictionary function to assign a numerical value to each one. Then when each output has specific value we will need to use one-hot encoding as well. Therefore the activation function and loss function of this model will be the same as the last one.

3.6.3.2 Model Design

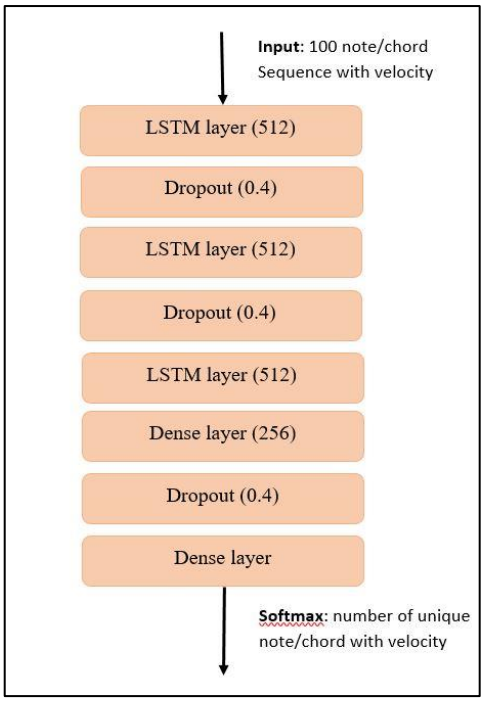

Figure 3.28: Model with velocity architecture.

There are two key differences in the velocity model and the standard model, one is the input, second is the dropout rate as seen in figure 3.28. The input of the model will be a two-dimensional sequence with a length of 100 notes. Therefore when using the model we need to specify the dimension too. The dropout rate is increased from 0.3 to 0.4 since we believe it should help reduce more overfitting the data songs.

3.6.3.3 Generating a song

The process of predicting an output note is the same as the last model, but the difference is after we get the prediction we need to separate the note and velocity. Another task is to converge the note and velocity back from normalization. Thus the note prediction will multiply with the number of unique note/chord and velocity will multiply with 128 as seen in figure 3.29.

```
create prediction_note list
create prediction_velocity list
    for all the prediction
        if it is note/chord
            multiply with number of unique note/chord
            add to prediction_note list
        if it is velocity
            multiply with 128
            add to predcition_velocity list
```

Figure 3.29: Pseudocode for separate note and velocity.

Last part is to converge the prediction from the model back to a midi file, but this time we need to specify the velocity too. The method of converging to the midi file is similar but we will specify the velocity of each note/chord by using the prediction output as seen in figure 3.30.

```
set offset_count to 0
create output_note list
for number of prediction output
    if the prediction is chord
        set the offset of current chord to offset_count
        set the current chord instrument to piano
        set the current chord velocity to the prediction_velocity
        add chord into the output_note list
    else is note
        set the offset of current note to offset_count
        set the current note instrument to piano
        set the current note velocity to the prediction_velocity
        add note into the output_note list
    offset_count + 0.5
convert to midi file using output_note list
```

Figure 3.30: Pseudocode for converge prediction with velocity to a midi file.

3.6.4 Model generate songs with timeshift and velocity

Building upon the concept of the model with velocity but leaning into the first method of the simple generating model and expand it by adding the time element and velocity into the account. The key difference is to create LSTM input into one sequence instead of two as in the velocity method. Another difference is in the pre-processing part of the data.

3.6.4.1 Pre-processing

The pre-processing part using the same concept as the other method and adding the duration and offset list. But the difference is we also add the indicator to the end of each song as seen in figure 3.31.

```
create note/chord, velocity, duration, offest list
for each song in dataset
    for each element in the midi file
        check if the current element is note
            add note into note/chord list
            add velocity into velocity list
            add duration into duration list
        check if the current element is chord
            add chord into note/chord list
            add velocity into velocity list
            add duration into duration list
        add offset into offset list
    add song ending indicator
```

Figure 3.31: Pseudocode for getting notes, velocity, duration, and offset.

After the end of each song, we will add the indicator to each list. Therefore when creating the training input using the same method as the other model, but in this time when going through the list and encounter the end, it'll skip to the next song. The objective for skipping is to avoid the sequence where two songs collide as seen in figure 3.32. While creating the sequence using the previous method might not be a problem, however, using the data of one song to predict the other seem to make less sense than using only within the same song.

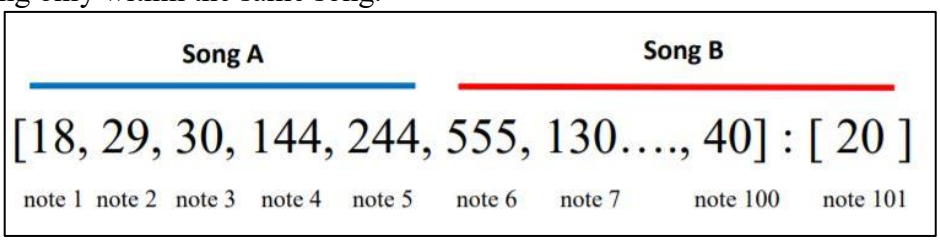

Figure 3.32: Input sequence for the original process.

Once we separate each song into sequences, we use the padding function with the start sequence of each song to create the N-gram sequence as seen in figure 3.33. We create N-gram sequence to train the model in order for the input from the user, in which it may not be the length that we train the model. Thus the model will be familiar with this type of input.

Input	Output
[0, 0, 0, ..., 0, 0, 45]	: [132]
[0, 0, 0, ..., 0, 45, 132]	: [20]
[0, 0, 0, ..., 45, 132, 20]	: [34]
[0, 0, 0, ..., 132, 20, 34]	: [50]

Figure 3.33: Example of N-gram sequence.

When using the Dictionary function we use all the factor into the calculation. For example, The note A5 with velocity, duration and offset of 120, 2, 1 respectively will be assigned to one key. while the note A5 with different velocity or duration or offset will be assigned to separate key as well. Thus making the input be only one sequence.

3.6.4.2 Model Design

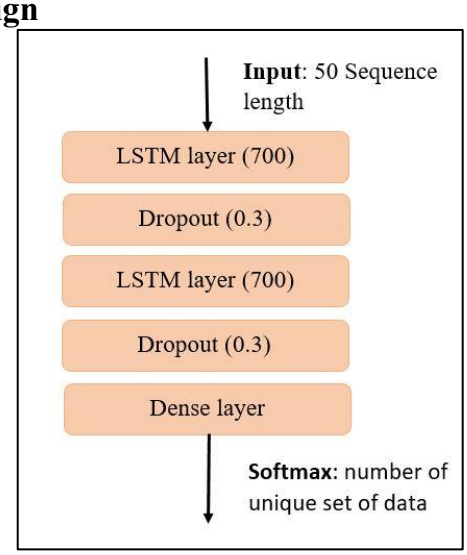

Figure 3.34: Model generate song with timeshift and velocity.

The model number of layers is less than other models due to the number of the class would be much more, so we decrease the number of layers so the model would be trained faster. However, the size of each layer is increased in order to compromise with the higher number of class as seen in figure 3.34.

3.6.4.3 Generating a song

The method of generating a song is not different than the others but adding the element of duration and offset. For the duration, it's the same as velocity, therefore when adding the prediction note or chord we also adding the duration of itself too. But for offset at the end of each step, we then add the prediction offset cumulatively. Thus the generating part is not much different than the other two models.

Chapter 4
Results and Discussion

4.1 Mood classification methods result

The result from the 3 methods shows that they are unable to properly classify the mood for the song. Due to the low amount of accuracy and precision, so we are unable to classify our dataset using these methods. Therefore at the moment, we are using our opinion to classify the dataset into a separate mood. Below are more of the explanation of each method result and discussion in more detail.

