Faculty of Engineering
Alexandria University

Generating Music Based on a Given Sentiment Using Deep Learning

Authors :

Muhammad Salah
Mohamed Abdelhakem
Abdelrahman Kamal
Abdelfattah Mohamed
Amr Khaled

Supervisors :

Prof. Dr. Nagwa El-Makky
Prof. Dr. Khaled Nagi

Graduation Project submitted in fulfilment of the requirements for the degree of BSc.

in the

Computer and Systems Engineering Department

June 2021

Abstract

In the field of music composition, sentiment is widely acknowledged as a key parameter for creativity assessments and music usability. While deep learning models have achieved compelling results in the task of music generation, it's very hard to condition such deep learning models to generate music with a given sentiment. We aim to build a deep learning music composition system that can be conditioned to generate music with sentiment.

This work provides a complete usable deep learning system that can be directed to compose music with a given sentiment from scratch or from a given primer. In addition to the generative model, our system can also be used for sentiment analysis of symbolic MIDI music.

We evaluate the sentiment of the generated music using our sentiment classifier, and using a subjective evaluation survey. We also use an objective evaluation system, based on musical features to evaluate the quality of the music generated. The results show that the model can generate high quality music that captures the given sentiment with high probability.

Acknowledgments

We would like to express our sincere gratitude to Prof. Dr. Nagwa El-Makky and Prof. Dr. Khaled Nagi for their sage supervision, patient guidance and convivial support. We also want to express our thanks to the Information Technology Industry Development Agency (ITIDA) for sponsoring this work.

We also recognize the efforts of our colleagues who participated in the subjective evaluation survey

Contents

List of Figures

List of Tables

List of Acronyms

ADL	Augmented Design Lab	31
BERT	Bidirectional Encoder Representations from Transformers	28
CNNs	Convolutional Neural Networks	14
CTRL	Conditional Transformer Language Model	84
GANs	Generative adversarial networks	11
GPT-2	Generative Pre-trained Transformer -2	19
IOI	Average Inter-Onset-Interval	73
LM	Language Model	35
LSTMs	Long-short term memory network	21

MAESTRO	MIDI and Audio Edited for Synchronous TRacks and Organization	18
MD5	message-digest checksum	31
MIDI	Musical Instrument Digital Interface	9
NC	Note Count	73
NLH	Note Length Histogram	73
NLTM	Note Length Transition Matrix	73
PC	Pitch Count	72
PCH	Pitch Class Histogram	72
PCTM	Pitch Class Transition Matrix	73
PI	Average Pitch Interval	73
PR	Pitch Range	73

REST API	Representational state transfer Application Programming Interface	62
RNNs	Recurrent Neural Networks	14
SBBS	Stochastic Bi-Objective Beam Search	28
VGMIDI	Video Game Musical Instrument Digital Interface	26

Chapter 1

Introduction

In this chapter, we discuss the motivation and scope of our work and present a general overview of the report. section 1.1 introduces the problem of music generation with sentiment, with a brief summary of previous related work that will be later discussed in detail in Chapter 2, and discusses our motivation for this work. section 1.2 exhibits our goal and the scope of this work. In section 1.3, we outline the organization of the rest of the report.

1.1 Problem And Motivation

Writing music needs creativity, and delivering a specific emotion using music needs a higher level of creativity. Music is considered one of the most common communication languages, while some people understand a specific language like Arabic or English, almost all people understand music, listen to music, and feel the music.

While computers can be used to solve systematic problems, some researchers are trying to make computers creative, teaching them how to mimic human intelligence, and how to generate human-like contexts.

The problem of music generation using computers is a very important problem in the field of artificial intelligence (AI). Recently, Deep Learning Models and techniques showed very impressive results in the task of content generation, including high quality music generation. Computers now can generate music that humans will love to listen to, or even generate music in the style of a specific composer or a specific music genre.

With the recent improvements in computer generated music, we think it's time to start teaching computers to deliver a feeling, emotion, or a sentiment along with the music generated. Although it's easy for computers to learn the complex structure of music as a language using deep learning from the large amounts of musical data available in the cloud, it's still a hard problem to teach computers to understand the sentiment of a music piece and generate a music that delivers a specific emotion to the listener.

The very first entry barrier to this problem is the definition of sentiment in music, it's very hard to quantitate sentiment, or even to understand what sentiment means and how can some sequence of notes deliver a specific sentiment to the listener and another sequence deliver another sentiment. It is even subjective, one listener can feel something while listening to a piece of music and another listener might feel something else.

With this subjectivity in the analysis of sentiment in music, the lack of data appears as a more clear and bigger entry barrier to the problem of

generating music with sentiment, This is because language models, in general, need large amount of data for unsupervised training, i.e., they don’t need labeled data for training.

One more important limitation is the lack of quantitative evaluation metrics when it comes to music in general, how would you know that one music sequence is better than another, and this limitation also happens when it comes to sentiment too, it’s important to ask how can we compare between two computer systems that generate music, and how can we evaluate the sentiment in the music generated by a computer system?

The motivation to build a system that generates music based on a given sentiment, is derived by the importance of such a system. Music is a universal language, and a lot of things and activities in our world depend on a good music composition such as movies, theatres, video games, songs, and more.

The existence of a usable system that can be used to generate music based on the given sentiment can be beneficial to composers as an inspirational tool, to movie and video makers to add the proper background music to their productions, to video games companies to help them develop a proper immersion with the right music, and to many industries that rely on music.

1.2 Goal And Scope

In this work, our aim is to get a step closer to human level intelligence, by modeling one of the most difficult aspects of human intelligence which is the emotions embedded in the music. In this work we purpose a complete music generation system that generates music based on a desired sentiment or emotion. This system can be of use to help composers to deliver the sentiment they seek to compose for. In the future the system can be developed to compose for movies or video games the music that better suits the scene and the emotions attached to it.

Due to the lack of large sentiment-labeled musical datasets, the previous work in music generation based on a given sentiment has never been able to make use of large language models such as transformers which require a large amount of data to be efficient. In this work we propose an approach to generate symbolic music based on a given emotion represented by the valence-arousal model, using a Stochastic Beam Search Algorithm and transfer learning applied to a pre-trained music emotion classifier and a pre-trained music transformer.

1.3 Report Organization

The rest of this report is organized in the following way. Chapter 2 provides a detailed literature review and related work studies. Chapter 3 describes the baseline model, the bardo composer , and its components in detail. Chapter 4 discusses the implementation details of the system, the different components, and the interactions

between them. In Chapter 5, the various experiments adopted to evaluate our work, along with a detailed analysis based on the obtained results, are given. And finally, in Chapter 6, we conclude the report and point out some directions to possible future work.

Chapter 2

Background And Literature Review

2.1 Introduction

In this chapter, we briefly review background and some work related to the generation of music generally and the generation of music based on sentiment specifically. In section 2.2, We will discuss some background aspects of music generations. Since we reviewed so many papers and related work at the beginning, we will only list the closest ones to our work. In section 2.3 we will show some of the related work in the music generation field . Then , in section 2.4 we will further discuss the related work on music generation based on sentiment.

2.2 Background

2.2.1 Music Modeling Types

Music generation has been widely modeled in the symbolic domain like music score, MIDI sequences or a sequence of chords. This domain provides a more abstract way of music representation which makes modelling the problem tractable and has less computational requirements. Piano rolls are one of popular representations of symbolic domain, as they capture timing, pitch and velocity. Although piano rolls cover all degrees of freedom that a performer can control, most other instruments have many more degrees of freedom. Therefore, having a high-level representation, that can accurately capture all information in other instruments, becomes more challenging. Audio waveform models can provide more bit rate and so more degrees of freedom, but it comes with the cost of higher computational requirements for training. Spectrograms can be used as a digital representation of audio waveforms. Waveform modelling is

more common in the generative setting, as spectrograms are commonly used by discriminative models of audio.

2.2.2 Music Generative Models

Generative models can learn to approximate the underlying distribution $P_X(x)$of a given dataset X of examples $x \in X$. These models are capable of generating new samples that look like those of the dataset and they have two types: implicit and explicit models. Implicit models can generate new samples x where $x \sim P_X(x)$, but cannot calculate the likelihood of the new sample. However, explicit models can do so. We will discuss different types of generative models that target our problem:

1. Conditional generative models: we can provide some influence over the generated sample through a conditioning signal c, where the model fit to conditional distribution of $P_X(x|c)$ instead of $P_X(x)$. We could condition music generation on sentiment, composer, instrument, tempo or timbre.

2. Likelihood based models: they seek to optimize its parameters to maximize the likelihood of the data under the model. The model parameterize $P_X(x)$, so we can infer the likelihood of generative models. WaveNet [1], SampleRNN [2], parallel WaveNet [3] and Wave2Midi2Wave [4] are good examples of Likelihood based models.
3. Adversarial models: Generative adversarial models (GANs) [5] has 2 networks that are trained simultaneously through two-player minimax game: a Generator G tries to produce examples that match the data distribution $P_X(x)$given latent vectors z , while the discriminator tries to tell apart generated and real examples. Examples of Music GANs are: MuseGAN and MidiNet, we will discuss them further in section 2.3.

Magenta's Music transformer [6] and OpenAI's MuseNet [7] show impressive results in the music generation domain and both of them belong to the likelihood-based language models.

2.2.3 A Circumplex Model of Affect [8]

Russel et al. [8] Discuss the way emotions can be modeled quantitatively. Before, the emotions were modeled in six to twelve monopolar factors of affect independent of one another such as a degree of sadness, anxiety, anger, etc. However the emotions are not independent of one another, but related in a highly systematic fashion. Emotions can be modeled and organized in a circular arrangement which means 2 bipolar dimensions instead of six to twelve monopolar degrees; these 2 dimensions are valence and arousal . Valence-Arousal Model of Affect : Emotions categories lay on the circumference of a circle where there is a 180 degree between each two opposite emotions. We would say that at degree 0 there's pleasure while displeasure is at degree 180, this makes excitement at 45, arousal at 90, distress 135, depression 225, and sleepiness at 270.

