AN EFFICIENT AI MUSIC GENERATION MOBILE PLATFORM BASED ON MACHINE LEARNING AND ANN NETWORK

Jiacheng Dai[1] and Yu Sun[2]

[1]International Department of the Affiliated High School of SCNU,
Zhongshan Road West No.1, Guangzhou, China
[2]California State Polytechnic University, Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

The aim of this paper is to provide a solution to the growing need for fresh music to use in media, as adding music can greatly enhance the media's atmosphere and the viewers' experience [6]. Our solution to this issue was the creation of a mobile application named MFly that can output music using the sentiment from an inputted message. To test the effectiveness of this new music-generating method, an experiment was conducted in which twenty-three participants inputted a message with a positive and negative sentiment each and recorded whether each outputted musical piece accurately represented the sentiment from the message [7]. A post-experiment survey was also provided to each of the participants to gauge the convenience and practicality of the application. The results indicated that MFly was largely successful at conveying messages into appropriately fitting music. However, the practicality of the application could use some work, as generating music based on the sentiment does not always seem to match up with the original inputted message's sentiment, especially with messages that have a negative sentiment. Furthermore, feedback from participants indicated that the application could still improve with the addition of more features, such as the ability to save the generated music for later use.

KEYWORDS

Machine Learning, AI, Mobile application.

1. INTRODUCTION

Music has played a very prevalent part in human culture for centuries. From films and television shows to festivals and funerals, they are present in many aspects of life. Music has been shown in studies to provide benefits to those who listen to it. One of the benefits of music is improved memory because the brain can link sounds and music with certain experiences and places. Music can provide other various benefits related to improving the mental well-being of those who listen to it, such as reducing stress levels while boosting mood and improving performance in cognitive tasks [1]. In real life, music can also provide people with a path for them to express their creativity and channel their thoughts and emotions, help connect people with different cultural backgrounds with the power of music, and help create certain atmospheres like the ones required in night clubs.

Music is significant because it is capable of enhancing an event or a form of media [8]. By adding music to a film or a video game, the viewer or player can experience a greater level of immersion by having the music reflect and intensify the atmosphere [9]. Another quality that can

David C. Wyld et al. (Eds): SIP, CRBL, WiMNeT, BIOM, EDUPT, NLDM, SOFTFM, COMIT - 2022
pp. 43-55, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121705

makes music an important part of society is that it can help bring people together [10]. People can bond with each other over their passion for music, whether it be by listening to music, playing music on an instrument, singing, or creating their own music. This sense of bonding is not limited to a personal level; entire communities and cultures can blend together with the help of music. By having two cultures share their music with one another, both cultures can gain a better understanding of each other, which can improve cultural diversity globally. These many common uses of music call for a quick and convenient method of generating new music.

There have been attempts to generate music using artificial intelligence by taking in an existing song or melody as input [15]. Some of these programs have options in order to provide more flexibility to those who want a certain quality or element of music in particular. One way this has been done is by producing multiple different music pieces based on the one melody or audio file that was inputted. The individual parts that make up the music, such as the melody, the bass, and the drums, could also be mixed and matched in some cases. This gave the program users much control over the generation of the final output music. Another way to bring the program users more flexibility is to provide them with values to adjust. Using sliders, they can adjust qualities such as whether the song is in a major key or minor key and how conventional the song is.

However, the ability of these programs to generate music is limited by whatever is being inputted. Because audio usually has to be inputted in order for the artificial intelligence to generate music, it will usually be very similar to whatever the original audio is. Despite the flexibility and increased options of having multiple possible musical elements to choose from and adjust, the generated music still bears a striking resemblance to what was inputted in most scenarios. There are relatively few existing programs that can competently output music based on user-inputted values without any inputted reference music. Another general issue with AI-generated music is that certain “personal touches” that human musicians with superior knowledge and understanding of music and specific music styles are generally difficult to emulate or reproduce with artificial intelligence. Therefore, artificial intelligence that specializes in generating music may need to be further refined before it can see widespread usage.