4.1.1 Mood Classification using ground Truth table result

```
Song Name :  midi_songs\0fithos.mid
Mean Pitch :  334.62702802383643
Tempo :  110.0


Song Name :  midi_songs\8.mid
Mean Pitch :  306.0950262797389
Tempo :  140.0


Song Name :  midi_songs\ahead_on_our_way_piano.mid
Mean Pitch :  366.19735355245933
Tempo :  120.0


Song Name :  midi_songs\balamb.mid
Mean Pitch :  405.88064124065113
Tempo :  120.0


Song Name :  midi_songs\bcm.mid
Mean Pitch :  376.37822901754316
Tempo :  87.0
```

Figure 4.1: Result of getting mean pitch and tempo.

Some of the results from extracting pitch and tempo of each song in the dataset are shown in figure 4.1. We use 92 songs as our test dataset, in order to test the method for classification with the ground truth table. The majority of the song contains the mean pitch around 200-400 Hertz, while only 8 song contains more than 500 Hertz. For the tempo result, only 2 songs contain a tempo more than 200 bpm but neither of those songs contains the mean pitch more than 400 Hertz.

This means that none of the songs in the dataset is classified for the mood happy. Because with the ground truth table we using, refer that the happy song would have a mean pitch of 967.47 Hertz and tempo of 209.01 bpm. Therefore there is the only song in the dataset with a mean pitch close to 900 Hertz and another song with a tempo near 209 bpm as seen in figure 4.2.

```
Song Name :  midi_songs\Kingdom_Hearts_Dearly_Beloved.mid
Mean Pitch :  838.8621307227936
Tempo :  128.0
Song Name :  midi_songs\great_war.mid
Mean Pitch :  399.79256301515363
Tempo :  210.0
```

Figure 4.2: Example of the song with a result similar to happy quantity.

We then try to use the song that we believe is a happy song and try to compare with the truth table. The song that we use called "Marry you" By Bruno Mars, and we test with 5 of the people to ensure that the song is indeed the happy mood. After confidence that the song is happy, we then run it through the program and get the mean pitch and tempo of the song as seen in figure 4.3. The result is that this song still not met the criterion of the table to be classified as a happy song.

```
Song Name :  MarryYou.mid
Mean Pitch :  493.109478614883
Tempo :  159.0
```

Figure 4.3: The mean pitch and tempo of "Marry you" song.

Therefore we believe that using this method may not be suitable since there is a problem when classifying for the mood happy. Another problem is the numeric value of the table is represented in term of average value. Therefore it may not be suitable since we have to set the range between each mood by ourselves and making the result not reliable.

4.1.2 Mood Classification using VAE and K-Mean clustering result

Figure 4.4 shows the result when preparing the data for VAE training. Each column is represented by note, velocity, and tempo, and it is repeated in this respective order until it reaches the minimum song note. In this case, the minimum song in the dataset contains 137 notes. Therefore the number of features for one sequence would be 411 features.

0.9541547277936963	0.9541547277936963.1	0.8080229226361032	0.9541547277936963.2	0.9914040114613181	0.9742120343839542	0.9570200573065902	0.9541547277936963.3	0.991
0.954155	0.808023	0.954155	0.991404	0.974212	0.957020	0.954155	0.991404	
0.808023	0.954155	0.991404	0.974212	0.957020	0.954155	0.991404	0.988539	
0.954155	0.991404	0.974212	0.957020	0.954155	0.991404	0.988539	0.905444	
0.991404	0.974212	0.957020	0.954155	0.991404	0.988539	0.905444	0.971347	
0.974212	0.957020	0.954155	0.991404	0.988539	0.905444	0.971347	0.954155	
0.957020	0.954155	0.991404	0.988539	0.905444	0.971347	0.954155	0.959885	

Figure 4.4: Notes, velocities, and a tempo data example.

We then use the data we prepare to train the VAE model in order to reduce the 411 features to 3 features. The training process took 15 seconds per epoch, and we train the model for 30 epoch. Therefore the total amount of time in training this model is 7.30 minutes. The training and validation loss is plotted with visualize graph as seen in figure 4.5. The validation loss is around 0.617 while the training loss is around 0.584. We

believe that the loss value for both training and validating is not too overfitting and not underfit, therefore we will use the model for the clustering.

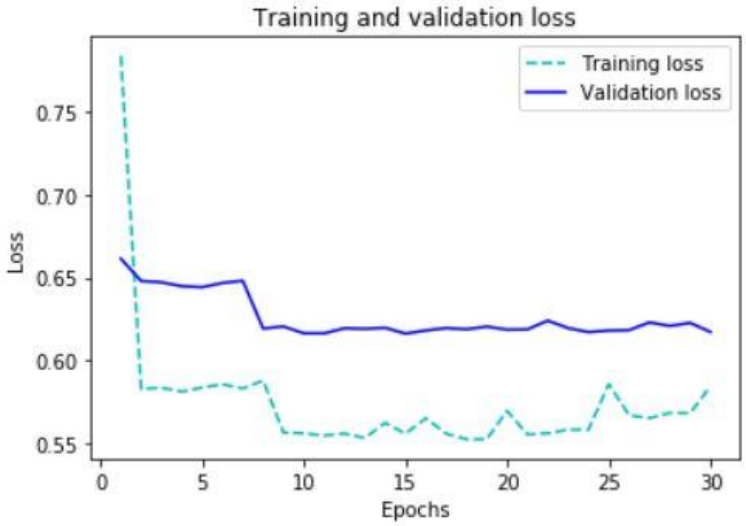

Figure 4.5: VAE model training and validation loss graph.

After the VAE model training completed, we will use the model to predict the output to use in clustering the data. But before clustering, we need to train the k-mean model to find the best number of a cluster first. By training the k-mean model with the output we will get the inertia and the score of the cluster as seen in figure 4.6. To choose the number of clusters we used the maximum difference of inertia and the score of the cluster, therefore we use 6 clusters for our data.

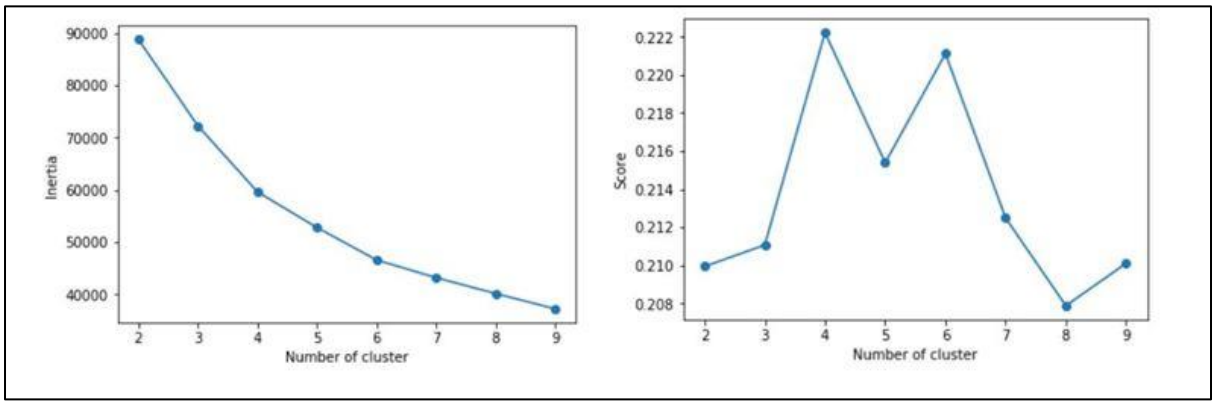

Figure 4.6: Inertia and score graph.

After we train the k-mean model and get the number of clusters, we then use the plot function to create a visualization of the output data as seen in figure 4.7. Because the prediction data is 3 features, therefore, the visualization is in 3 dimensional. Each color of the data is represented as a group of a cluster.