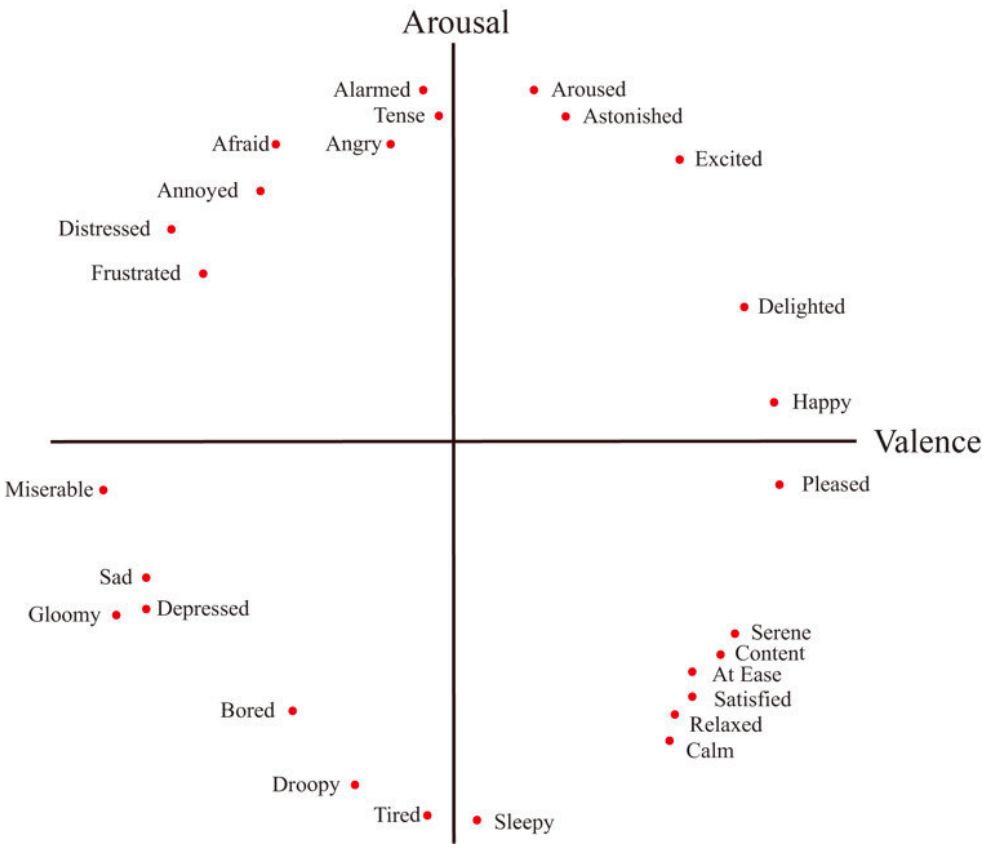

Figure 2.1: Russell's emotion model

2.3 Music Generation Using Deep Learning

2.3.1 MidiNET: A Convolutional Generative Adversarial Network For Symbolic-Domain Music Generation [9]

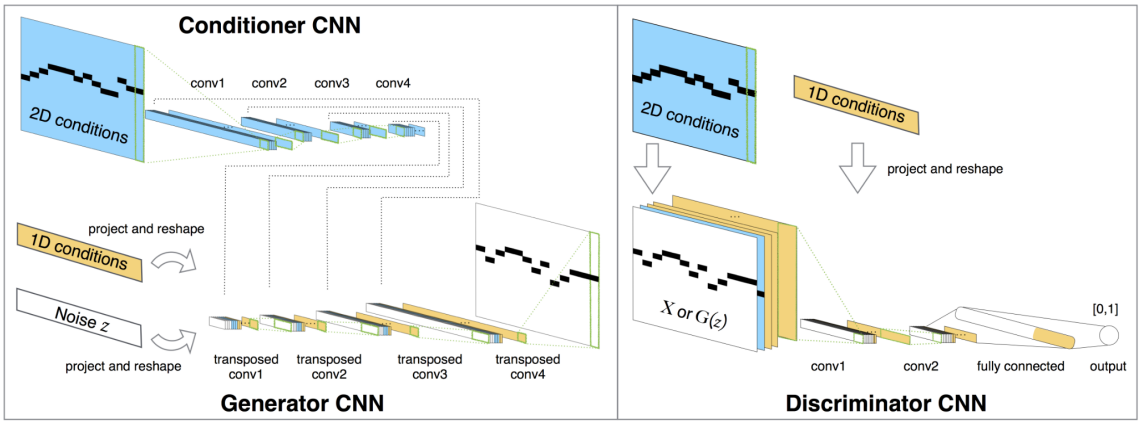

Figure 2.2: MidiNET Architecture

Most of the music generation tasks are conducted by Recurrent Neural Networks (RNNs) before the transformer-based models appeared, but also Convolutional Neural Networks (CNNs) can be used in music generation. This paper benefits from CNN's ability to be parallelizable in training. The authors introduce a CNN based model that generates music in the symbolic domain, beside the Generator, they use a Discriminator to learn the distributions of the music under Generative adversarial networks (GANs) [5]. They propose a novel conditioning mechanism to generate melodies from scratch or by using primer melody sequence, conditioning melody. They refer to this model as MidiNET.

The model applies convolution on a 2-D matrix, where one dimension represents timesteps and the other represents the bar notes mask. Transposed Convolution is used in the CNN generator to perform transformation of randomness to a 2-D matrix that appears to be real to the Discriminator network. This is done under GAN minimax training procedure. To capture the temporal behaviour in the music, A Conditioner CNN is used to ingest previous melody feature representation to their corresponding ones in the Generator CNN so as to mimic the RNN behaviour of learning temporal information.

This model could generate music from scratch or from a conditioning matrix that can be fed into Conditioner CNN, even with the same conditioning input, the model is capable of generating new music every time, as a result of randomness fed to Generator CNN. Therefore, the authors could guide their model through conditioning to produce music of their interest. Also, this model is extensible to multi-channel tensors producing multi-track output.

2.3.2 MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment [10]

This paper aims to generate multi-track polyphonic music with harmonic and rhythmic structure and temporal structure. Music can be created either by improvisation or A composer arranges instruments, So, the authors of this paper introduce three Generative Adversarial Network (GAN) models with different architectures and input: the jamming, the composer and the hybrid model. The model can learn to generate music from scratch or given an input track that is accompanied by other tracks.

The jamming model is composed of Multiple generators and discriminators, each with its corresponding input causing independent music tracks generation. The composer model has only one generator with one input vector and discriminator which provides the generator with critics, while the generator outputs multi-track output. The

Hybrid model combined the two aforementioned models with multiple generators and one discriminator providing more inter-track dependency. Also they target the temporal structure through a GAN model that learns to generate latent vectors that can be provided as input to either of the three models. Finally the whole model presented, named MuseGAN, combines the temporal GAN and any of the 3 models. They used a piano-roll dataset combining the Lakh MIDI dataset with the Million Song dataset. An objective Evaluation is conducted using different metrics like ratio of empty bars, tonal distance (measuring the harmonicity between track pairs).

2.3.3 Music Transformer: Generating Music With Long-Term Structure [6]

This paper presents procedures to generate long pieces of music, by showing Music Transformer [6], an attention-based neural network that can generate music with improved long-term coherence.

Instinctively, self-attention [11] can be a great coordinate for this assignment over recurrent neural networks since Self-attention permits an autoregressive model to get to any portion of the already generated output at each step of generation.

Music has different measurements in which relative contrasts matter more than absolute values, the two most critical are timing and pitch. The authors extend a relation-aware version of the self-attention [11] approach that significantly decreases the memory footprint, allowing them to scale melodic sequences on the order of minutes, to capture relative timing and optionally also pitch, which yields an improvement in both sample quality and perplexity for the JSB Chorales dataset.

Finally, the authors evaluate the transformer with their relative attention mechanism on two datasets, JSB Chorales and Maestro, and get state-of-the-art results on the last mentioned.

2.3.4 MuseNet [7]

OpenAI proposes a deep neural network, MuseNet, that can generate up to 4 minutes of music composition with 10 different instruments. It can also combine different styles of famous composers. MuseNet uses deep Learning to learn the music patterns of harmony and rhythm and style in a large number of MIDI files. It uses GPT-2 as a language model, which is a large scale transformer trained to predict the most probable next token in music sequence generation.

MuseNet uses tokens of composer or style and instruments to control the generated music samples. These tokens are accompanied to each sample while training, so that the model could learn to use this conditioning information to bias the generation. During generation time, users can input this conditioning information to the model to generate samples that match that condition. To capture long-term structures, MuseNet uses the recompute and optimized kernels of Sparse Transformer [12]. Musenet was trained on dataset collected

from different sources like ClassicalArchives [13] and BitMidi [14] and MAESTRO dataset [15].

2.3.5 Conditioning Deep Generative Raw Audio Models for Structured Automatic Music [16]

Music generation with deep learning can be classified into two approaches: raw audio models and symbolic models. Symbolic models, that train and generate music at the note level, are most common and can capture long-term dependencies in the music structure but lack realism of audio generation. However, raw audio models, such as DeepMind's WaveNet[1], are trained on sample audio waveforms enabling them to generate more realistic music but unstructured. This paper introduces a combination of the two approaches producing realistic and structured music that capture long-range dependencies.

The authors use biaxial Long-short term memory network (LSTMs) to model symbolic melody. The term biaxial refers to leveraging LSTMs in two dimensions: the temporal dimension and the pitch dimension at each timestep. These biaxial LSTMs are trained on MIDI files of a specific genre of music. The generations from this model are fed as a second time series conditioning signal within the raw audio generator (WaveNet).

The model was trained on MusicNet dataset which contains symbolic melodies aligned with raw audio at each timestamp. This model is evaluated quantitatively by comparing loss functions of conditioned and unconditioned WaveNet (without second time-series input). First, unconditioned WaveNet was fine tuned for unstructured music generation and it was found that it is capable of capturing short-term dependencies but lacks the long-term ones and produce unstructured music. Second, conditioned WaveNet was evaluated by ear on famous melodies and generated audio was found to follow the conditioning input. Also, it was evaluated on other LSTMs generations by cross-correlation where this evaluation emphasizes capturing structured long-term as well as

short-term dependencies in the generated music. Finally, this model was capable of editing raw audio by making small changes to the MIDI input.

