Symbolic Music Sentiment Transfer Through Extending a Style Transfer Model

Cardiff University School of Computer Science and Informatics
Computing MSc
3rd January 2024

Ellis Burgess - 22025409
Supervised by Andrew Jones and Dave Marshall
Moderated by Jing Wu

Abstract 3
Acknowledgements 3
Introduction 4
1. Background 7
I. Introduction 7
II. Music Theory 8
III. Music Emotion Recognition 10
IV. Audio Attributes for Music Emotion Recognition 14
V. Style and Sentiment Transfer in Symbolic Music 15
VI. Attribute Specification and Modification 17
VII. Symbolic Representations of Music in MIR 18
VIII. Gap Analysis 19
2. Problem Statement 19
I. Problem Definition 19
II. Primary Goal 20
III. Supporting Secondary Goals 21
IV. Methodology 21
a. Sentiment Annotated Dataset Selection 21
b. Attribute Selection 22
c. Model Architecture Modification and Retraining 23
d. Generation Modification and Results Analysis 23
3. Features of MuseMorphose and Potential for Extension 24
I. Introduction 24
II. MuseMorphose Architecture 24
III. Processing of Existing Attributes 26
IV. Training Process 28
V. Generation 30
4. Extending MuseMorphose for Sentiment Transfer 32
I. Introduction 32
II. Dataset Selection and Issues 32
III. Review of Attributes 34
a. Method 34
b. Melodic Elements 34
c. Harmonic Elements 36
d. Rhythmic Elements 37
e. Remaining Musical Attributes 38
IV. Development of Attribute Extraction Methods 38

a. Melodic Extraction 38
b. Melodic Pitch Height 41
c. Melodic Contour 42
d. Harmonic Pitch Height 44
e. Tonality 45
V. Attribute Analysis 48
a. Rhythmic Intensity 49
b. Polyphonicity 50
c. Melodic Contour 52
d. Pitch Height 53
e. Tonality 55
VI. Model Retraining 56
a. Experiment One 57
b. Experiment Two 58
c. Experiment Three 60
VII. Output Generation 63
5. Results and Discussion 64
I. Results 64
a. Results from First Generation Experiment 64
b. Results from Second Generation Experiment 67
c. Results from Third Generation Experiment 69
II. Discussion 71
a. Second Generation Experiment 71
b. Third Generation Experiment 72
6. Conclusion 73
7. Learning Reflection 74
References 77

Abstract

This dissertation attempts symbolic music sentiment transfer across Russell's four quadrants through bar-level control of musical attributes associated with changes in perceived sentiment. Based on a survey carried out by Panda et al., musical attributes associated with changes in perceived sentiment were identified and methods to extract them were created (2023). These were added to the existing MuseMorphose model, a program designed for style transfer through attribute control. The model was retrained on these additional attributes and sentiment-transferred output was generated to be fed back into an emotion classification tool.

The results from this research support found that transferring high valence pieces to low valence while leaving arousal unchanged were the most often correctly identified as their target sentiments. In addition, modifications to control of attributes related to changes in perceived arousal yielded higher accuracy in classification in later experiments, although no cases where accuracy rose above 25%. This suggests that through further changes to how attributes are controlled in sentiment transfer, symbolic music sentiment transfer across Russell's four quadrants through control of musical attributes will be achievable. Suggestions for further work are included in the conclusion of this dissertation.

Acknowledgements

I would like to thank Professor David Marshall and Dr Andrew Jones for their support and guidance while supervising this dissertation. I would also like to thank Barry Hogan for his support in navigating situations where external factors were impacting my ability to progress with the dissertation.

Introduction

The expression of emotion is one of music's core functions. Whether one's engagement with music is its creation or its consumption, music's inherent ability to communicate emotion is a major factor in our appreciation of it. As Roger Scruton said, "music inspires and consoles us partly because it is unencumbered by the debris that drifts through the world of life" (1999). It is no surprise, then, that an active research area within the field of music information retrieval is the development of programs which can identify emotion in music and generate music to fit a target emotion. In the last few years, there has been particular interest in the use of machine learning methods to achieve sentiment identification and sentiment-based generation of symbolic music, or non-audio representations of music. This is largely due to the relative simplicity compared to audio-based sentiment identification and generation. The investigations in this dissertation fall within the field of symbolic music emotion recognition and generation.