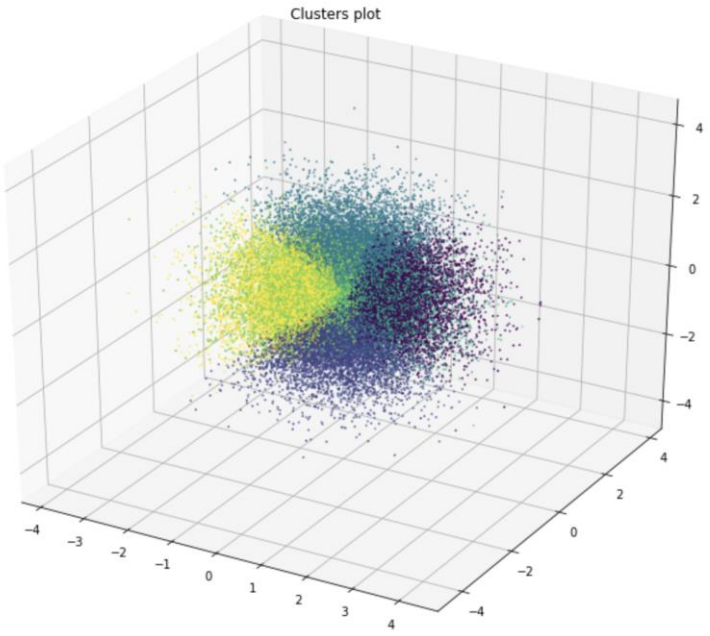

Figure 4.7: Visualization plot of clustering from the k-mean model.

When listening to a song in each cluster, the result is the song in the same cluster does not give the same feeling of moods. We believe that might be a problem due to the number of features used in the encoding model. The number of features may be too small to represent and thus not giving the result we hope. Another reason might be because of the way we represent the data, we may need to adjust the step when preparing the data input. Therefore in this state, the k-mean clustering is unable to classify the mood.

4.1.3 Mood classification using Spotify data results

	Song_Name	Mode	Mode_confidence	Tempo	Tempo_confidence	Key	Key_confidence	Time_Signature
0	The Night We Met (feat. Phoebe Bridgers)	1	0.694	174.127	0.169	2	0.564	3
1	Minimum	0	0.723	177.869	0.158	0	0.890	4
2	Hold Your Head Up High	1	0.570	132.094	0.669	9	0.583	4
3	Such A Simple Thing	1	0.769	76.350	0.245	10	0.768	4
4	Olalla	1	0.518	129.983	0.479	3	0.136	4
5	People Change	1	0.933	80.169	0.122	7	0.870	4
6	The Meetings of the Waters	1	0.689	105.016	0.683	11	0.619	4
7	Motion Sickness (Demo - Bonus Track)	1	0.616	95.136	0.307	1	0.629	4
8	Petals	0	0.423	152.205	0.142	9	0.272	4
9	Thousand (feat. Lisa Hannigan)	1	0.743	118.004	0.800	1	0.878	4
10	Foreign Hands	0	0.508	140.996	0.496	3	0.598	4
11	Sinking Ship	0	0.531	92.750	0.245	4	0.534	4
12	For a While	1	0.598	115.343	0.151	2	0.563	5
13	Call It Dreaming	1	0.654	80.329	0.434	5	0.738	4
14	Recording 15	1	0.452	92.441	0.370	11	0.242	4
15	Georgia	1	0.563	143.554	0.201	11	0.578	4
16	Beyond	1	0.457	76.014	0.339	6	0.383	4
17	Spirit Cold	1	0.606	78.247	0.012	11	0.593	4
18	Where's My Love	0	0.402	103.984	0.702	0	0.055	4
19	Even The Darkness Has Arms	1	0.688	101.275	0.675	11	0.751	4
20	Vagabond	1	0.019	141.922	0.793	4	0.355	4

Figure 4.8: Example Data obtain from Spotify.

From figure 4.8 shown the example of the data that obtain from Spotify which consists of the song title, key, tempo, and time signature. We use the Spotify API to obtain the data from approximately 200 playlists which contain around 2500 songs for each mood. We design to use the data and create a classification model by using 2 methods; decision Tree and Random Forest.

```
In [104]: Decision Tree
                    precision    recall  f1-score   support

                1.0       0.49      0.41      0.45       181
                2.0       0.45      0.12      0.19       218
                3.0       0.51      0.86      0.64       279

          micro avg       0.50      0.50      0.50       678
          macro avg       0.48      0.46      0.43       678
       weighted avg       0.49      0.50      0.45       678


In [105]: RandomForest
                    precision    recall  f1-score   support

                1.0       0.36      0.33      0.34       181
                2.0       0.36      0.43      0.39       218
                3.0       0.51      0.45      0.48       279

          micro avg       0.41      0.41      0.41       678
          macro avg       0.41      0.40      0.40       678
       weighted avg       0.42      0.41      0.41       678
```

Figure 4.9 Decision Tree model and Random Forest model score.

From figure 4.9 shows the result of both method in order to classify the song into 3 moods. The first method is the Decision Tree, the result from the classification report using with the test dataset show the average precision and score around 0.49 and 0.45 respectively. The second method is the Random forest, with the average precision and score around 0.42 and 0.41. It shows that even though the score of the Decision tree is better but still it's not high enough to be able to use. The problem might be because of the factor that we obtain might not be able to differentiate between song mood.

4.2 Model generate song result

```
Epoch 1/200
57509/57509 [==============================] - 613s 11ms/step - loss: 4.7577
Epoch 2/200
57509/57509 [==============================] - 609s 11ms/step - loss: 4.6889
Epoch 3/200
57509/57509 [==============================] - 609s 11ms/step - loss: 4.7426
Epoch 4/200
57509/57509 [==============================] - 609s 11ms/step - loss: 4.7102
Epoch 5/200
57509/57509 [==============================] - 608s 11ms/step - loss: 4.7120
Epoch 6/200
57509/57509 [==============================] - 609s 11ms/step - loss: 4.7138
Epoch 7/200
57509/57509 [==============================] - 609s 11ms/step - loss: 4.7081
Epoch 8/200
 9024/57509 [===>..........................] - ETA: 8:31 - loss: 4.7046
```

Figure 4.10: Training process of the model.

The training of the generating song with only note took around 10 minutes for each epoch as seen in figure 4.10. At first, we set it to train for 200 epochs but when it reached 92 epoch the loss value is around 0.5207. We think that with the loss value around 0.5207 should be enough so we stop the training process. Therefore the training process took around 920 minutes or around 15 hours.

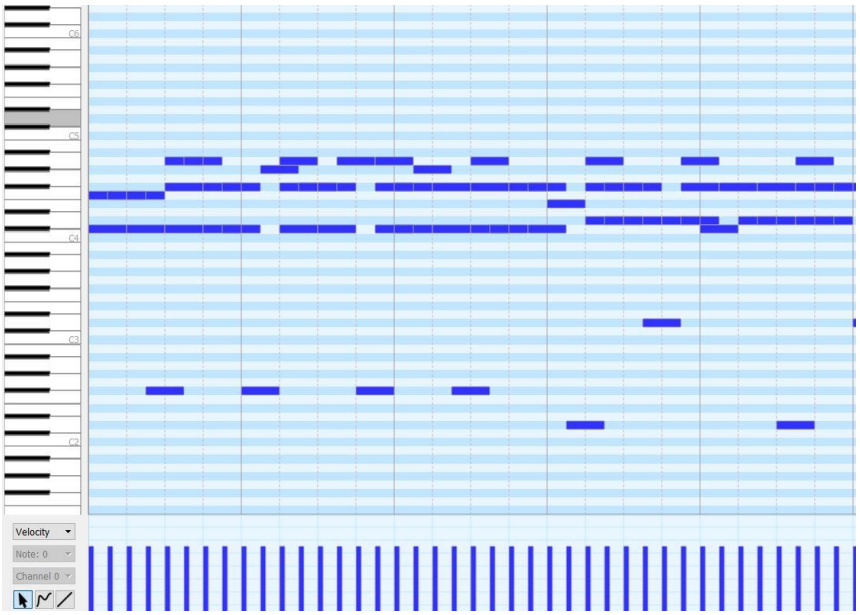

Figure 4.11: The first song generated by the model.