The method presented in the research is a mobile application named MFly that generates music using artificial intelligence. The two main features of the application are generating music based on inputted reference audio and generating music based on the sentiment of inputted messages. With the reference audio, the artificial intelligence will attempt to generate a musical piece that sounds similar to the reference audio. Using messages, the application will attempt to derive the sentiment based on the connotation of the words used. Sentiments can either be positive or negative. Then, the application will output music that fits as closely to the selected sentiment as possible.

Other applications have a similar approach in terms of using reference audio to generate music, almost as if the application was trying to predict what an extension of the original audio would sound like. What sets MFly apart from many other applications that were developed for the same purpose is that this application aims to convert messages into music. While it is not advanced enough to detect specific keywords and make intricate changes like adjusting the specific style of music, this project aims to be a stepping stone towards an application that can eventually do so.

The remainder of the paper is structured into sections labeled 2 through 6. Section 2 provides insight into the challenges that were faced when creating the application. Section 3 dives into the overall development of the application as well as some specific key parts of the application, while Section 4 explains the experiments that took place and how the experiments’ results proved the effectiveness of this application at properly providing music based on sentiment. In Section 5, related works involving music generated by artificial intelligence are explored and

compared to this paper. Lastly, Section 6 ends the paper with concluding remarks and how the limitations of the application can be resolved in the future.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Brainstorming the Music-Generating Methods

One major hurdle that had to be overcome when developing the MFly mobile application was figuring out what options the users should be provided with when generating the music in the MFly application. To make the application more welcoming to those who are familiar with applications of the same purpose, the traditional implementation of inputting a reference song was added as a choice for the users to generate music. However, the application would preferably have a relatively unconventional method of generating music compared to most other AI-generated music applications. When brainstorming ideas for what such a method could be, the concept of being able to return musical pieces using written messages sounded unique and intriguing. Our vision behind this second music-generating method was that depending on what the message said and the connotation that the words in the message had, the application would output a musical piece to match the tone of the message.

2.2. Implementing AI-Generation of Music using Messages

The next challenge, however, came with the method of generating music using messages. Such a task sounded incredibly daunting at first, as there does not seem to be one simple step to convert a written message into a piece of music. Rather, the message must instead undergo multiple steps to reach the final desired output. The message would first be converted into two possible sentiments: positive or negative. The plan would be to use the keywords and the connotation of certain words to determine the sentiment. After retrieving the sentiment, the application can select a song based on a list of songs, and the one that is selected depends on whether the mood of the message is determined to be positive or negative. By breaking the process down into two main parts, the issue of outputting a song to the user with artificial intelligence using a written message was successfully overcome.

2.3. Testing the AI-Generation of Music

Lastly, one of the largest challenges was gauging the accuracy and quality of the AI-generated music through experiments or surveys. As people have varying tastes in music and what sounds like good music to some might sound like terrible music to others, whether the music generated by the AI was high-quality or not is very subjective. While such a question could still be included in a post-experiment survey, a more concrete value to test for was whether the sentiment in the written message was reflected accurately in the music. For example, a positive message should result in a positive and happy-sounding piece of music, and a sad and negative message should result in a sad-sounding piece of music. As there are still differences in opinion as to what music accurately reflects the sentiment of a message and what does not among people, as many participants as possible from multiple diverse communities would be needed to reflect the opinion of the general public as accurately as possible.

3. SOLUTION

MFly was created using Flutter and Python; Flutter primarily makes up the front-end of the code, while Python acts as the back-end of the code. Using multiple Dart files, the application was able to cycle through various screens, including the splash screen when first opening the application and the home screen that appears right after [11]. There are two possible options on the home screen labeled by two buttons. The first button leads to a screen for inputting audio, and the second button leads to a screen for inputting a message. On both screens, after the data has been inputted, the application transfers over to the screen that plays the generated music. The Python file was responsible for handling the artificial intelligence aspect of the application. Artificial intelligence is primarily made using Keras, which is an open-source software library. In particular, the Long Short-Term Memory neural network was used, which is a kind of recurrent neural network that utilizes sequential information and learns through gradient descent. Because the Long Short-Term Memory neural network has the capability to recognize and encode patterns over an extended period of time, it excels at tasks such as music generation. When generating music, the application starts with a MIDI file [12]. First, the sequences used by the neural network are prepared. Then, the model is created and is used to generate a prediction of how the music should continue using the patterns it recognizes. Lastly, the MIDI file is converted back into an audio file for the audio player screen.