To categorise the sentiment conveyed by a piece of music, a method of categorising emotions needs to be defined. While there are a few approaches to this task, the most prevalent is based on Russell's four quadrants (1980), in which emotions are mapped across the axes of arousal and valence. For example, contentment would fall in the low arousal high valence quadrant, while anger would fall in the high arousal low valence quadrant.

While there has been substantial work done in music emotion recognition and generation, there has not yet been a tool developed which can carry out sentiment transfer on symbolic music, or the modification of a piece of music to express a given sentiment. Such a program could prove incredibly useful to composers, particularly composers for media, where a single piece of music may need to be used in multiple emotional contexts (Kalinak 2010). Therefore, the central research question of this dissertation concerns the development of such a tool: can symbolic music sentiment transfer across Russell's four quadrants be achieved through musical attribute extraction and control?

Beyond aiming to close a research gap, a tool for symbolic music style transfer could be relevant to several fields. In any area where the emotion conveyed by music is a priority, being able to quickly produce music to fit a given sentiment could be of benefit.

One example of where a sentiment transfer tool could be useful is in composition of music for film and TV, where recurring themes are often repeated and adapted to enhance the emotion conveyed in a scene (Kalinak 2010). Such a tool could by used by a composer to generate ideas on how to alter an existing piece to fit a different emotional context. This could both serve as a valuable tool in a composer's workflow, enabling them to quickly generate ideas when working on short deadlines; and provide composers with ideas for how to develop an existing piece of music in ways they may not have otherwise considered, thereby enriching their output.

A sentiment transfer tool may be useful to software developers working on projects where the sentiment conveyed by music needs to change dynamically. An example of such a project is DearDiary.ai, a web application where music is generated as a user inputs text, informed by the sentiment of the text being entered (Lane et al. 2021).

Finally, such a tool could also be useful to music listeners for a variety of reasons. Music listeners may desire a song which sounds like one they have a particular bond with, but which fits a different sentiment, for creative purposes such as storytelling, or for personal explorations of music. Additionally, being able to listen to songs they are familiar with in different emotional contexts may help listeners better understand why certain songs elicit particular emotions in them, helping them become more thoughtful consumers of music.

Attempting to create a sentiment transfer program from scratch would be beyond the scope of this dissertation. Therefore, sentiment transfer in this project will be attempted through extending an existing symbolic music style transfer tool, MuseMorphose (Wu and Yang 2022). This program was found to be suitable for the task as it approaches style transfer

through musical attribute control, and as it is well-suited to extension for novel applications. A discussion of how it is well-suited to extension is presented in Section 3.

The methodology adopted in this dissertation involves first selecting attributes which are found to be suitable for extraction from symbolic music, and which have an apparent relationship with changes in perceived sentiment. Then, functions for extracting these attributes from symbolic music is developed. The MuseMorphose architecture is updated to be able to handle the additional attributes, and then the model is retrained. Once the model has been retrained, a sentiment annotated dataset is used as input to the program for the generation of sentiment-transferred output. This output is passed into a separate music emotion recognition tool and the accuracy of identifying the target sentiment in generated music is measured.

Attempting symbolic music style transfer is a novel task with many considerations. For this reason, and to maintain feasibility of completing this project within the limitations of this dissertation, the scope of the project is limited in a few ways. As mentioned earlier, development of the program will involve extending the functionality of the existing MuseMorphose model, instead of building an entirely new program. This model was trained predominantly on pop music for solo piano, so this program will similarly be limited to this genre and instrumentation. Finally, it is not within the scope of the project to assess the musical quality of the generated sentiment transferred music. This would be a potential area for future research if the results indicate that the proposed approach for symbolic music sentiment transfer is successful, and will be discussed further in Section 5 of this dissertation.

This project draws together research from a number of areas, which are explored more thoroughly in Section 1. In particular, the fields of music emotion recognition, symbolic music generation, style transfer, musical attribute extraction, and music theory will be relevant.

Section 2 of the dissertation will explore the central research question in more depth and outline in detail the methodology used. Section 3 will contain a review of MuseMorphose and its potential for extension; in particular, this section will focus on design features of MuseMorphose which increase the ease with which the model can be extended for new applications.