2.3.6 Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset [15]

This paper proposes Wave2Midi2Wave model which is capable of transcribing, composing and synthesizing audio waveforms along the model pipeline using their new release of MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) dataset. This dataset is composed of 172 hours of piano performances captured with fine alignment between note labels and audio waveforms. The WaveNet model [1] was the first work for generating music in audio waveforms with neural nets. However, Wavenet lacked a conditioning signal for their model which results in audio capturing short-term dependencies, but has chaos in it. The main contributions by the authors are as follows:

1. They combine a transcription model, a language model and a MIDI-conditioned WaveNet model to produce a factorized approach for musical audio generation.
2. They introduced a new dataset, MAESTRO, of piano performance recordings aligned with MIDI transcription, this dataset is an order of magnitude larger than the previous benchmarks.
3. They trained existing transcription model architecture on MAESTRO and achieved state-of-the-art results on a piano transcription benchmark.

For piano transcription, Onsets and Frames [17] was used as a baseline model with several modifications informed by coarse hyperparameter search. The modified transcription model enables training language models on a larger set of unlabeled piano data. Therefore, it was used to build a transcripted version of the training set MAESTRO-T.

Music transformer was used as a language model and two models were trained, the first on MIDI from the MAESTRO [15] dataset and the other on the transcripted version MAESTRO-T.

For piano synthesis, WaveNet [1] is conditioned on MIDI sequence to capture long scale structure. Three models were trained:

1. Unconditioned: trained only on audio from the MAESTRO dataset.
2. Ground: trained on audio conditioned with MIDI sequence.
3. Transcripted: trained as Ground model but conditioned on transcripted MIDI produced from Onsets and Frames [17].

The authors use human listening tests to judge the model and draw samples from 5 different sets:

1. Clips selected randomly from ground truth MAESTRO dataset
2. Clips generated by unconditioned WaveNet model
3. Clips generated by ground truth WaveNet conditioned on MIDI sequences from MAESTRO test split.

4. Clips generated by the transcripted WaveNet conditioned on MIDI produced by the transcription model from MAESTRO test split.
5. Clips generated by the transcripted WaveNet conditioned on MIDI generated by Music transformer, this set makes an end-to-end test of the whole model.

2.4 Generating Music With Sentiment

2.4.1 SentiMozart: Music Generation Based On Emotions [18]

The aim of this framework is to extract the facial emotion from a picture then feed the corresponding dataset to a music generation model, so it generates music related to this dataset that matches the emotion. So the main idea is to change the dataset for each emotion. The proposed framework is divided into two models. First, the Image

Classification Model that performs the classification and the identification of the facial expression into one of seven major sentiment categories: Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral, using convolutional neural networks (CNNs). Second, the Music Generation Model, which is essentially a Doubly Stacked LSTM architecture that generates the music . There is an intermediate step after getting the emotion classification of the picture, where the authors use the corresponding emotion dataset to generate music based on this emotion.

2.4.2 Learning To Generate Music With Sentiment [19]

This work uses a method based on the work in [20] which generates product reviews (in textual form) with sentiment. The authors also created a dataset for midi music labeled with sentiment called VGMIDI and encoded the MIDI files in a text format to be able to apply the work in [20]. which needs to be applied in a text format. Using words like t_120 (tempo = 120), v_76 (Velocity = 76), etc., they

were able to encode a complete music sheet into text. This enabled them to capture the composition and also the performance notations of the music. The VGMIDI dataset is collected from video games soundtracks. The dataset consists of 823 pieces, from 26 seconds to 3 minutes of piano music per piece. 95 pieces are annotated according to a 2-dimensional model that represents emotion (valence-arousal). They explored the weights to find the set of neurons responsible for sentiment and applied a genetic algorithm to tune these weights according to the given sentiment. Besides music generation, the same model can be used for sentiment analysis of symbolic music.

2.4.3 Computer-Generated Music for Tabletop Role-Playing Games [21]

Since this is our baseline model we will further discuss it in detail in Chapter 3. The proposed bardo composer model operates on some text (sentences) from tabletop role-playing games and extracts its emotion using a text emotion classifier then generates music pieces that matches this emotion using a language model and a music emotion

classifier. The work in this paper is based on the Bardo system. Since it's challenging to generate music pieces with target emotion, the authors of this paper introduced Stochastic Bi-Objective Beam Search (SBBS), a variant of Stochastic Beam Search [22] to generate music pieces that match the desired emotion. They are using BERT [23] as a text emotion classifier and GPT-2 [24] as a language model and as a music emotion classifier.

2.5 Music Generation Datasets Study

2.5.1 MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization)

MAESTRO [15] is considered one of the largest piano datasets, it's composed of 200 hours of virtuosic piano performances including both note labels and audio waveform with fine alignment.

The dataset was collected during each installment of the International Piano-e-Competition [25], where pianists perform on a Yamaha Disklaviers Piano which has an integrated high-precision MIDI capture and playback system.

The dataset was collected over 10 years of the International Piano-e-Competition to contain 200 hours of paired audio and midi recordings with ~3 ms accuracy.

The midi data also includes the velocity and the sustain pedal positions, the meta-data of each piece includes the composer name, the title of the composition and the year of performance.

Magenta, which is a research project exploring the role of machine learning in the process of creating art and music, suggests a train/validation/test split to prevent the same piece that might be performed by multiple contestants from appearing in multiple subsets.

2.5.1.1 Dataset Statistics

The following table shows statistics of the MAESTRO dataset.

Split	Performances	Duration (hours)	Size (GB)	Notes (millions)
Train	962	159.2	96.3	5.66
Validation	137	19.4	11.8	0.64
Test	177	20.0	12.1	0.74
Total	1276	198.7	120.2	7.04

Table 2.1: MAESTRO dataset statistics

2.5.2 The Lakh MIDI Dataset [26]

The Lakh MIDI dataset [26] contains 176,581 unique MIDI files, 45,129 of them have been matched and aligned to entries in the Million Song Dataset [27]. The dataset was built for the purpose of large scale music information retrieval.

The MIDI files were scraped from publicly-available sources on the internet, and then de-duped according to their message-digest (MD5) checksum.

2.5.3 ADL Piano MIDI Dataset [21]

This one is a dataset containing 11,086 MIDI files from different genres based on the pieces from the Lakh MIDI dataset that have been matched with the Million Song Dataset. The tracks in each piece were filtered to contain tracks with instruments from the "Piano Family" (MIDI program numbers 1-8). These files are mainly Rock and Classical pieces. So, to increase the genre diversity (e.g. Jazz, Blues, and Latin) of the dataset, the authors included an additional 2,065 files scraped from public sources on the Internet2. All files in the final collection were de-duped according to their MD5 checksum.

2.5.4 VGMIDI Dataset

VGMIDI is a dataset composed of 823 pieces extracted from video game soundtracks in MIDI format.

The dataset contains only piano arrangements of the soundtracks and they vary in length from 26 seconds to 3 minutes.

The first version of the dataset contained 95 pieces that were annotated with sentiment by 30 human subjects according to a 2-dimensional model [8] that represents emotion using a valence-arousal pair. Then the dataset was extended in version 2 to contain 200 annotated pieces out of the 823 total pieces. To the best of our knowledge, VGMIDI is the only dataset that is annotated with sentiment.

The video game soundtracks were chosen because they are normally composed to keep the player in a certain emotion and thus they are less subjective pieces

Chapter 3

Baseline Model and Proposed System

3.1 Introduction

In this chapter, we further discuss the baseline model, the bardo composer [21], and the proposed system with its components in detail. In section 3.2, we briefly discuss the baseline model. In section 3.3, we discuss the architecture of the proposed model as a whole. In sections 3.4, 3.5 and 3.6, we illustrate each component: the music emotion classifiers, the language model and the Stochastic Multi-Objective Beam Search algorithm used, respectively. And finally the Music Synthesizer in section 3.7.

3.2 BaseLine Model [21]

The work in this model is based on the work in bardo system [28] which automatically selects the background music from a labeled library based on the story being told by the players. In this model, they extend it to generate emotion based music.

The proposed bardo composer model operates on some text (sentences) from tabletop role-playing games and extracts its emotion using a text emotion classifier then generates music pieces that match this emotion using a language model and a music emotion classifier. The authors are using BERT [23] as text emotion classifier and GPT-2 [24] as a language model and as a music emotion classifier. Both the music and text emotion classifiers consist of 2 separate classifiers, one for valence and the other one for arousal.

In order to have the same model of emotions between stories and emotions, the authors make a mapping from the bardo emotion model (4 classes) to the valence-arousal (v-a) model, that is (v = 0, a = 0) for

Suspenseful, (v = 1, a = 1) for Happy, (v = 0, a = 1) for Agitated and (v = 1, a = 0) for Calm.

The authors chose a pre-trained BERT model with a classification head added on top of it as well as fine-tuning all the parameters using the Call of the Wild dataset.

As there was no publicly available high-capacity LM pretrained with large (general) datasets in the symbolic music generation domain, the authors pre-trained a general high-capacity GPT-2 architecture as a language model using a new dataset they created called ADL (Augmented Design Lab) Piano MIDI dataset [21].

They used a similar approach to the story emotion with the music emotion. However, the authors fine-tuned the pretrained GPT-2 instead of the BERT, they explain; because it is better suited for sequence generation. An extended version of the VGMIDI dataset was used in the end-to-end fine-tuning of all the parameters.

Since it’s challenging to generate music pieces with target emotion, the authors introduced Stochastic Bi-Objective Beam Search (SBBS), a variant of Stochastic Beam Search [22] to generate music pieces that match the desired emotion.

3.3 Architecture Of The Proposed System

In this section we describe the architecture of the proposed system as a whole, inputs, outputs and the sequence or the flow of operation. Figure 3.1 shows a block diagram of the system as a whole.