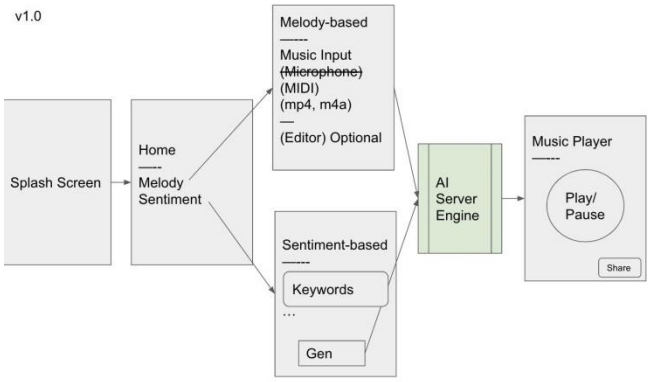

Figure 1. Overview of the solution

When the application is first opened, the splash screen appears. This splash screen is handled with a single Dart file in which its sole purpose is to flash the application logos on the screen for a few seconds. This was performed by creating a Future that would switch to the main menu screen after a 3-second delay. From the main menu screen, elevated buttons allow the user to navigate around the different screens in the application with Flutter's Navigator. The two main options in the main menu are Melody AI and Sentiment AI; Melody AI uses reference audio as input, while Sentiment AI uses a message as input.

To implement the "Melody AI" option of the main menu screen, the audio file is converted into a MIDI file. The audio is retrieved using the path where the file is located, then the audio is loaded with Librosa (a Python package for music and audio analysis). In the MIDI file, only the piano was used in order to make the training of the model as effective as possible. Therefore, the generated music's only instrument is the piano. A pre-trained model was fetched for the purpose of this application [5]. The notes of the MIDI file are extracted, then the notes used to train the pre-trained model are extracted and sorted so that all tested pitch names could be saved. The extracted notes from the input audio and the pitch names from the train data are used to prepare the neural network sequences, then a Long Short-Term Memory neural network is created.

Exactly 250 notes for the song are created using the LSTM model, and the generated notes are compiled into a MIDI file [13]. The MIDI file is converted into a WAV file, which appears on the music player screen and allows the users to listen to the video. During this process of conversion, a Transcription method based on Magenta Onsets and Frames is used, which is a newly improved transcription method with the help of CNN and LSTM models. The method hugely improves the capability of music transcription by separating note detections into two stacks of neural networks instead of one. After installing Magenda, importing libraries, and fetching Magenda, the MIDI file can finally be generated.