We then generate the first song from the model and open it via midi editor software as seen in figure 4.11. The result is the song generate from the model doesn't sound like the existing song in the dataset but may sound resemble some of the dataset songs. Therefore the model is able to generate a new song using this method.

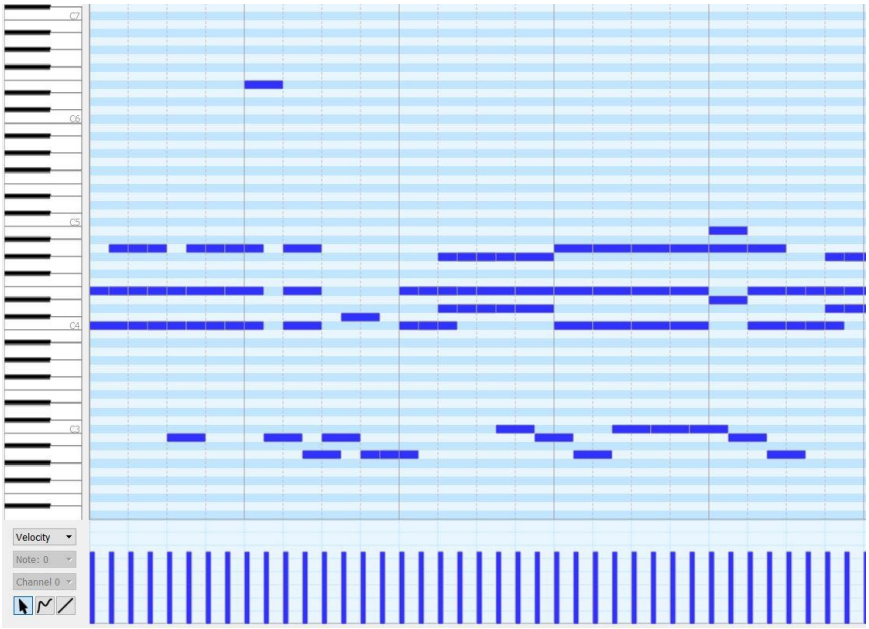

Figure 4.12: The second song generated by the model.

Next, we generate a second song from the model to compare and see if the song will be the same. The result is that both of the songs is not the same as seen in figure 4.12 since the randomly start pattern is not the same. But if we use the same start pattern the output song will be the same song. However, the probability that the song will be the same is 1 in 57077 which is the number of the pattern. Even though both songs are different the sound is similar due to the start chord is the same.

Another criterion we use is the pleasant or the melodic of the songs, which both of the songs did well. Based on our hearing we think that both of the songs are good and quite satisfying. We also played the songs for some of our friends and most of them seem to have the same idea as ours. However, the song is not yet resembled the human playing due to the lack of dynamic and timing. Since the model only predicted the note or chord and not other aspects of music.

4.3 Model generate song with more complexity result

For the model with complexity, we separate into 2 models, one with only velocity and the other with timeshift and velocity. The result of the first model can be seen in figure 4.13. which the model be able to predict the note and its velocity. However, in term of melodic the performance not as well as the simple note one. This may be because of the model input that is in two separate sequences, thus given the result not satisfying enough.

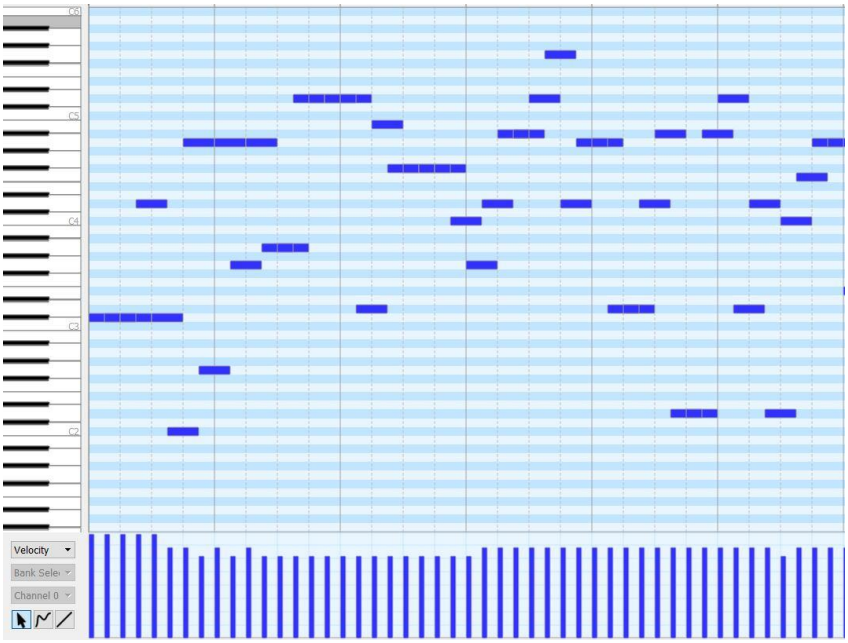

Figure 4.13: example song from the model with velocity.

The second model training using only one set of sequence and with velocity and timeshift element. The example song we generate from the model sound more pleasant and more melodic than the one with a separate sequence s seen in figure 4.14. The song has more resemblance to human playing because of the dynamic and timing. Another reason is because of the dataset that we use to train the model is the actual song record by human playing. Therefore for the finalize model, we will use this model structure and dataset for the web application.

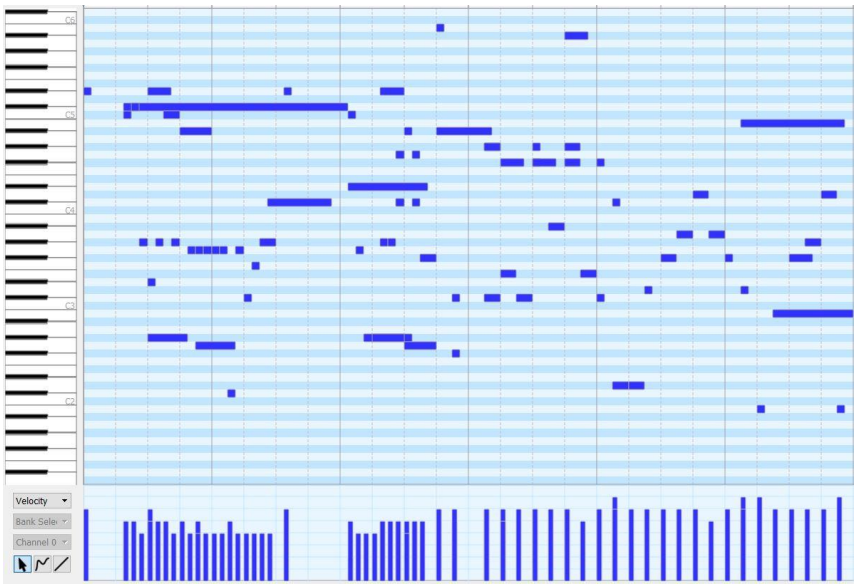

Figure 4.14: Example song from the model with timeshift and velocity.