Section 4 will cover the process of modifying MuseMorphose for sentiment transfer in depth, from the selection of an appropriate dataset through to modifying the music generation function so that sentiment-transferred music can be generated for analysis. Section 5 will then present the results of this music generation, along with a discussion of the results, what they indicate about how successful the project was, and what could be changed in future experiments in symbolic music sentiment transfer.

The results obtained in this project indicate that symbolic music sentiment transfer is possible through attribute control. However, this is an area in which further research is required, with some suggestions of future research directions discussed in Section 6.

1. Background

I. Introduction

This section introduces research areas and concepts which are required for this project. An explanation of music theory concepts relevant to this project is presented. The field of music emotion recognition is introduced, with examples of relevant projects provided. Panda et al.'s review audio attributes and their relationship to perceived emotion is discussed, as it is used extensively in identifying musical attributes suitable for use in sentiment transfer. This is followed by an overview of style and sentiment transfer in symbolic music, with a focus on MuseMorphose and another relevant project undertaken by Sigel et al. (2021). Finally there is a brief discussion of relevant symbolic representations of music.

II. Music Theory

As feature extraction from music is a core component of this project, several musical concepts need to be understood going forwards.

In music, notes are auditory events comprised of a pitch (the perceived frequency of the underlying sound wave), duration, and volume. Pitches in Western music are organised into twelve repeating and evenly spaced classes with unique names (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). There are seven unique letters as most scales contain seven pitches. For example, a C Major scale contains the notes (C, D, E, F, G, A, B). A related concept is pitch class sets, which refers to pitches with numbers 0-11, and is used when the relationships between pitches are more important than the pitches themselves. For example, the pitch set of a major scale is (0, 2, 4, 5, 7, 9, 11). By normalising the first pitch to zero, major scales beginning on any pitch can be expressed this way.

Numbers are added to pitch classes to indicate specific frequencies. For example, the pitch A4 refers to 440Hz; A3 refers to 220Hz, which is one octave lower than A4; and A5 refers to 880Hz, which is one octave higher than A4. In MIDI, these specific frequencies are instead described with integer numbers, and the notes contained on a standard piano keyboard represented by numbers in the range 21-108.

All musical samples in this project are taken from the EMOPIA dataset (Hung et al. 2021) or the AILabs.tw-Pop1k7 dataset (Huang and Yang 2020), which exclusively contain pieces in the time signature 4/4 (also referred to as common time due to its prevalence in Western music). This means that the music is divided into beats of equal durations, which are grouped into bars of four beats. The rate at which beats occur is called tempo. There can be many notes within a single beat, and notes can occur on beats, in between beats, and across multiple beats. The patterns that emerge from note onset times are referred to as rhythm. Rhythmic intensity refers to the density of note onset events.

Some instruments are monophonic, meaning they can produce only one note at a time.

This project is concerned with music for solo piano, which is a polyphonic instrument. This means it can produce multiple notes at a single time. A standard piano in standard tuning has 88 keys which each produce notes of a different pitch, which makes the piano capable of producing up to 88 notes simultaneously. Polyphony refers to any period of time in a piece of music which features multiple notes occurring at the same time.

Harmony in music is a means of organising notes which occur during polyphony. Three or more notes occurring together is called a chord, and there are many categories chords can fit into. Chords which are of particular relevance to this project are displayed in the table below.

Chord Quality	Pitch-class set notation	Example Chord
Augmented	0, 4, 8	D Augmented: D-F#-A#
Major	0, 4, 7	C Major: C-E-G
Dominant 7	0, 4, 7, 10	Bb Dominant 7: Bb-D-F-Ab
Sustained	0, 2, 7 or 0, 5, 7	G Sustained: G-A-D or G-C-D
Minor	0, 3, 7	A Minor: A-C-E
Minor 7	0, 3, 7, 10	E Minor 7: E-G-B-D
Diminished	0, 3, 6	B Diminished: B-D-F

Table I: Examples of chord qualities

Melody refers to an identifiable monophonic line. The melody notes of a piece often stand out, either by being extreme in pitch, or significantly louder than non-melody notes. An intuitive way to determine the melody in a piece of music is to listen to it and try to sing along. An important feature of melody is melodic contour, which refers to the shape of melody over time. For example, the melodic contour of the first four bars of “Twinkle

Twinkle Little Star," as displayed in Figure I, follows a convex shape, as it ascends and then descends.