To implement the "Sentiment AI" option of the main menu screen, the sentiment analysis was performed by taking the words of the message as a String and running the words through a method to generate the sentiment. The Natural Language Toolkit, or NLTK for short, is imported into Python and the VADER lexicon is downloaded. VADER stands for Valence Aware Dictionary and sEntiment Reasoner, which is a rule-based sentiment analysis tool that specializes in the ability to detect the sentiment of texts. A sentiment intensity analyzer that was imported from the Natural Language Toolkit is created. Then, a method to find the polarity scores using the aforementioned sentiment intensity analyzer is called with the String as a parameter and its value is returned. The polarity score can be anywhere between -1 and 1, in which a negative score means that the message has a negative sentiment and a positive score means that the message has a positive sentiment. Depending on what the score is, a variable representing mood will be assigned a certain value to determine which song will be chosen to be played.

```
void uploadFileToServer(var filePath) async {
  print("uploading ... ");
  setState(() {
    loading=true;
  });
  var url = Config.serverUrl + '/melody';
  print(url);
  http.MultipartRequest request = http.MultipartRequest('POST', Uri.parse('$url'));

  request.files.add(
    await http.MultipartFile.fromPath(
      'song',
      filePath,
      //contentType: MediaType('audio', 'midi'),
    ),
  );
  // The following line will enable the Android and iOS wakelock.
  Wakelock.enable();
  request.send().then((r) async {
    print(r.statusCode);
    // print(json.decode(await r.stream.transform(utf8.decoder).join()));
    if (r.statusCode == 200) {
      Directory tempDir = await getTemporaryDirectory();
      String tempPath = tempDir.path + "/test.wav";

      File f = new File(tempPath);
      await f.writeAsBytes(await r.stream.toBytes());
      print(f.path);
      print(f.length());
      setState(() {
        loading=false;
      });
      // The next line disables the wakelock again.
      Wakelock.disable();
      Navigator.of(context).push(MaterialPageRoute(builder: (context) => PlayerPage(title: tempPath)));
```

```
StreamBuilder<PlayerState>(
  stream: player.playerStateStream,
  builder: (context, snapshot) {
    final playerState = snapshot.data;
    final processingState = playerState?.processingState;
    final playing = playerState?.playing;
    if (processingState == ProcessingState.loading ||
        processingState == ProcessingState.buffering) {
      return Container(
        margin: EdgeInsets.all(8.0),
        width: 64.0,
        height: 64.0,
        child: CircularProgressIndicator(),
      );
    } else if (playing != true) {
      return IconButton(
        icon: Icon(Icons.play_arrow),
        iconSize: 64.0,
        onPressed: player.play,
      );
    } else if (processingState != ProcessingState.completed) {
      return IconButton(
        icon: Icon(Icons.pause),
        iconSize: 64.0,
        onPressed: player.pause,
      );
    } else {
      return IconButton(
        icon: Icon(Icons.replay),
        iconSize: 64.0,
        onPressed: () => player.seek(Duration.zero),
      );
    }
  },
),
```

```
def generate(song_path, output_filename):
  song_notes = extract_notes(song_path)
  """ Generate a piano midi file """
  #load the notes used to train the model
  with open('/content/Classical-Piano-Composer/data/notes', 'rb') as filepath:
      notes = pickle.load(filepath)

  # Get all pitch names
  pitchnames = sorted(set(item for item in notes))
  # Get all pitch names
  n_vocab = len(set(notes))

  network_input, normalized_input = prepare_sequences(song_notes, pitchnames, n_vocab)
  model = create_network(normalized_input, n_vocab)
  prediction_output = generate_notes(model, network_input, pitchnames, n_vocab)
  create_midi(prediction_output,output_filename)
```

```
def create_network(network_input, n_vocab):
  """ create the structure of the neural network """
  model = Sequential()
  model.add(LSTM(
      512,
      input_shape=(network_input.shape[1], network_input.shape[2]),
      recurrent_dropout=0.3,
      return_sequences=True
  ))
  model.add(LSTM(512, return_sequences=True, recurrent_dropout=0.3,))
  model.add(LSTM(512))
  model.add(BatchNorm())
  model.add(Dropout(0.3))
  model.add(Dense(256))
  model.add(Activation('relu'))
  model.add(BatchNorm())
  model.add(Dropout(0.3))
  model.add(Dense(n_vocab))
  model.add(Activation('softmax'))
  model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

  # Load the weights to each node
  model.load_weights('/content/Classical-Piano-Composer/weights.hdf5')

  return model
```

```python
def generate_notes(model, network_input, pitchnames, n_vocab):
  """ Generate notes from the neural network based on a sequence of notes """
  # pick a random sequence from the input as a starting point for the prediction
  start = numpy.random.randint(0, len(network_input)-1)


  int_to_note = dict((number, note) for number, note in enumerate(pitchnames))

  pattern = network_input[start]
  # print(type(pattern))

  prediction_output = []

  # generate 250 notes. Generate a 30 sec songs
  for note_index in range(250):
      prediction_input = numpy.reshape(pattern, (1, len(pattern), 1))
      prediction_input = prediction_input / float(n_vocab)

      prediction = model.predict(prediction_input, verbose=0)

      index = numpy.argmax(prediction)
      result = int_to_note[index]
      prediction_output.append(result)

      pattern.append(index)
      pattern = pattern[1:len(pattern)]

  return prediction_output
```

```python
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import ast
nltk.download('vader_lexicon')

def nltkSentiment(sentence):
  #NLTK anaylzer provides 4 datapoints in a dictionary --> compound datapoint is used;
  anaylzer = SentimentIntensityAnalyzer()
  return anaylzer.polarity_scores(sentence)['compound']

print(nltkSentiment("Sentiment analysis has not been good."))
```