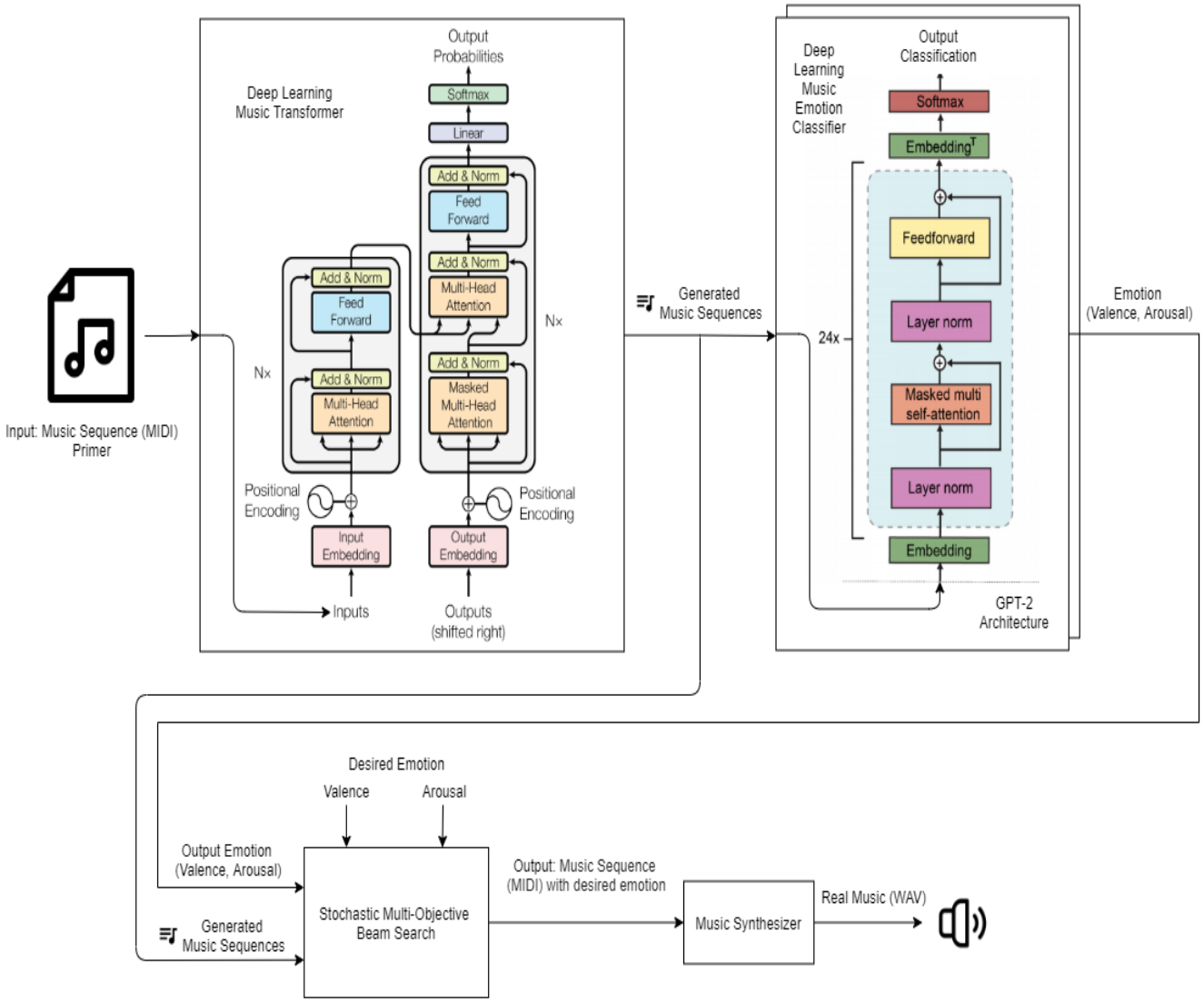

Figure 3.1:The block diagram of the proposed system.

First, the input is the initial tokens that can be treated as the primer sequence for the generation process, in other words, the generated sequence will try to match the primer sequence. The generation could

be from scratch if we didn't provide the initial tokens. As it can be seen, we ignored the text part of the bardo composer system because it's out of our interest, we focused more on the music part. As a result, we also provide as inputs the target emotion (valence and arousal) and the duration of the music sequence to be generated. These were extracted from the text sentences in the bardo composer.

The language model in our system is the music transformer, instead of the GPT-2 in the original model, the input to the music transformer is the current sequence after the previous step and the output is the next token or sequence of next tokens and their corresponding scores, that is the maximum likelihood of the output, this will be described in more details in section 3.5.

The music emotion classifier consists of 2 classifiers, one for valence and the other one for arousal, it takes as input the current music sequence after adding the next tokens from the language model, and its output is converted to a probability that describes how likely this

sequence matches the target emotion. The emotion classifier will be described in detail in section 3.4.

Each sequence from the previous beam will be expanded to k new sequences, where k is the beam size, by passing each one of the current sequences as an input to the transformer and returning as output the k next tokens with the corresponding scores, we then concatenate each next token with the old sequence to get the new sequence with its score, we then pass each one of the new sequences to the music emotion classifier as input and return as output the probability of this sequence having the same emotion as the target emotion.

Then, the beam search algorithm takes all the new sequences each one with its 2 probabilities and chooses the new beam of sequences to continue with to the next step based on some criteria, the chosen new sequences are the old sequences for the next step and it repeats, at the final step, only one sequence will be chosen and it will be the output

of the beam search, This is the generated sequence. The beam search algorithm and its criteria will be illustrated in detail in section 3.6.

The midi file representing the generated sequence is then passed to the music synthesizer, described in section 3.7, to be transformed to audio file, a file with .wav extension, and this is the final output of the system, which is the music piece generated based on the desired sentiment.

3.4 Music Emotion Classifier

We used the same music emotion classifier used in the bardo composer model. In this model, the authors used a transfer learning approach due to the limited amount of MIDI pieces labeled according to emotion.

The GPT-2 Language Model has 4 layers (transformer blocks), context size of 1024 tokens, 512 embedding units, 1024 hidden units, and 8 attention heads. It was pretrained on the ADL dataset.

The music emotion classifier consists of two GPT-2 models, one for valence and the other one for arousal, after adding the classification head on top of the GPT-2, they fine-tuned each one of them on the extended version of VGMIDI dataset. All parameters were end-to-end trained for 10 epochs using an Adam optimizer with learning rate 3e-5 in mini-batches of size 16, dropout of 0.25, and using data augmentation, slicing each piece into 2, 4, 8 and 16 parts of equal length. It was used with the VGMIDI to help the model generalize better with different lengths.

Table 3.1 shows the results of the emotion classifiers results that were described in the paper compared to the LSTM model.

Since we are using the same emotion classifiers in our system, we reproduced the results of the same structure with the same hyper-parameters, the results are given in table 3.2. We had approximately the same accuracies, but due to the randomness in train/test splits, the values were not exactly the same.

Model	Valence	Arousal
Baseline LSTM	69	67
Fine-tuned LSTM	74	79
Baseline GPT-2	70	76
Fine-tuned GPT-2	**80**	**82**

Table 3.1: Accuracy in % of both the GPT-2 and LSTM models for music emotion classification.

Model	Valence	Arousal
Reproduced Fine-tuned GPT-2	78	80

Table 3.2: Accuracy in % of the reproduced fine tuned GPT-2 for music emotion classification.

3.5 Deep Learning Music Transformer

3.5.1 Transformers

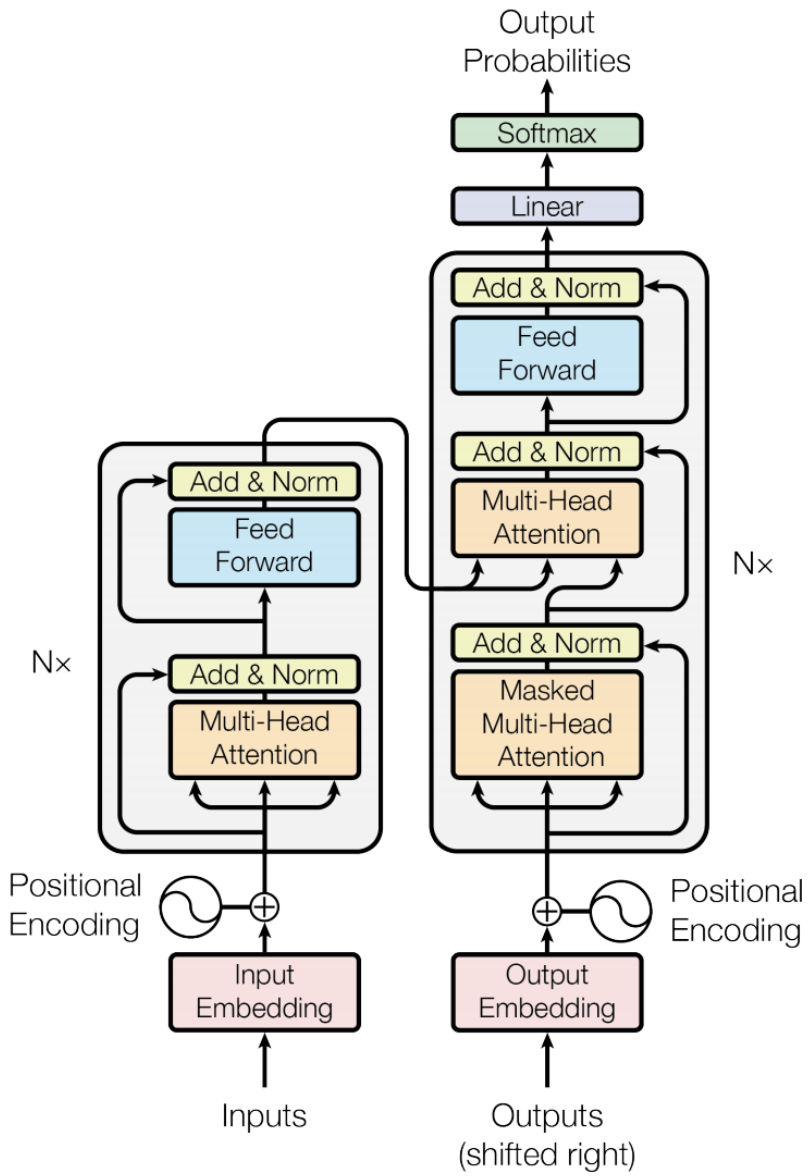

Figure 3.2 Transformer Model

The transformer model shown in figure 3.2 follows the encoder-decoder structure, the encoder maps an input sequence of tokens to an intermediate representation, and then the decoder takes the intermediate representation and generates the output sequence of tokens, one token at a time.