Happy	Relax	Sad
loss: 0.9830 - perplexity: 6.1591	loss: 0.6929 - perplexity: 4.3145	loss: 0.8185 - perplexity: 4.3274
loss: 0.9330 - perplexity: 6.0365	loss: 0.7120 - perplexity: 4.5088	loss: 0.8110 - perplexity: 4.4094
loss: 0.9196 - perplexity: 5.9573	loss: 0.7150 - perplexity: 4.5806	loss: 0.7428 - perplexity: 4.0923
loss: 0.8763 - perplexity: 5.9910	loss: 0.6753 - perplexity: 6.1006	loss: 0.7302 - perplexity: 4.0464
loss: 0.8511 - perplexity: 5.9497	loss: 0.6742 - perplexity: 4.7179	loss: 0.7049 - perplexity: 3.9660
loss: 0.7991 - perplexity: 6.2241	loss: 0.7327 - perplexity: 7.4482	loss: 0.6652 - perplexity: 3.9061
loss: 0.8095 - perplexity: 5.7603	loss: 0.6326 - perplexity: 4.5281	loss: 0.6661 - perplexity: 3.8382
loss: 0.7548 - perplexity: 5.7837	loss: 0.5640 - perplexity: 4.0877	loss: 0.6174 - perplexity: 3.5849

Figure 4.15: Training result of three models.

From figure 4.15 shows the loss value and perplexity from the training process of Happy, relax, and sad model. Using the perplexity to be our evaluation criterion for our model and setting the early stop based on the perplexity as well. Therefore the result of happy, relax, and sad perplexity are 5.7837, 4.0877, and 3.5849 simultaneously.

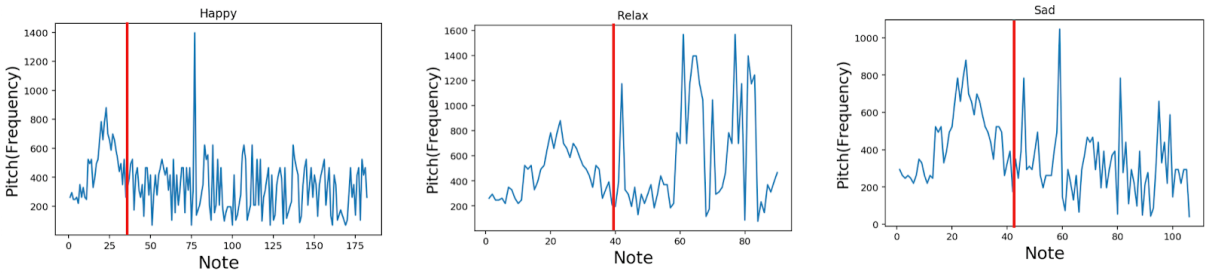

Figure 4.16: Pitch frequency of the input song with model output song.

From figure 4.16 shows the frequency of the input song and the output song of each model. The red bar in each graph represents the connection between the input and the output. The Y-axis represents the frequency of the note while the X-axis represents the note in the song. All of the output songs generated with a duration of 20 seconds. Therefore it shows that the more notes indicate the higher Beat Per Minutes. In this case, the happy model produces the song with the highest notes while the lowest is relax model.

4.4 Backend result from using Flask

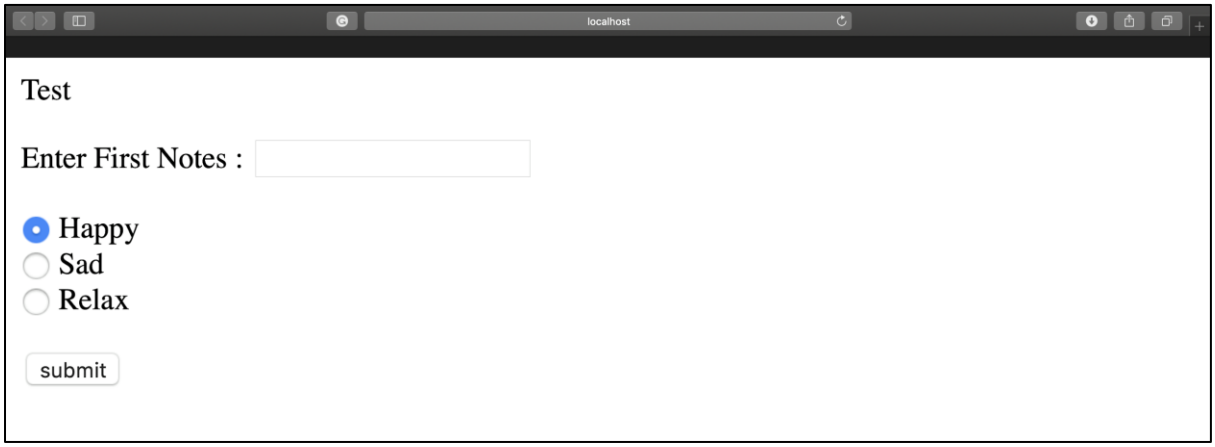

Figure 4.17: Example of receiving input.

From figure 4.17 shows the sample user interface for the backend of the web application in order to test receiving the user input. We create a simple HTML page in order to get the note input and mood from the user. Then we will try to pass the input into the model that merged with our backend.

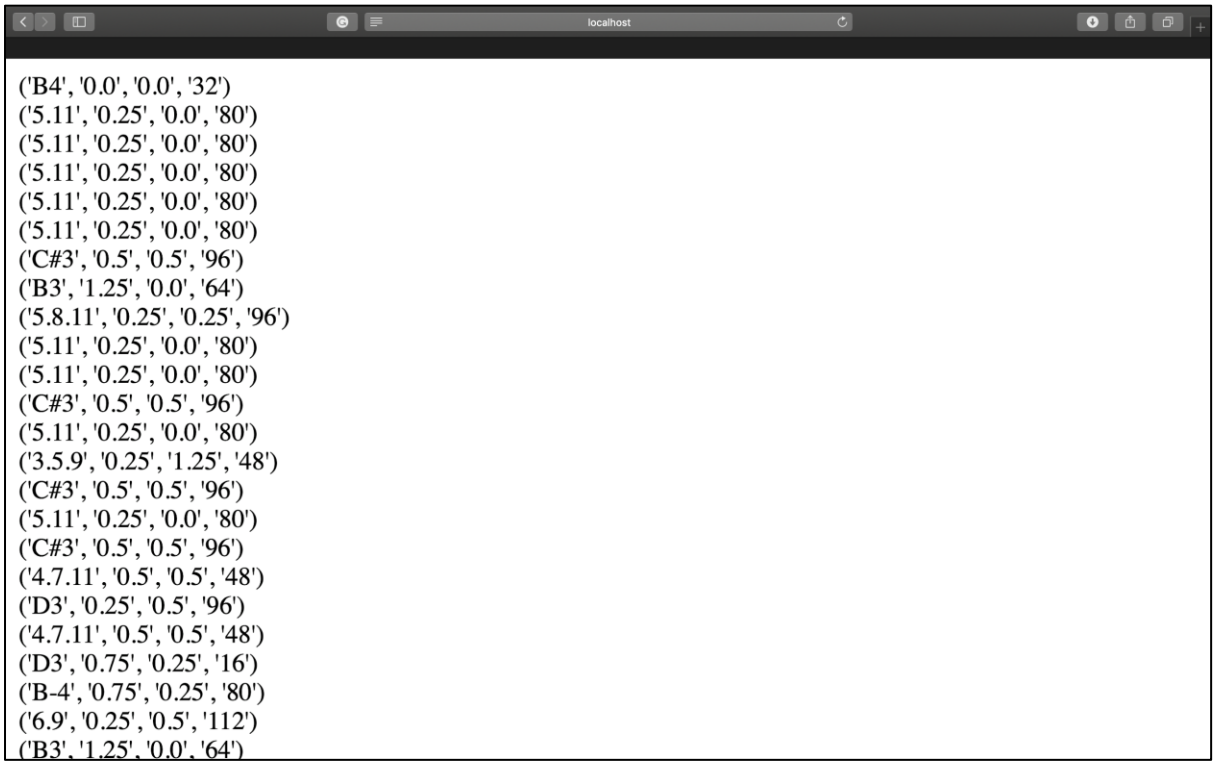

Figure 4.18: Result from the model view from HTML page.