Figure 2. Code of MFly

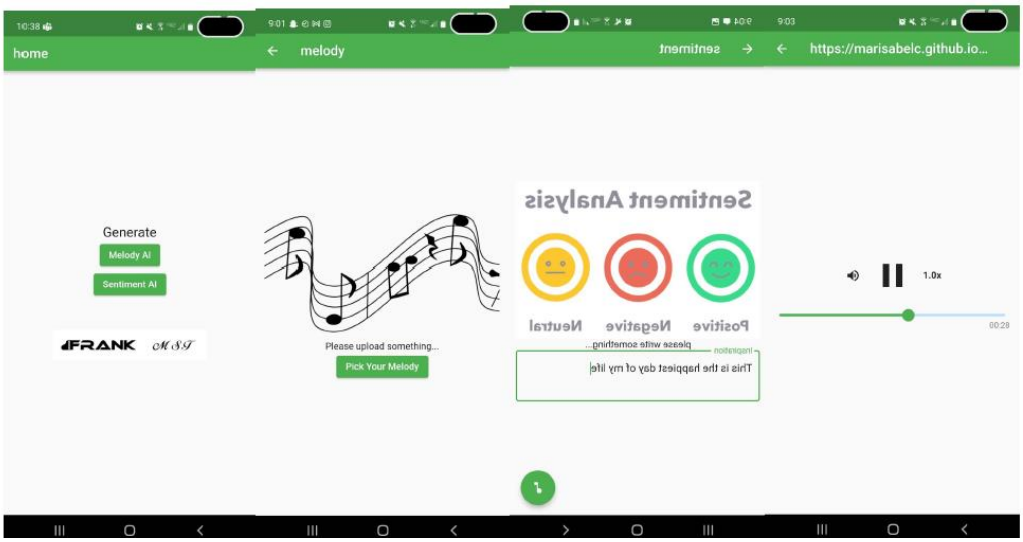

Figure 3. Screenshots of MFly

4. EXPERIMENT

4.1. Experiment 1

MFly resolves the issues of struggling to obtain music that suits people's needs in terms of atmosphere and mood. In order to test how effective the system of generating music is at

accounting for the intent of the user when using a message from the user as input, twenty-three participants downloaded the mobile application on their mobile phones, then tested this implementation by going into the sentiment page of the application, typing in a message, and recording whether the music that was outputted matches the sentiment of the message. The participants would perform this process two times, one for each possible sentiment (positive and negative).

Participant #	Positive Sentiment Accurately Portrayed	Negative Sentiment Accurately Portrayed
1	Yes	Yes
2	Yes	Yes
3	No	No
4	Yes	No
5	Yes	Yes
6	No	Yes
7	Yes	Yes
8	Yes	Yes
9	Yes	Yes
10	Yes	No
11	Yes	No
12	Yes	Yes
13	Yes	Yes
14	Yes	Yes
15	Yes	Yes
16	Yes	Yes
17	Yes	No
18	Yes	No
19	Yes	Yes
20	Yes	Yes
21	Yes	Yes
22	Yes	Yes
23	Yes	No
Accuracy	21/23	16/23