The transformer uses stacked self-attention layers that we will describe shortly, and point-wise, fully connected layers for both the encoder and decoder as shown in figure 3.2.

In a self-attention layer, all of the queries (Q), keys (K), and values (V) come from one place. In the case of transformers, Q, K, V come from the output of the previous layer of the encoder.

In the typical transformer architecture described in “Attention is all you need” [32], the encoder consists of 6 identical stacked layers, each contains two sub layers; the first is a multi-head self-attention layer, and the second is position-wise fully connected feed-forward network. The decoder has the same architecture but each of the 6 stacked layers

has one more sub-layer; which performs multi-head attention over the output of the encoder stack.

3.5.2 Music Transformer

Overview

Since transformers, based on self-attention, have shown very promising results in modeling long-range coherence in generation tasks, this suggests that self-attention can be good for music modelling, because music depends on repetition to build structure and meaning, such as pieces with the ABA structure in which self-reference occurs on several timescales.

In addition to self-reference, relative timing is very important in music modelling. Some approaches were developed to represent relative positional information in transformers like “Self-Attention with Relative Position Representations” [33] which were found impractical for music because of the memory complexity for intermediate relative information, which is quadratic in the sequence length.

Data Representation

A musical sequence is represented by a sequence of discrete tokens that belongs to a vocabulary which is determined by the dataset.

Training on the Piano-e-Competition dataset, which has expressive timing information the encoding is done using a vocabulary of 128 NOTE-ONs, 128 NOTE_OFFs, 100 TIME_SHIFTs, and 32 VELOCITY bins, this encoding was proposed in This time with feeling [34] which allows expressive timing at 10 ms and expressive dynamics.

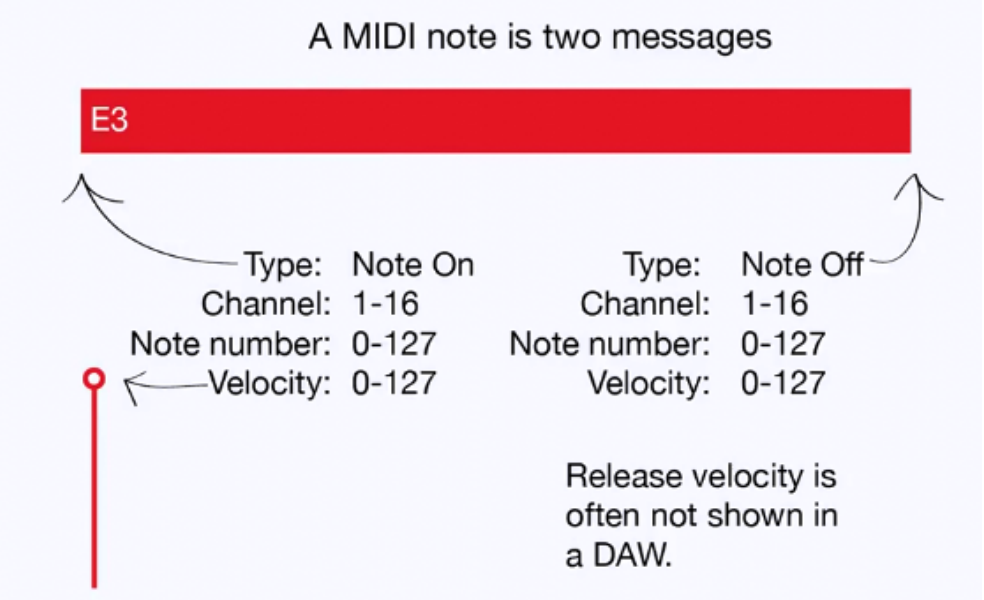

Figure 3.3 how a note is encoded

Several MIDI events are required to represent a music sequence as given below.

1. VELOCITY: represents how strong the next note(s) will be played.
2. TIME_SHIFT: adds a time interval between MIDI events.
3. NOTE_ON: starts a note with a specific number.
4. NOTE_OFF: ends a note with a specific number.

So, for example, to play note as in figure 3.3, four midi events can be used, first a velocity event to set the velocity of the next note, a note on event to start the note on time 0, then a time shift with value x to keep the note on for a while, and finally a note off event to end this note at time x.

Each of the 4 events is then mapped to a number representing the event and the value to obtain a vocabulary of 388 tokens.

Memory Efficient Implementation Of Relative Position-Based Attention

Music transformer uses a memory efficient implementation of the relative position-based attention proposed in “Self-Attention with

Relative Position Representations" [33] to allow attention to be informed by how far two positions are apart in a sequence.

Relative Local Attention

Since music sequences are considered to be very long compared to text, the quadratic memory complexity of the baseline transformer model is considered to be impractical. Local attention was used by chunking the input sequence into non-overlapping blocks, Where each block attends for itself and the previous block. The music transformer also extends the relative attention to the local use which reduces the memory requirements and allows for modelling longer sequences.

Pretrained Music Transformer

In our work, we use the pretrained music transformer[6] trained on the Piano-e-Competition dataset as our language model instead of the GPT-2 that was trained on the ADL dataset, we use it in our system for music generation as explained by figure 3.4.

Figure 3.4 Pretrained Music Transformer

The music transformer takes as input the previous tokens as an input sequence and has two outputs:

1. Output tokens: a sequence of tokens of a prespecified length that represents the next likely tokens to occur after the input tokens.
2. Score: this is the log of the probabilities of the output tokens (Maximum Likelihood Probability). Due to the large number of possibilities for each token in the output sequence the probability of the output sequence happens to be too small, and hence the log of the probabilities is returned instead of the probability itself.

So we use the music transformer to generate a beam of output sequences for each input sequence with the probability of each

sequence in the beam; this helps us implement the stochastic multi-objective beam search described in section 3.6.

Using the music transformer instead of the GPT-2 language model helps our system to generate higher quality classical music as will be shown in Chapter 5.

3.6 Stochastic Multi-Objective Beam Search

This component is an adaptation to the Stochastic Bi-Objective Beam Search in [21]. In the original paper, the beam search algorithm aims to generate music sequences that are both realistic, pleasant to hear, and that match the target emotion at the same time, hence the term Bi-Objective. Also, the stochasticity of SBBS comes from the fact that it samples from distribution to get the next beam sequences rather than selecting the highest probability greedily.

We improved the selection criteria of the bi-objective beam search which used the music emotion probability and the music tokens probabilities as selection metrics.

We first doubled the weight of the music valence and arousal to strongly capture the required sentiment, which improved our search algorithm with respect to the sentiment delivered.

Second, we added a penalty to the sequences that have unwanted repetitions by lowering the probability of this sequence according to the number of repetitions in it.

And finally we removed any sequence that is a duplicate of another sequence in the current beam by multiplying its probability with a very low number, to avoid the log of zero. The cause of these duplicate sequences is that the music transformer might generate the same token for the same input sequence if it has a very high probability.

In the rest of this section we describe in detail the beam search algorithm, and the algorithms we used in an attempt to solve the problems of using the music transformer instead of the GPT-2.

3.6.1 Detecting Repetitions

We found that the generated sequences from the system have lots of annoying repetition like repeating a single note of a single chord or even a small music composition. Repetitions are required in music as we stated earlier but to the limit where it's required to repeat some parts or notes one or more times.

We designed a special algorithm to detect repetition in a sequence of tokens. To formally state the problem in simple words, the required algorithm takes as an input an array of length n that contains integers. The output of the algorithm should find the index and the length of the first smallest repeating block in the array where the rest of the array is just a repetition of this block. For example:

Input array: [1, 2, 4, 3, 6, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

Output: index is 5 and length is 3.

You can notice in the previous example that the array contains some normal tokens at first and then some block keeps repeating to the end.

Our algorithm does a simple sliding window with different sizes until it finds the repeating block:

```
Function findRepeatingBlock(arr):
  Window_size = 1
```

```
  Starting_index = 0
  While starting_index < n:
    While window_size < n/2:
      If isRepeatingBlock(arr, starting_index, window_size)
          Return starting_index, window_size
      Else
        Window_size = window_size + 1
    Start_index = start_index + 1

Function isRepeatingBlock(arr, starting_index, window_size):
  For each block of length window_size in arr starting at starting_index:
    Check if all blocks are identical.
```

When we first applied this algorithm to the output sequences which we know there is a repetition in them, we didn't get the expected results, this was the result of the different midi events in a sequence. It's not only notes, so we first filtered the sequence to obtain only the NOTE_ON events, this achieved the target we want.

The penalty applied to the sequence probability is 1/m where m is the number of repeating blocks in a sequence, the more repetitions in a sequence the less weight it gets in the selection process.

3.6.2 Search Process

The first step in the search process is to initiate the beam, which can be initiated from a primer sequence or from scratch.

Second, the search algorithm calculates the initial probabilities of the music by calculating the music valence and arousal probabilities, repetition penalty, and music likelihood, finally, the probability is obtained by multiplying all the factors.

During each of the next steps in the search algorithm, new tokens are generated and the beam is expanded as in the pseudo code at the end of section 3.6.3, then the beam probabilities are calculated and the top p approach is applied to select the next beam.

The search steps are repeated until the length of the sequences in the beam reaches the required length.

Finally, the sequence with the highest probability is chosen as the output.