The backend will generate the song and send it back in JSON format to the HTML file as seen in Figure 4.18 Each of the lines contains the note, duration, offset, and velocity respectively. Therefore it shows that the backend of the web application now can communicate with the model and able to generate the song using the input from the user.

4.5 Piano integration result

From figure 4.19 show the screenshot capture from when playing the piano and sending the input to the web application. After selecting the mood which will indicate the model to use. When pressing record and play the piano, all the notes, and its velocity, duration and offset will be recorded and will be used as the input for the selected model. However, due to the midi API problem, we are unable to receive the playing from the piano in the form of the midi message. Therefore while receiving the raw data from the midi controller, we need to process the data and calculated each duration, offset, and chords manually. Thus some of the chords might not be recognized at the moment.

Figure 4.19: Example of playing the piano and sending the input to the web application.

After receiving the input from the user and into the generate part, the model will use the input and start the prediction process as seen in figure 4.20. The upper part indicates the number that is according to the dataset dictionary. While the lower part indicates the result translates from the dictionary into the notes, duration, offset, and velocity. Then using the result to translate back into the midi. Therefore from this test show the integration between piano and web application.

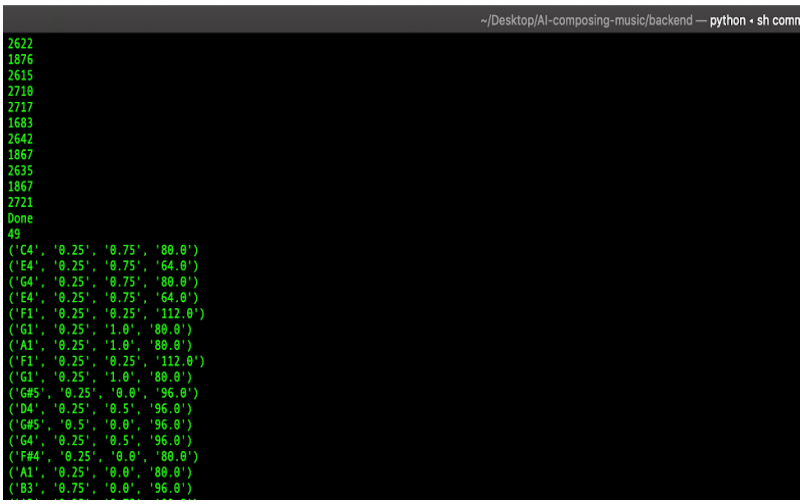

Figure 4.20: The result generates from the model.

4.6 User Evaluation

Figure 4.21: Testing the program with users.

From figure 4.21 we evaluate the model by gathering the feedback from 39 users according to the model performance and the inspiration receiving from listening to the generated song. After each user using the piano and generate the song from each model we evaluate by using the rating between the score of 1-5 (1 = very low, 5 = very high). The user will be rating model based on the feeling while listening to the song whether it matches the chosen mood or not. Another score is the overall melodiousness from the model generating songs as seen in figure 4.22.

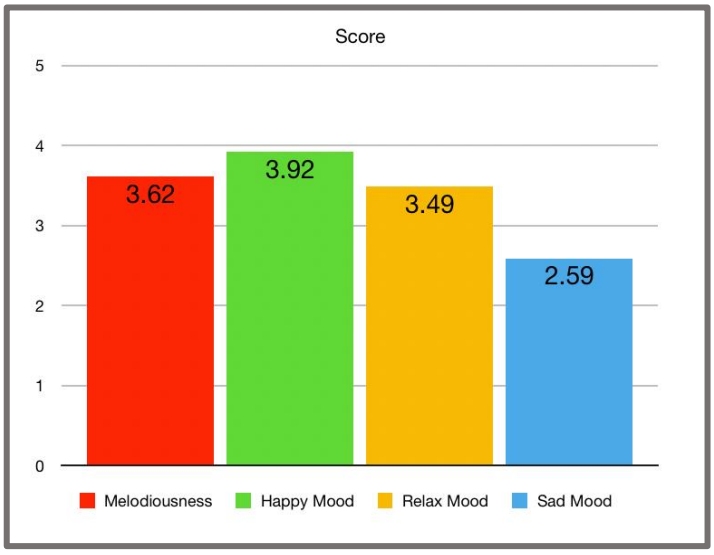

Figure 4.22: Bar chart representing the average model scores and overall melodiousness.

The highest rating from all of the three models is the happy model while the lowest is the sad model. The reason for this scoring might be because of the happy model have the lowest lost and perplexity values, while the sad model is the opposite. Even though both models lost and perplexity values might not differentiate much. For happy the lost and perplexity is 0.53 and 4.84 respectively. While the sad model lost and perplexity is 0.65 and 5.81. It shows that from the human perspective there is some noticeable result thus making the sad score lower than happy. However, there is another factor which is the dataset of the sad songs might not be a good representative of the sad mood.

The overall melodiousness from the model generates song score indicate that many of the users find that the song is quite melodic. With the average scores of 3.62 in shows that in term of song quality the model has been working the decent job. However, the song that generated might not be related to the user input much. Due to the low number of dataset thus making the song unable to truly fit with the user input.

From figure 4.23 shows the comparison score between people who know music theory and people who don't in term of the inspiration received after listening to the model generated songs. People with music knowledge tend to get more inspiration from listening to the model generated songs. With the percentage of 83 of people with music knowledge given the score around 3 or higher. On the other hand, people without music knowledge tend to give less score with the majority given the score of 1 and 3. It shows that the majority of people with music knowledge get the inspiration to create music.

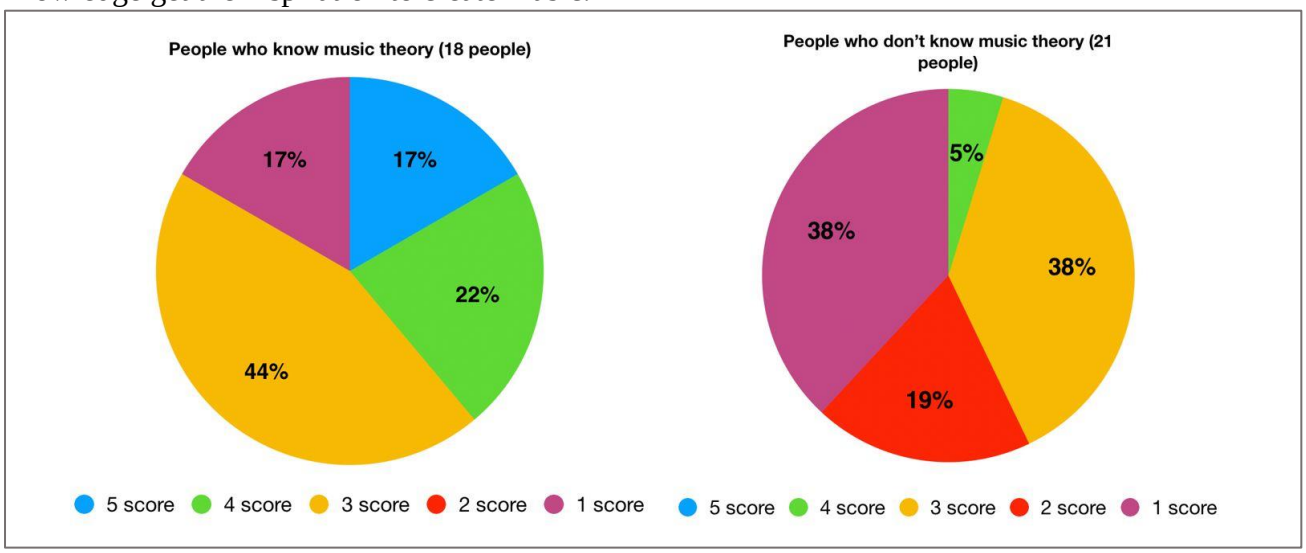

Figure 4.23: Pie charts comparison of inspiration scores between people with music theory knowledge and people without music theory.