Figure 4. Table of possible sentiment

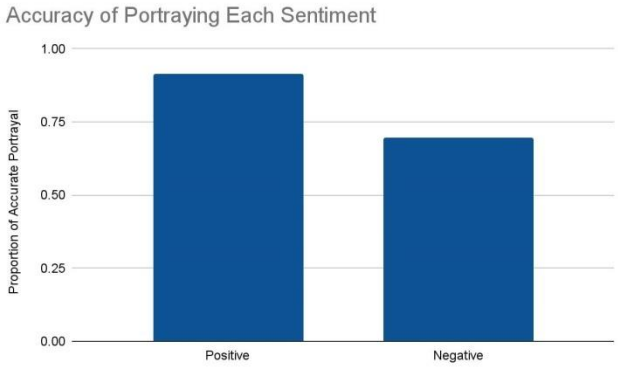

Figure 5. Accuracy of portraying each sentiment

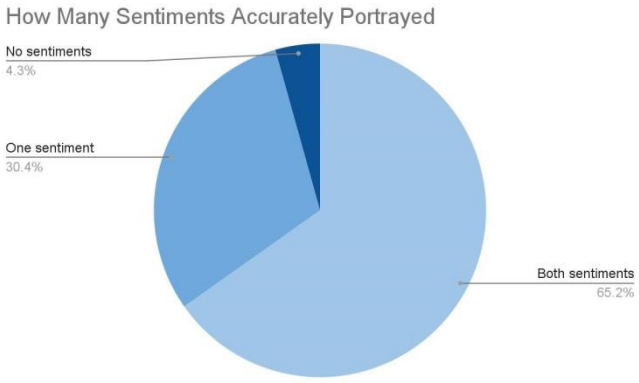

Figure 6. Chart of sentiments accurately portrayed

According to the charts and graphs, it appears that the accuracy of properly portraying the sentiment of a message is highest in messages that have a positive sentiment, as 21 out of the 23 participants recorded that their inputted message with a positive sentiment resulted in a musical piece that they believed to fit a positive atmosphere. On the other hand, the lowest accuracy comes from messages with a negative sentiment, with only 16 out of 23 participants indicating that their negative message returned appropriately fitting music. A possible explanation for these results is that the AI that is being used to select the music is not well-trained enough in sentiment analysis to choose the correct song. As for each participant, approximately 65.2% of the participants believed that their positive and negative messages both returned appropriately fitting music. Therefore, 34.8% of the participants found that at least one of the music outputs did not match the corresponding original message's sentiment.

4.2. Experiment 2

This application also seeks to provide convenience for those who wish to quickly have access to freshly created music. After the previous experiment is finished, the participants will take a postexperiment survey that gauges how convenient using the application was and whether they would use it again in the future. These values will be asked in two separate questions, "How convenient is the application to use?" and "How practical is using this application in daily life?", and the participants will have the option to answer them using a scale from one to ten. An optional free-response question for any additional feedback will also be provided to the users at

the end of the survey. As the survey features the same twenty-three participants from the other experiment, there is more than enough of a sample size to account for variability.

Participant #	Convenience of the Application	Practicality of the Application
1	9	6
2	8	7
3	10	5
4	8	5
5	7	9
6	9	6
7	8	8
8	10	8
9	7	6
10	8	5
11	6	4
12	9	7
13	7	7
14	8	6
15	7	8
16	8	7
17	8	6
18	9	5
19	6	5
20	7	7
21	8	7
22	9	8
23	9	7
Average	8.0435	6.4783