3.6.3 Sampling Mechanism

We found that the method of sampling a constant beam size from the generated sequences to form the next beam, which is the originally used (top K method), is not efficient enough. Therefore, we followed the approach of nucleus sampling introduced in “The Curious Case of Neural Text Degeneration” [35] which truncates the unreliable tails of

the probability distribution and samples from the nucleus that contains the vast majority of the probability mass. The idea behind this method is to use the probability distribution shape to determine the set of sequences to continue with to the next step, the top-p sequences are determined as the smallest set that satisfy the following condition:

$$\sum_{i=0}^{len(top-p\ sequences)} P(S_i) \geq p$$

Where p is the only parameter to this method, the S's are the selected sequences, we implement this mechanism by sorting all the generated sequences according to their probabilities and select the top n sequences that have a probability summation bigger than the parameter p. In our work we set the parameter p to 1.5-2, this actually helps our algorithm to have more variety when it needs and less variety when it's certain about some sequences.

The intuition behind the nucleus sampling, is when we have multiple sequences with low probabilities we don't want to just keep the top k, we want to keep more sequences to give the algorithm more variability in the next step, but when the generated sequences have high probability, the algorithm doesn't need to have more variability.

```
for each search step:
feed_batch = extract the last feed_len tokens from each sequence in the beam
gen_tokens, probs = calculate_new_tokens_with_probabilities(feed_batch)
sorted_probs = sort(probs)
for seq in gen_tokens:
     if seq.last equals end_token:
         sorted_probs[seq] = 0
expanded_beam = old_beam + gen_tokens
valence , arousal = calculate_emotion_probabilities(expanded_beam)
rep_panalty = calculate_repetition_penalty(expanded_beam)
dup_penalty = calculate_duplication_penality(expanded_beam)
beam_probs = valence * valence * arousal * arousal * probs * rep_penalty *
dup_penalty
next_beam = sample_top_p(beam_probabilities, p)
```

3.7 Music Synthesizer

Music synthesizer is a machine that electronically generates and modifies sounds, usually with the use of a digital computer. Synthesizers are used for the composition of electronic music. Software synthesizers are computer programs that generate digital audio, especially for music, this task used to require a dedicated hardware device, but now with the advances in processing speed and the use of floating point processing SoftSynth is used widely.

SoundFonts, with extension sf2, are files that house a collection of sounds that are grouped together to form instruments. They are then loaded into a software that reads and plays the sf2 format, called sound card, which can then be triggered by midi controllers. You can loop the wav files, allocate layers, split points, key ranges, apply filters and lfos, add effects and so on. Since most SoundFonts are produced from sampled real instruments, their quality can be exceptional.

After generating the midi file representing the generated music sequence, we then use fluidsynth [29] with salc5light-piano.sf2 to generate the audio of this midi file representing the music generated by the system.

Chapter 4

Implementation Details

4.1 Introduction

In this chapter we discuss the implementation details of the system, the different components, the interactions between them, and the difficulties we faced while building and integrating the different components. In section 4.2, we will go through the pretrained Music Transformer. In section 4.3, we will describe the integration between the Music Transformer and the Music Composer in detail as well as the problems we faced and the solutions we came up with. In section 4.4, we will talk about the system parameters and their final values after hyper-parameter tuning.

4.2 Pretrained Music Transformer

As we explained earlier, we use the pretrained music transformer as a black box to some extinct in our system to avoid the hidden details of the language model and the details of the underlying modules developed by Magenta and tensorflow.

The available music transformer is based on tensorflow version 1.x, and was trained as a tensor2tensor problem with a vocabulary size of 388 tokens as described earlier.

Ignoring most of the irrelevant details, the generation process takes 2 parameters:

1. Decode_length, which is the length of the output tokens.
2. Targets, which are the primer tokens that the transformer will use to deliver a continuation, if the targets are empty the transformer will generate tokens from scratch.

After specifying these two variables the transformer samples the next tokens and returns multiple outputs, from which we are interested in only two of them:

1. Output tokens, these are tokens of length decode_length, and should be concatenated with the targets to obtain the output sequence.
2. Score, which is the log probability of the output tokens with base e.

We implemented two functions on top of the music transformer to help us integrate the music transformer module with the music composer:

1. Beam Initialization:

   To initialize the beam for the search algorithm in the music composer this functionality generates 300 different initial sequences by setting the targets to one token from the vocabulary and generating a sequence, then repeating this procedure for 300 different targets with a single token.

2. Generating a Continuation

   This functionality generates n new tokens given a primer sequence of tokens.

Those are the two basic functionalities we need to integrate the music transformer with the music composer. In section 4.3, we will describe the integration between the music transformer and the music composer in detail, this will deliver a good understanding of the complete system.

4.3 Integrating Music Transformer With Music Composer

In this section we explain the details and the steps taken in the integration process between the two modules, we also discuss some problems that occurred during the integration and how we tackled them.

4.3.1 Client-Server Architecture

Due to the incompatibility between the two modules with respect to the working environment where the music transformer uses Tensorflow version 1.x and the music composer is based on Tensorflow version 2, we had to walk around this incompatibility on the purpose of building a functioning prototype instead of diving into the details of upgrading the music transformer to match the environment of music composer.

In order to overcome this problem we adopted a client-server architecture in which the client here is the music composer, and the server is the music transformer.

We built a simple, yet effective, REST API for the transformer, with multiple endpoints that reacts to the orders of the music composer to generate the required music tokens. As we described in section 4.2, we built two main functions on top of the music transformer; those

functions can be viewed as different end points that the search algorithm communicates with to obtain the generated tokens.

The client-server architecture worked perfectly although it has its own drawbacks such as the generation speed, which is one of the major problems we tried to address in our system. We will explain some of the solutions in section 4.3.2.

4.3.2 Vocabulary Incompatibility

Yet another problem to fix, was the vocabulary domain used in the two modules, this prevented us from using a single vocabulary, specially because of the sentiment classifier, which is trained on the vocabulary generated from the VGMIDI dataset. This vocabulary represents the music differently from the vocabulary of the Music Transformer.

To overcome this problem we had to use both vocabularies, with a conversion mechanism to the sequences from one domain to the other

domain, since both vocabularies can be converted to MIDI at the end, we used this conversion in our advantage to build the conversion mechanism.

We decided to add the conversion mechanism in the server side, because the logic to read the music transformer tokens and convert them to MIDI was incompatible with the music composer environment. Hence, we added two additional end points for the conversion mechanism:

1. Decoder: decodes the transformer tokens to music composer tokens by first converting the sequence to midi representation, then converting the midi messages to text representation used by music composer, the text is returned, and the client side uses it as is or translates the text to the ids of the tokens in the vocabulary.
2. Encoder: encodes the music composer tokens represented as text to the music transformer domain, by converting the text to midi and the midi back to tokens.

To reduce the overhead of the conversion, we used the music transformer tokens in every part of the system, except for the sentiment evaluation of the sequences where we had to convert the tokens so that the sentiment classifier can produce correct outputs.

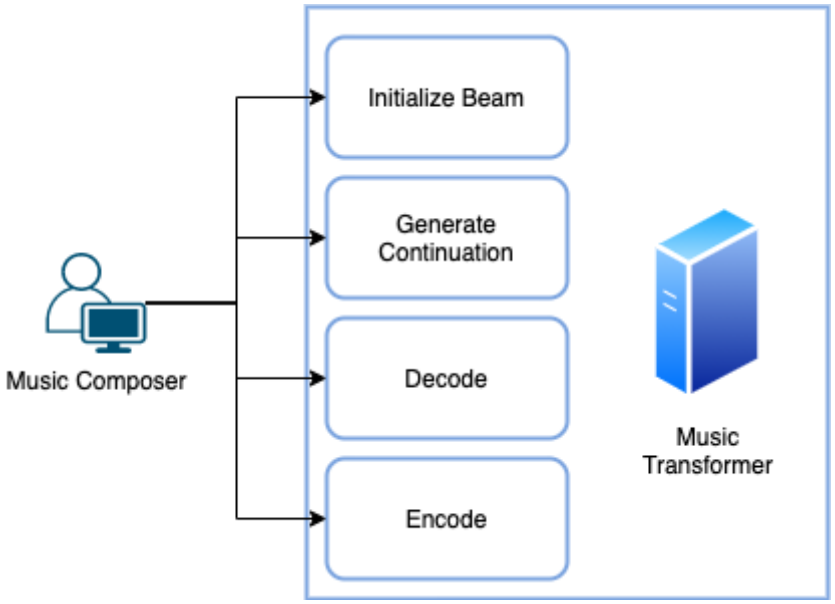

Figure 4.1: Music Composer and Music Transformer as a client-server model

4.4 System Parameters

In order to obtain the best performance with respect to generation speed, music quality, and emotion capturing, we tried different sets of parameters, we built our system to have several changeable parameters which include:

1. Beam size: in case the traditional beam search with static beam size is used, this parameter specifies the fixed beam width.
2. Duration: the length of the output in terms of the number of tokens in the output sequence, this means that the output duration in seconds can't be prespecified, instead, we use the number of tokens.
3. Initial tokens length: if it's zero, the system will sample random tokens from the VGMIDI dataset with the desired emotion. Otherwise, this parameter specifies the initial sequence length to initiate the beam.
4. Step size: this is one of the most important parameters, which tells the system how many tokens should be generated in each

step. We observed that the model generates music of higher quality if the step size is high enough, but it will limit the search algorithm from capturing the right emotion, so we should use high enough step size to obtain good quality music, but not too high to give the search higher search space to find the right emotion.

5. Branching factor: this is used in both the traditional beam search and the beam search with nucleus sampling [35], it specifies how many variations should be generated out of each sequence in the beam in each step. Again, this factor affects the generation speed heavily, so we shouldn't make it too high due to the computational limits and the bottleneck we created using the client-server architecture.
6. Valence: the valence dimension of the desired emotion.
7. Arousal: the arousal dimension of the desired emotion.
8. Probability threshold: this is specific to the nucleus sampling method, denoted as p, specifies the threshold of the probabilities summation described earlier.

9. Primer path: this specifies the path to a midi file, if this parameter is set, the model generates continuation to this midi file.
10. Feed length: this parameter controls how many tokens from the sequence should be fed to the music transformer, which means that not all of the sequence is sent to the music transformer to generate the next tokens, only the last f tokens. This helps the system to generate the required number of tokens faster without exponential latency that increases with the size of the generated tokens.