Another comparison is between the people with the piano skill and people without as seen in figure 4.24. Due to the song that the model generated and the user input is based on the piano instrument only. Therefore we try to specify these people and try to evaluate the inspiration scores around them.

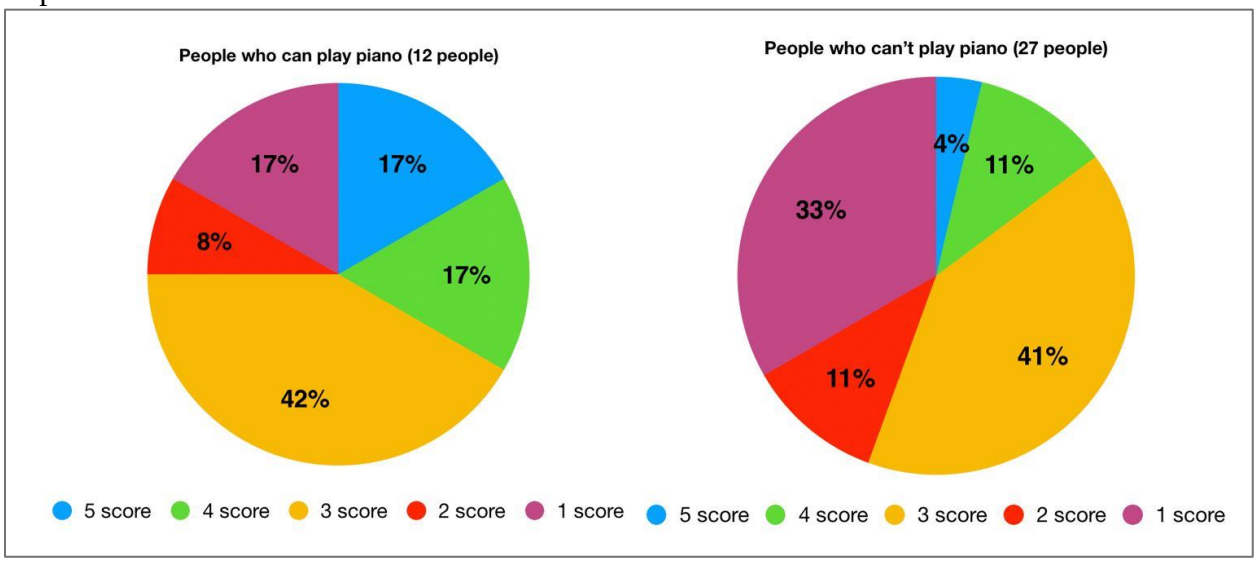

Figure 4.24: Pie charts comparison of inspiration scores between people with piano skill and people without piano skill.

While the majority of people who have the piano skill rate the inspiration with 3 or higher score. However, the number of percentages is less than the last one according to the result. In the other hand, people who do not possess the piano skill gain more inspiration. It shows that there are some people without piano skill who receive the inspiration by listening to only the piano instrument songs.

Chapter 5
Conclusion

5.1 Project Summary

Our project is to develop a machine learning application for composing piano songs with a different kind of mood in order to give inspiration for the musicians. Using the LSTM methods and train on 3 separate datasets based on 3 moods happy, sad, and relax. Then using the web application as the user interface and main interaction between the user and the model.

The input will be via from user playing the piano and selecting the mood and duration of a generate song. While the output will be the prediction from the model combine together with the user input which then converts into the MIDI files. Then we evaluate the model result and its objective via user testing.

5.2 Completion Status

Table 5.1: Completion status of the deliverables.

Deliverable Type	Deliverable	Status
Machine Learning	Collect Data	Completed
	Separate Mood of each song	Completed
	Training the machine learning model	Completed
Web Application	Design UI	Completed
	Backend	Completed
	Frontend	Partially Completed
Experimental Result	User-Evaluation	Completed

From table 5.1 shows the completion status of the project deliverables for each task.

We separate the deliverable type into 3 parts, Machine learning, Web application, and Experiment result. For the machine learning part, all the tasks are completed, however, the sad model might not be the best quality compared with the other two models. For the web application part, the main core of the project is complete however, the frontend might not be the same as the design but still useable. Therefore the status of the frontend is marked as partially completed. For the experimental result, we test the program with users and collected the result in term of model performance and inspiration and evaluate them. Thus completing the user-evaluation part.

5.3 Issues Encountered and Solved

5.3.1 Dataset Issue

The main issue we encountered with the dataset is the mood classification part. Since we want to create a model to generate songs based on different kind of moods. Therefore we need to train the model using the song with separate mood. However, the problem occurred when we want to identify the song mood. We try to use 3 methods to classify the song; ground truth table, K-mean clustering, and Spotify data. But neither of the three methods work. Therefore the method we used to separate the dataset is by using human judgment. We listen to each song in our dataset and find the song that we agreed to be the best representation of the mood. Thus given each mood we only use the number of songs around 10 songs.

5.3.2 Model Issue

In the model performance part, the problems can be summarized into 2 factors; the overfitting of the data and the data processing. For the data processing part, the problem that we encountered is from the method that we use both input and output.

The input we used the dictionary function to map the dataset into key values. Thus using this method each of the values will have no correlation with others even if it's the same notes. Therefore when reading the input with no note or chord in the dataset, we need to cut the specific notes of the input to avoid the error. While the output we use the one-hot encoding method to create a multiclass prediction. However, with the amount of class depending on the number of value in the dictionary function. Thus making the number of output class tremendous. Therefore training the model would take longer time and making the loss value and perplexity higher.

For the overfitting data part, it occurred due to the process we used to data making the output class very high. Therefore when using the number of songs we are unable to use too many songs in order to prevent the number of output class to be excessive. Thus making the model generated songs quite similar to the dataset songs.

5.3.3 MIDI API

The problem is occurred due to the unable to use the MIDI API for the web application. Therefore while receiving the raw data from the midi controller/piano, we need to translate the information into the note/chord, duration, and offset by ourselves. Thus making the chord receiving not as variant as the standard usual chord, since we need to map each chord individually.

5.4 What we had learned

During the time we develop this project, we acquired the knowledge in both computer engineering field and the musical theory. Since this project is related to creating music, therefore we need to learn about music theory first in order to understand the information while deconstructing the midi file and when processing with the data.

While the computer engineering field knowledge, we can separate into two part; machine learning and web application. For the machine learning part, since this is our first time creating the neural network model, we both have been experiment with the model. Before becoming the LSTM model we need to learn how to processing with the data, and to choose the structure of the model. Many trial and errors occurred before becoming the model we use in the latest version. Apart from the LSTM model we also learned about other methods such as the K-mean clustering through the music mood classification.

For the web application part, we can separate it into the frontend and the backend. In the frontend not only we learned about the basic part such as CSS, HTML, and javascript we also using the VUE as a framework, therefore we need to learn about its as well. In the backend part, we learned how to create Restful API by using flask in order to integrate the model with the web application.

Other important things we both learned besides the knowledge within the previous two fields are decision making and communication. During the time we develop the project, many decisions need to be made in order to progress into the next step. Even the decision such as how many moods to use could be a huge impact on the project. Therefore we both need to make decisions based on the time constraint and the limitation of knowledge.

Communication is another core thing we learned during working on this project. Without communication, we both would unable to perform even a simple task. The best example of the situation is Github. Both of us even though we are working on the different part, however, some of them are on the same file. Therefore both of us need to tell the other before

and after each push in order to avoid the overlapping of the same file. Then when we learned the communication skill the teamwork skill will be automatically learned.