Figure 7. Table of convenience and practicality of the application

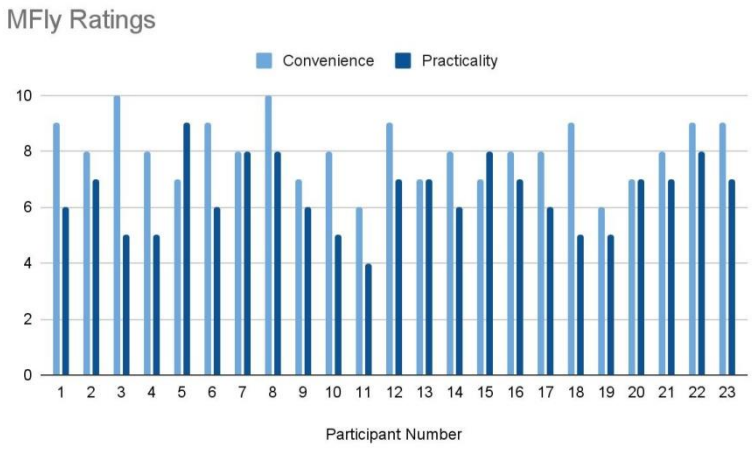

Figure 8. Graph of MFly ratings

Generally, both the convenience and the practicality of the application were rated somewhat highly. A pattern that can be seen among the participants is that they generally seem to rate the convenience of using the application higher than the practicality of the application. The difference between the two values being rated can be emphasized by comparing the average ratings, in which the average convenience rating of approximately 8.0435 is significantly higher

than the average of approximately 6.4783. The primary criticism from the participants 'feedback was that there was no way within the application to easily share the AI-generated music to other platforms or social media. Another recurring point of feedback was that the application could include more features to make quality-of-life improvements for the users by adding features. However, participants also noted that the simple look and the clearly labeled buttons made navigation of the application very easy.

According to the experiment that involves testing the success of the application at accurately reflecting the sentiment of the inputted message in the outputted music, a decent portion of the participants believed that there was at least one instance in which the sentiment from the message did not align with the tone of the music and the feelings and emotions that the music gave off. These results indicate that although the application can generate music with a specific sentiment to some degree, there is still work to be done when it comes to improving the accuracy of portraying the sentiment of a message in the outputted music properly.

From the post-experiment survey, it appears that while the application is convenient and simple to use for the general public, the application can still use some room for improvement when it comes to its practicality. The primary complaints based on the participants' feedback are that there is no way to share the audio outside the application and there is not enough features in the application, which indicates that the most urgent changes to make to the application should attempt to create more features to improve user experience, especially for exporting the audio from the application.

5. RELATED WORK

Ramon Lopez de Mantaras and Josep Lluis Arcos from the Spanish National Research Council surveyed computer music systems that used artificial intelligence techniques focusing on compositional systems, improvisation systems, and performance systems. Case-based reasoning has been shown to be an incredibly useful technique for composing music with artificial intelligence because it has the capability to directly utilize extracted knowledge from recorded human performance examples in an implicit manner [2]. While de Mantaras and Arcos thoroughly describe and analyze the AI techniques to generate music and explore the challenge behind giving these generated pieces of music the expressiveness of human-created music, this work aims to develop a mobile application that generates music based on inputted audio or message and evaluate its effectiveness at doing so.

Ryan Louie from Northwestern University teamed up with researchers from Google Brain to produce a study on developing tools to steer artificial intelligence so that music composers would be able to work alongside the AI in a clear and coherent manner. Some of these tools included directing the voices towards a specific range or adjusting certain values based on sliders. Results indicated that these tools were effective at improving the trust, understanding, and sense of control and productiveness of users [3]. Both the work of Louie et al. and this work focus on developing an interface for people to conveniently generate music to their liking; however, MFly focuses much more on using sentiment as a whole for the input for music generation than the interface created by Louie et al. does.

Emma Frid from the KTH Royal Institute of Technology collaborated with Celso Gomes and Zeyu Jin from Adobe Research to conduct studies on generating music with artificial intelligence by inputting an example reference of a song and mixing AI-generated parts of the song. In recent times, creating videos for social media platforms and finding suitable, copyright-free music to accompany them can be a difficult challenge, and online content creators for these platforms have indicated that composing music in conjunction with the AI is preferable [4]. The work of

Frid et al. is similar to this work in that an AI and a person both have a part in the creation of the final musical piece. Frid et al. focused on more complex ways to customize the music, such as mixing from a selection of generated melody and bass tracks, while this work simply used the sentiment of an inputted message to create the music.

6. CONCLUSIONS

The mobile application MFly was designed to generate new musical pieces using artificial intelligence [14]. The two implemented methods of music generation that the user can choose from are inputting reference audio or inputting a message. By taking in reference audio as input, the artificial intelligence would attempt to emulate the audio's musical style while generating the music. By taking in a message as input, the application would extract the sentiment of the message and use the sentiment to influence the generation of the music. The goal of the application is to become a convenient and flexible source of music for its users.