In the following table, we show the final values for these parameters after doing hyper-parameter tuning:

Parameter	Value
Beam Size	5
Initial tokens length	16
Step size	8-16
Branching factor	3-5
Probability Threshold	1.5-2
Feed length	The last 100 tokens

Table 4.1: Final set of parameters in our proposed System

Chapter 5

Evaluation

5.1 Introduction

In this chapter, we briefly show our Experiments and results to evaluate our proposed model and compare it with the baseline model. In section 5.2 we will discuss the objective evaluation. And in section 5.3 we will discuss the subjective evaluation.

5.2 Objective Evaluation [36]

The objective evaluation methods on data-driven music generation are divided into the following three categories: probabilistic measures such as likelihood and density estimation, model-specific metrics and metrics using general musical domain knowledge for various humanly interpretable metric and the advantage taking into account not only in their interpretability, but also in their generalizability and validity. We will focus on general musical domain knowledge. It is to be noted that our objective evaluation is mainly based on the framework proposed in [36] and presented in Figure 5.1. As given the figure, the approach depends on extracting musical domain knowledge features. These features will be explained in the next section. We then use absolute and relative measurements. The absolute measurements used are the mean and standard deviation for each feature. The relative measurements used are the mean and standard deviation of the intra-set distances for each feature and the mean and standard deviation of the inter-set distances for each feature.

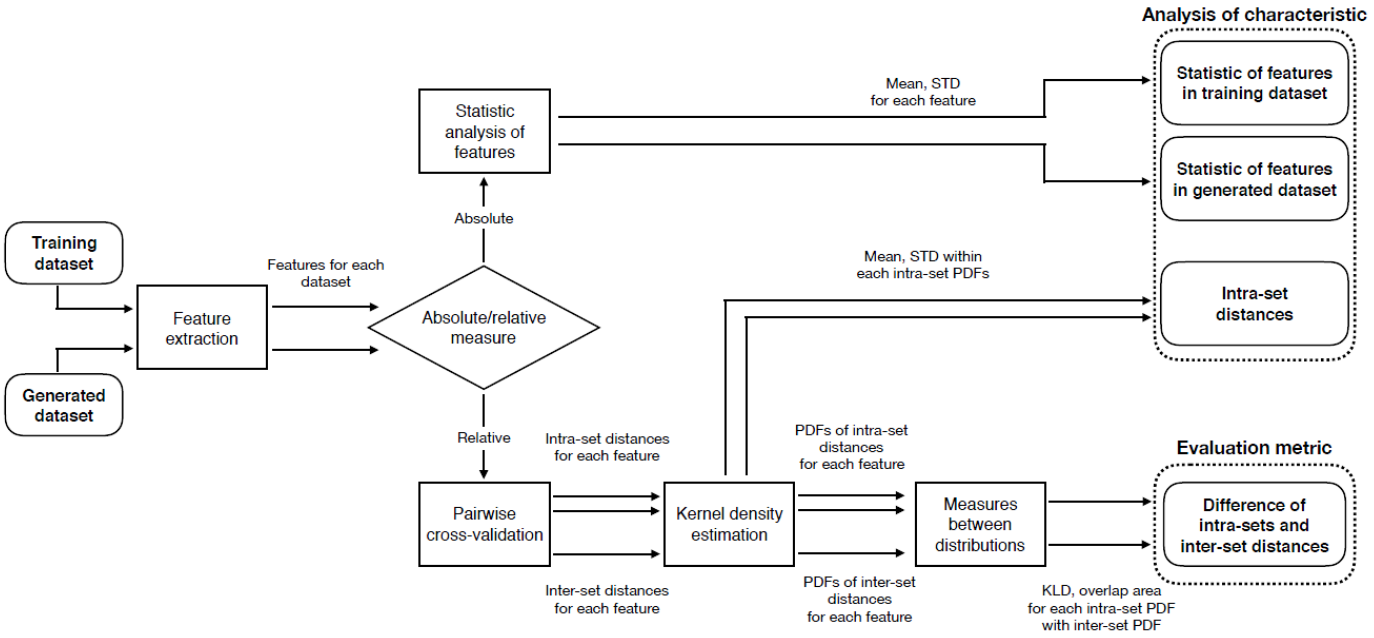

Figure 5.1: General workflow of the evaluation method

5.2.1 Extracted features Explanation

We based our evaluation on some extracted features. These features are computed for both, the entire sequence, and for each measure in order to get some structural information. The following is an explanation of the extracted features.

1) **Pitch count (PC):** The number of different pitches within a sample.

2) **Pitch class histogram (PCH):** represents the octave-independent chromatic quantization of the frequency continuum.
3) **Pitch class transition matrix (PCTM):** The two-dimensional pitch class transition matrix is a histogram-like representation computed by counting the pitch transitions for each (ordered) pair of notes.
4) **Pitch range (PR):** The pitch range is calculated by subtraction of the highest and lowest used pitch in semitones.
5) **Average pitch interval (PI):** Average value of the interval between two consecutive pitches in semitones.
6) **Note count (NC):** The number of used notes.
7) **Average inter-onset-interval (IOI):** To calculate the inter-onset-interval in the symbolic music domain.
8) **Note length histogram (NLH):** To extract the note length histogram, we first define a set of allowable beat length classes, Then the classification of each event is performed by dividing the basic unit into the length of (barlength)=96, and each note length is quantized to the closest length category.

9) **Note length transition matrix (NLTM):** the note length transition matrix provides useful information for rhythm description [37].

5.2.2 System Comparison

We will present our system comparison experiments where we evaluate two music generation systems. This experiment compares the main component of the baseline model (pre-trained GPT-2) discussed in section 3.2 and the main component of the proposed model (pre-trained Music Transformer) discussed in sections 3.3-3.5. Since, we don't have the training set of the compared systems, we base all the absolute measurements on features from the generated datasets. Also,the intra-set distances are applied only on the generated set from each system. While the inter-set distances are applied between the 2 sets that are generated from the two compared systems.

Generated Dataset

The baseline (Pretrained GPT-2) model and the pre-trained Music Transformer model generate 100 samples each. Each sample contains a melody with 8 bars. All samples are generated from scratch.

Analysis And Discussion

The results of the system comparison experiment are shown in Table 5.1. It can be observed that the absolute measurements PC, NC and PR indicate that the Music Transformer tends to contain more pitches and use more notes and has a higher average pitch range than Pretrained GPT-2. This interesting result of the higher quality of the music generated by the music transformer in the subjective study given in section 5.3, can be related to the characteristics of higher pitch range, pitch count, and note count that we find in the absolute measures. Also, the higher mean of intra-set distance for some of the features of Transformer, especially, NC and PC compared to GPT-2 indicates that the Transformer samples can have higher diversity. This may be related to the promising results of the transformer compared

to GPT-2 in terms of accuracy in reflecting the right emotion (as will be shown in section 5.3).

metrics	GPT-2				Music Transformer				Inter Set	
	Intra Set		Absolute		Intra Set		Absolute			
	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD
PC	9.14	7.31	18.23	8.23	13.19	10.11	29.01	11.69	14.47	10.54
PC/bar	11.43	5.08	-	-	11.8	5.41	-	-	11.72	5.34
NC	133.33	115.41	246.64	124.1	202.10	144.13	315.39	174.64	184.9	128.1
NC/bar	39.11	27.60	-	-	21.46	14.55	-	-	37.34	26.90
PCH	0.46	0.15	-	-	0.42	0.13	-	-	0.439	0.139
PCH/bar	1.80	0.37	-	-	1.48	0.316	-	-	1.66	0.34
PCTM	75.60	62.95	-	-	117.67	74.45	-	-	100.3	72.05
PR	14.55	10.84	30.07	12.76	13.52	9.89	45.62	11.78	19.37	12.97
PI	3.52	2.82	6.86	3.17	3.42	3.088	11.09	3.24	5.08	3.57
IOI	0.084	0.099	0.14	0.09	0.12	0.11	0.24	0.11	0.138	0.11
NLH	0.56	0.30	-	-	0.435	0.17	-	-	0.733	0.18
NLTM	154.53	97.57	-	-	98.53	58.54	-	-	161.3	93.65

Table 5.1: Experimental result for characteristic comparison of generation models

5.3 Subjective Evaluation

Since music and sentiment are both subjective and may vary from a single human being to another, we found that a subjective user study would be very insightful.

To design the user study we generated a total of 16 music pieces, 8 from our proposed system, and another 8 pieces from the baseline bardo composer. Each set of pieces contained 2 pieces for each of the 4 emotions.

Piece NO.	Piece Emotion	Generating System
1	Suspicious	Proposed Model
2	Happy	Proposed Model
3	Suspicious	Proposed Model
4	Calm	Baseline
5	Calm	Proposed Model
6	Aggressive	Proposed Model
7	Happy	Baseline
8	Aggressive	Baseline
9	Calm	Proposed Model
10	Happy	Baseline
11	Happy	Proposed Model
12	Aggressive	Baseline
13	Suspicious	Baseline
14	Suspicious	Baseline
15	Calm	Baseline
16	Aggressive	Proposed Model

Table 5.2: Order of data in survey

The 16 pieces were shuffled as in table 5.2, and the 46 human subjects who participated in the survey were asked to identify the sentiment of each piece by selecting one of 4 emotions, and also were asked to rate the quality of each piece on a scale from 1 to 5.

To ensure that the survey was not biased the distribution of the pieces generated from each system was equal, and the human subjects who took the survey didn't know that there were two different systems being evaluated in the survey, they were only asked to evaluate a computer generated music.