5.5 Future extending suggestion

5.5.1 Improve the machine learning model

One of the future extension which would be the very core of the project is to improve the machine learning. There could be a better way of processing the input and output of the model besides mapping one to one like the dictionary function and one-hot encoding. If we could reduce the number of output class the perplexity value would be reduced, thus making the model better. While mapping input in order to make the note and chord having the relation between them could prevent the discarding of the unknown input.

Since the machine learning model is well-received at the current time, there are many new algorithms or methods that could be used to create a better model. It could be more efficient in term of training time or it could generate a song which holds the long term effect better.

5.5.2 Improve the integration between the piano and web application

Other important suggestion is to improve the integration between the piano and the web application. At the moment the MIDI API still unable to use therefore most of the input from the piano were manually process. Therefore to improve the integration if not using the MIDI API then it needs to improve the chord receiving from the piano. By adding more chord to the process and improve the identification in order to get the chord more accurate.

5.5.3 Improve the user interface

Even though the user interface might not be the main core of the project, however, it’s the project main interaction with the user. One of them is to add the animation for the pressed notes in order for the user to know the previous pressed note. Other than that user interface could add the way to represent the midi song in order for the musician to see the result better than only listening to the result.

5.5.4 Add more moods and instruments

Adding more moods and instruments should give more variety to the user when generating the music. By adding more moods the user can choose to create the song in which is based on the feeling that the user intends to create. The other is to add more instrument to give more option other than piano only should give the player more tone of the music, since the same note on the different instrument gives the different sound.

References

1. Malik, I. " **Neural Translation of Musical Style** ". University of Bristol (2017).
2. Nuzzolo, M. (2018)."**Music Mood Classification**" Electrical and Computer Engineering Design Handbook. Retrieved from https://sites.tufts.edu/eeseniordesignhandbook/2015/music-mood-classification/
3. BRANNON DORSEY, **"Using Machine Learning to Create New Melodies"**[Online], Retrieved from: https://brangerbriz.com/blog/using-machine-learning-to-create-new-melodies/(2018 October 1).
4. Andrew Pouska, **"The Elements of Music"[**Online], Retrieved from: https://www.studybass.com/lessons/basics/the-elements-of-music/2(2018 October 1).
5. Joe Wolfe, **"Note names, MIDI numbers and frequencies"**[Online], Retrieved from: https://newt.phys.unsw.edu.au/jw/notes.html
6. Quizlet, **"Chorus- Ten elements of music"**[Online], Retrieved from: https://quizlet.com/2289212/chorus-ten-elements-of-music-flash-cards/
7. colah's blog, **"Understanding LSTM Networks"**[Online], Retrieved from github: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
8. Amit Shekhar, **"Understanding The Recurrent Neural Network"**[Online], Retrieved from medium: https://medium.com/mindorks/understanding-the-recurrent-neural-network-44d593f112a2
9. Skymind, **"A Beginner's Guide to LSTM"**[Online], Retrieved from: https://skymind.ai/wiki/lstm#recurrent
10. Jon Brantingham,**"The Secrets Behind the Music Composing Process"**[Online], Retrieved from: https://www.artofcomposing.com/music-composing-process
11. Jonathan Wright, **"Seven ways to find Inspiration to Compose Music"**[Online], Retrieved from: http://www.jonathanwrightmusic.com/seven-ways-find-inspiration-to-compose-music/
12. Keras Team, **"Keras"**[Online], Retrieved from" https://keras.io/
13. Serdar Yegulalp, **"What is TensorFlow? The machine learning library explained"**[Online], Retrieved from: https://www.infoworld.com/article/3278008/tensorflow/what-is-tensorflow-the-machine-learning-library-explained.html
14. Association, T. (2018). "**About MIDI-Part 1:Overview"**. Retrieved from https://www.midi.org/articles/about-midi-part-1-overview

15. Michael Cuthbert, **"music21"**[Online], Retrieved from: http://web.mit.edu/music21/doc/about/what.html#
16. Bartu Kaleagasi , **"A New AI Can Write Music as Well as a Human Composer"**[Online] Retrieved from: ,https://futurism.com/a-new-ai-can-write-music-as-well-as-a-human-composer
17. Pierre Barreau, **"AIVA"**[Online], Retrieved from: https://www.aiva.ai/
18. Emma Featherstone, **"Introducing the next generation of music makers"**[Online], Retrieved from: https://www.theguardian.com/small-business-network/2017/aug/29/computer-write-music-jukedeck-artificial-intelligence
19. Jukedeck Team, **"Jukedeck"**[Online], Retrieved from :https://www.jukedeck.com
20. Kavin, "**Variational Autoencoders Explained**"[Online], Retrieved from: http://kvfrans.com/variational-autoencoders-explained/
21. Irhum Shafkat, "**Intuitively Understanding Variational Autoencoders**"[Online], Retrieved from : https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf
22. Ajit Rajasekharan, "**What's the difference between a Variational Autoencoder (VAE) and an Autoencoder?**"[Online], Retrieved from :https://www.quora.com/Whats-the-difference-between-a-Variational-Autoencoder-VAE-and-an-Autoencoder
23. François Chollet, "**Building Autoencoders in Keras**"[Online], Retrieved from :https://blog.keras.io/building-autoencoders-in-keras.html
24. Dr. Michael J. Garbade, "**Understanding K-means Clustering in Machine Learning**"[Online], Retrieved from :https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
25. Andrea Trevino, "**Introduction to K-means Clustering**"[Online], Retrieved from :https://www.datascience.com/blog/k-means-clustering
26. Spotify, "**Spotify Dashboard**"[Online], Retrieved from :https://developer.spotify.com/dashboard/applications/03ca22b369fc4b69a6c18d6322a17fae
27. Spotify, "**Get Audio Analysis for a Track**"[Online], Retrieved from :https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-analysis/
28. Aerin Kim, "**Perplexity Intuition (and its derivation)**"[Online], Retrieved from :https://towardsdatascience.com/perplexity-intuition-and-derivation-

105dd481c8f3?fbclid=IwAR2PznKQJPdq4F3YdzLoyj2VQTf3bTpp1bNUsevVDf4D6RjzXmS1Bn2wmEY

29. Surya Bhupatiraju, "**What metrics should I use to evaluate a generative model quantitatively, and how reliable are they?**"[Online], Retrieved from :https://www.quora.com/What-metrics-should-I-use-to-evaluate-a-generative-model-quantitatively-and-how-reliable-are-they?fbclid=IwAR2PNau3Ifin4ry9XU3xAmgJMw_zWDkwRoOlmZVRRbzG8rAOMo1qWq2-nBQ
30. lukkiddd, "**Language modeling** มาสร้างประโยคกันเถอะ"[Online], Retrieved from :https://lukkiddd.com/language-modeling-%E0%B8%A1%E0%B8%B2%E0%B8%AA%E0%B8%A3%E0%B9%89%E0%B8%B2%E0%B8%87%E0%B8%9B%E0%B8%A3%E0%B8%B0%E0%B9%82%E0%B8%A2%E0%B8%84%E0%B8%81%E0%B8%B1%E0%B8%99%E0%B9%80%E0%B8%96%E0%B8%AD%E0%B8%B0-64e81efc7d92?fbclid=IwAR1qEE1lYAO2dq_k9Wuewkb0-huL5PIDP7UzXYAhLju9WBlQ_raIjzemTDU
31. Naveen Manwani, "**Generative Deep Learning: Let's seek how AI Extending, not Replacing Creative Process**"[Online], Retrieved from :https://towardsdatascience.com/generative-deep-learning-lets-seek-how-ai-extending-not-replacing-creative-process-fded15b0561b
32. Prakash Pandey, "**Deep Generative Models**"[Online], Retrieved from :https://towardsdatascience.com/deep-generative-models-25ab2821afd3