In order to test the effectiveness of the application, an experiment was conducted with the method of music generation that involves an inputted message. Twenty-three participants were asked to input one message with a positive sentiment and one message with a negative sentiment. Then, they would record whether they believed each generated piece of music accurately portrayed the sentiment of the corresponding message. The results of this experiment were that the application was the most accurate when generating music with a positive sentiment and the least accurate when generating music with a negative sentiment. A post-experiment survey was also conducted using the same participants, in which each participant would rate the convenience and practicality of the application from 1 to 10. Survey results indicated that the application was very simple to use and was fairly practical in daily usage. However, the participants also mentioned that the addition of a feature to share or export the generated music in the application would greatly enhance the user experience.

One of the most limiting features of the MFly mobile application is the ability to save the music that is created. Once the music is generated by artificial intelligence, the application moves to a screen in which the generated audio is played. However, there is no current way to save the without the use of external tools, such as a mobile screen recorder that captures the audio playing on the device. Without such a feature, the application would be much less practical for the general public, as the audio would not be easily transferable outside of the application for other means.

To tackle this limitation in the future, something that could be done is updating the interface to create a download button. Once pressed, this download button will save the AI-generated music onto the device, and the user will be free to share it online in various social media platforms. This will greatly improve the quality of life for MFly's users.

REFERENCES

[1] How does music affect the mind and body? Medford Leas. (2022, July 19). Retrieved August 30, 2022, from https://medfordleas.org/how-does-music-affect-the-mind-and-body/

[2] Frid, E., Gomes, C., & Jin, Z. (2020). Music Creation by Example. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 1 – 13. https://doi.org/10.1145/3313831.3376514

[3] Louie, R., Coenen, A., Huang, C. Z., Terry, M., & Cai, C. J. (2020). Novice-ai music co-creation via AI-steering tools for deep generative models. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 1 – 13. https://doi.org/10.1145/3313831.3376739

[4] de Mantaras, R. L., & Arcos, J. L. (2002). AI and Music: From Composition to Expressive Performance.AI Magazine, 23(3). https://doi.org/https://doi.org/10.1609/aimag.v23i3.1656
[5] Skuldur. (2019). Skuldur/classical-piano-composer. GitHub. Retrieved August 30, 2022, from https://github.com/Skuldur/Classical-Piano-Composer
[6] Collins, Karen, ed. From Pac-Man to pop music: interactive audio in games and new media. Routledge, 2017.
[7] Bown, Oliver Roland, and Aengus Martin. "Autonomy in music-generating systems." Eighth Artificial Intelligence and Interactive Digital Entertainment Conference. 2012.
[8] Juslin, Patrik N., László Harmat, and Tuomas Eerola. "What makes music emotionally significant? Exploring the underlying mechanisms." Psychology of Music 42.4 (2014): 599-623.
[9] Donnelly, Kevin J., William Gibbons, and Neil Lerner. "Music in Video Games." All Your Bass are Belong to Us (= Routledge Music and Screen Media Series), New York (2014).
[10] North, Adrian C., David J. Hargreaves, and Jon J. Hargreaves. "Uses of music in everyday life." Music perception 22.1 (2004): 41-77.
[11] Weisser, Martin. "DART – The dialogue annotation and research tool." Corpus Linguistics and Linguistic Theory 12.2 (2016): 355-388.
[12] Cambouropoulos, Emilios. "From MIDI to traditional musical notation." Proceedings of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Models for Composition, Performance and Analysis. Vol. 30. 2000.
[13] Xue, Hao, Du Q. Huynh, and Mark Reynolds. "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction." 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2018.
[14] Dubnov, Shlomo, et al. "Using machine-learning methods for musical style modeling." Computer 36.10 (2003): 73-80.
[15] Fetzer, James H. "What is Artificial Intelligence?." Artificial Intelligence: Its Scope and Limits. Springer, Dordrecht, 1990. 3-27.