Piece No	Intended Emotion	System	Participants	Correct Identifications
1	Suspicious	Proposed Model	46	31
2	Happy	Proposed Model	46	42
3	Suspicious	Proposed Model	46	23
4	Calm	Baseline	46	8
5	Calm	Proposed Model	46	37
6	Aggressive	Proposed Model	46	37
7	Happy	Baseline	46	9
8	Aggressive	Baseline	46	21
9	Calm	Proposed Model	46	33
10	Happy	Baseline	46	1
11	Happy	Proposed Model	46	40
12	Aggressive	Baseline	46	13
13	Suspicious	Baseline	46	16
14	Suspicious	Baseline	46	14
15	Calm	Baseline	46	1
16	Aggressive	Proposed Model	46	30

Correct Identification is the number of people who correctly identified the intended emotion

Table 5.3: Emotion evaluation

Piece No	System	Participants	Average Quality Rating
1	Proposed Model	46	3.1
2	Proposed Model	46	4.0
3	Proposed Model	46	3.7
4	Baseline	46	3.7
5	Proposed Model	46	4.4
6	Proposed Model	46	2.7
7	Baseline	46	3.5
8	Baseline	46	2.1
9	Proposed Model	46	3.6
10	Baseline	46	2.7
11	Proposed Model	46	3.6
12	Baseline	46	3.1
13	Baseline	46	2.8
14	Baseline	46	2.8
15	Baseline	46	3.1
16	Proposed Model	46	3.0

Table 5.4: Average rating of music quality

After collecting the responses from the human subjects (see tables 5.3, 5.4), we analysed the data to summarize the survey in meaningful numbers. First, we calculated the accuracy of each piece by dividing the number of people who were able to correctly identify the intended emotion of the piece by the total number of participants, then we

calculated the average accuracy per system to summarize the sentiment accuracy in a single number for each system.

For the music quality we calculated the average rating per system.

Piece No	System	Accuracy	Average Quality Rating
1	Proposed Model	67.39%	3.1
2	Proposed Model	50.00%	4.0
3	Proposed Model	80.43%	3.7
4	Baseline	71.74%	3.7
5	Proposed Model	86.96%	4.4
6	Proposed Model	91.30%	2.7
7	Baseline	65.22%	3.5
8	Baseline	80.43%	2.1
9	Proposed Model	30.43%	3.6
10	Baseline	34.78%	2.7
11	Proposed Model	2.17%	3.6
12	Baseline	17.39%	3.1
13	Baseline	19.57%	2.8
14	Baseline	2.17%	2.8
15	Baseline	45.65%	3.1
16	Proposed Model	28.26%	3.0

Table 5.5: Accuracy and average rating for each pieces

System	Average Accuracy	Average Quality Rating (out of 5)
Proposed Model	74.18%	3.52
Baseline	22.55%	2.97

Table 5.6: Final results for the survey

It was noticed from the survey results that the baseline model was able to model the arousal of the music to some level that was acceptable, but the valence of the music was poorly modeled, which resulted in a confusion between the pieces - recall the mapping of the four emotions to the Valence-Arousal emotional model discussed in section 2.2.3. Also, the bi-objective search algorithm in the baseline model equally weighted the music emotion objective and the music quality objective, which resulted in generating ambiguous pieces of music, while in our proposed model, we increased the weight of the music emotion objective to make sure that the generated music strongly matches the desired emotion.

Table 5.5 shows the summarization of the survey results. It's obvious from the survey that human subjects were able to identify the intended emotion of the music generated from our proposed system easily and smoothly. This explains the higher accuracy improvement shown in table 5.5. While we were able to generate music with the intended emotion, we also focused on improving the music quality. This was a result of using the music transformer which is the state of art model for music generation. So it was expected to outperform the baseline model in terms of quality, in addition to our modified search techniques that ensure that the generated music is as real as possible. This can be shown in table 5.6 by comparing the average quality rating of the two systems.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In our work, We aim to build a completely usable deep learning system that can generate music with a given sentiment from scratch or from a given primer, by improving the baseline model, the bardo composer [21], using deep learning music transformer and improved search techniques. This allows the proposed model to have a better understanding of the sentiment and to give promising results in terms of quality and reflection of the right emotion.

We evaluate the sentiment of the generated music using our deep learning sentiment classifier, and using a subjective evaluation survey.

We also use an objective evaluation system to evaluate the quality of the music generated.

6.2 Future Work

As mentioned before, most of the problems arise due to the limitation of the data, so we plan to collect more data from some music platforms like Youtube and Spotify to enhance the accuracy of the system and generalize the type of generated dmusic.

Our system could be extended to provide a background music for videos by training a model that captures emotions in videos and feeding our model with these emotions to generate the desired-emotion music.

We are inspired by CTRL (**C**onditional **Tr**ansformer **L**anguage Model) [38], this model is capable of generating text conditioned on control codes that specify domain or style. Therefore, we aim to modify this model and train it on musical data pre-appended with

control codes of sentiment i.e., Happy or Sad. One advantage of this model can be its lesser generation time, as no need to pass each sequence to the emotion classifier to calculate the joint probability of the desired emotion used in beam search. Another advantage is that the model is trained end-to-end. However, the main drawback in using CTRL is the huge need for data. In fact CTRL is trained on 140 GB text data [38]. One solution for this drawback is to do data scraping from different websites.

Bibliography

[1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, “Wavenet: A generative model for raw audio,” arXiv preprint arXiv:1609.03499, 2016.

[2] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio, “SampleRNN: An Unconditional End-to-End Neural Audio Generation Model,” arXiv preprint arXiv:1612.07837, 2017

[3] Oord, Aaron, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche et al. "Parallel wavenet: Fast high-fidelity speech synthesis." In International conference on machine learning, pp. 3918-3926. PMLR, 2018.

[4] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel and Douglas Eck, “Enabling Factorized

Piano Music Modeling and Generation with the MAESTRO Dataset,” arXiv preprint arXiv:1810.12247, 2019

[5] Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. "Generative adversarial networks: An overview." IEEE Signal Processing Magazine 35, no. 1 (2018): 53-65.

[6] Huang, Cheng-Zhi Anna and Vaswani, Ashish and Uszkoreit, Jakob and Shazeer, Noam and Hawthorne, Curtis and Dai, Andrew M and Hoffman, Matthew D and Eck, Douglas, “Music Transformer: Generating Music with Long-Term Structure,” arXiv preprint arXiv:1809.04281, 2018

[7] Payne, Christine. "MuseNet." OpenAI, 25 Apr. 2019, openai.com/blog/musenet

[8] Russell, James A. "A circumplex model of affect." Journal of personality and social psychology 39, no. 6 (1980): 1161.

[9] Li-Chia Yang, Szu-Yu Chou and Yi-Hsuan Yan, “MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation,” arXiv preprint arXiv:1703.10847, 2017

[10] Dong, Hao-Wen, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. "Musegan: Multi-track sequential generative

adversarial networks for symbolic music generation and accompaniment." In Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[11] Ankur P. Parikh, Oscar Täckström, Dipanjan Das and Jakob Uszkoreit, “A Decomposable Attention Model for Natural Language Inference,” arXiv preprint arXiv:1606.01933, 2016

[12] Rewon Child, Scott Gray, Alec Radford and Ilya Sutskever, “Generating Long Sequences with Sparse Transformers,” arXiv preprint arXiv:1904.10509, 2019

[13] https://www.classicalarchives.com/

[14] https://bitmidi.com/

[15] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel and Douglas Eck, “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset,” arXiv preprint arXiv:1810.12247, 2019

[16] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari and Brian Kulis, “Conditioning Deep Generative Raw Audio Models for Structured Automatic Music,” arXiv preprint arXiv:1806.09905, 2018

[17] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts,

Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore and Douglas Eck, “Onsets and Frames: Dual-Objective Piano Transcription,” arXiv preprint arXiv:1710.11153, 2018

[18] Madhok, Rishi, Shivali Goel, and Shweta Garg. "SentiMozart: Music Generation based on Emotions." In ICAART (2), pp. 501-506. 2018.

[19] Lucas N. Ferreira and Jim Whitehead, “Learning to Generate Music With Sentiment,” arXiv preprint arXiv:2103.06125, 2021

[20] Alec Radford, Rafal Józefowicz, and Ilya Sutskever, “Learning to generate reviews and discovering sentiment,” arXiv preprint arXiv:1704.01444, 2017

[21] Ferreira, Lucas, Levi Lelis, and Jim Whitehead. "Computer-generated music for tabletop role-playing games." In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 16, no. 1, pp. 59-65. 2020.

[22] Poole, David L., and Alan K. Mackworth. Artificial Intelligence: foundations of computational agents. Cambridge University Press, 2010.

[23] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, “Bert: Pre-training of deep bidirectional transformers for language

understanding,” arXiv preprint arXiv:1810.04805, 2018

[24] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multi task learners." OpenAI blog 1, no. 8 (2019): 9.

[25] https://www.piano-e-competition.com/

[26] Raffel, Colin. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. Columbia University, 2016.

[27] Bertin-Mahieux, Thierry, Daniel PW Ellis, Brian Whitman, and Paul Lamere. "The million song dataset." (2011): 591-596.

[28] Padovani, Rafael R., Lucas N. Ferreira, and Levi HS Lelis. "Bardo: Emotion-based music recommendation for tabletop role-playing games." In Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference. 2017.

[29] https://github.com/FluidSynth/fluidsynth

[30] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” arXiv preprint arXiv:1406.1078, 2014

[31] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” arXiv preprint arXiv:1409.0473, 2016

[32] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In Advances in neural information processing systems, pp. 5998-6008. 2017.

[33] Peter Shaw, Jakob Uszkoreit and Ashish Vaswani, “Self-Attention with Relative Position Representations,” arXiv preprint arXiv:1803.02155, 2018

[34] Oore, Sageev, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. "This time with feeling: Learning expressive musical performance." Neural Computing and Applications 32, no. 4 (2020): 955-967.

[35] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes and Yejin Choi, “The Curious Case of Neural Text Degeneration,” arXiv preprint arXiv:1904.09751, 2020

[36] Yang, Li-Chia, and Alexander Lerch. "On the evaluation of generative models in music." Neural Computing and Applications 32, no. 9 (2020): 4773-4784.

[37] Verbeurgt, Karsten, Michael Dinolfo, and Mikhail Fayer. "Extracting patterns in music for composition via markov

chains." In International conference on industrial, engineering and other applications of applied intelligent systems, pp. 1123-1132. Springer, Berlin, Heidelberg, 2004.

[38] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong and Richard Socher, “CTRL: A Conditional Transformer Language Model for Controllable Generation,” arXiv preprint arXiv:1909.05858, 2019