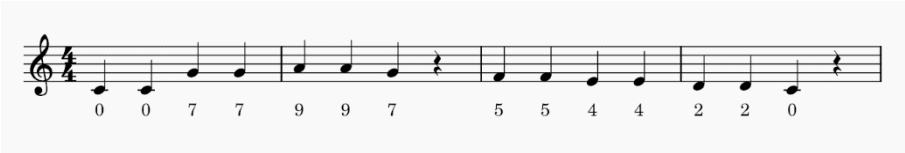

Figure I: The first four bars of *"Twinkle Twinkle Little Star" with pitch*-class set indicated.

The final musical feature relevant to this project is tonality. Tonality, or key, refers to the underlying sense of harmony during a given portion of music. Music without a clear underlying sense of harmony is described as atonal. The tonality of a portion of music can be described in terms of the mode, or scale, which best fits the implied harmony. The most common modalities in Western music are major (described earlier), and minor (0, 2, 3, 5, 7, 8, 10), although there are many others such as lydian, mixolydian, and phrygian, which all have unique pitch class sets.

The terms described here are important to understand as they relate to musical features which have a bearing on perceived emotion. They relate directly to the musical attributes extracted from sentiment data and controlled for sentiment transfer.

III. Music Emotion Recognition

Music Emotion Recognition (MER) is an area of study within the broader field of Music Information Retrieval (MIR) concerned with the use of technology, often machine learning, to classify pieces of music by their emotional content. It is a task with many applications, ranging from music recommendation systems to music therapy (Han et al. 2023). There are two major decisions in approaching MER that affect the approach and the resulting implementation: how the musical data is represented, and how emotions are

classified.

While music is an auditory art, it can also be represented symbolically, such as with music notation. Audio representations of music are stored as audio files (such as .WAV files). Symbolic representations of music, meanwhile, do not contain audio of a piece of music. Instead, they store information that can be decoded to recreate the music (for example, using a digital audio workstation or a virtual instrument) by containing information about the piece's tempo, time signature, note velocity, and note position. Music notation is a widely used form of symbolic music, where the symbolic music is "decoded" by a human performer using an instrument.

The computational approaches to understanding these two types of musical data are quite different from each other. As a result, choosing whether to work in the audio or symbolic domain majorly influences the design of MER implementations. Both musical representations have advantages and disadvantages.

One key advantage of working with symbolic music instead of audio is that it requires significantly less pre-processing. While audio data requires pre-processing to extract features such as the piece's spectrogram and Mel Frequency Cepstral Coefficients (MFCCs) from the waveform from which information about musical features can be inferred (Han et al. 2023), symbolic music already contains numerical representations of many key musical features. A flaw with SMER, however, was its limited scope for applications compared to audio domain MER, since a major application of MER is to aid in music retrieval for casual listening (Yang and Chen 2012). Thanks to a growing number of tools that can accurately transcribe audio to symbolic music representations, like the piano transcription tool proposed by Kong et al. (2021), this gap in applicability is reducing.

Having decided whether to work with music in the audio or symbolic domain, the next major decision in designing a MER tool is determining how to define the emotions being

classified. In MER, sentiments need to be clearly defined so that the program's model can understand them. There are many approaches to classifying emotions, which all fall into one of two broader methods: categorical or dimensional.

A categorical approach to classifying emotion involves the definition of several basic emotions. For example, MUSEC uses the emotions of happy, sad, angry, surprised, joy, and love to form the basis of its emotion classification function (Abboud and Tekli 2020). From the identified emotions, each one is given a weight, with the idea that human emotions of any complexity can be described as combinations of different amounts of a handful of simpler emotions (Yang and Chen 2012). This approach is viewed as being an intuitive way to understand emotions. One major issue with the categorical approach is the absence of a consensus on which basic emotions can be combined to fully encapsulate the range of human emotion. As a result, different tools using a categorical approach to emotion classification often cannot share datasets. Another issue with a categorical approach is in annotating emotions. It has been found that many tools do not identify enough basic emotions to capture the full range of complex emotions conveyed in music, and the subjective use of language around emotions can confuse subjects responsible for annotating sentiment in music (Yang and Chen 2012).

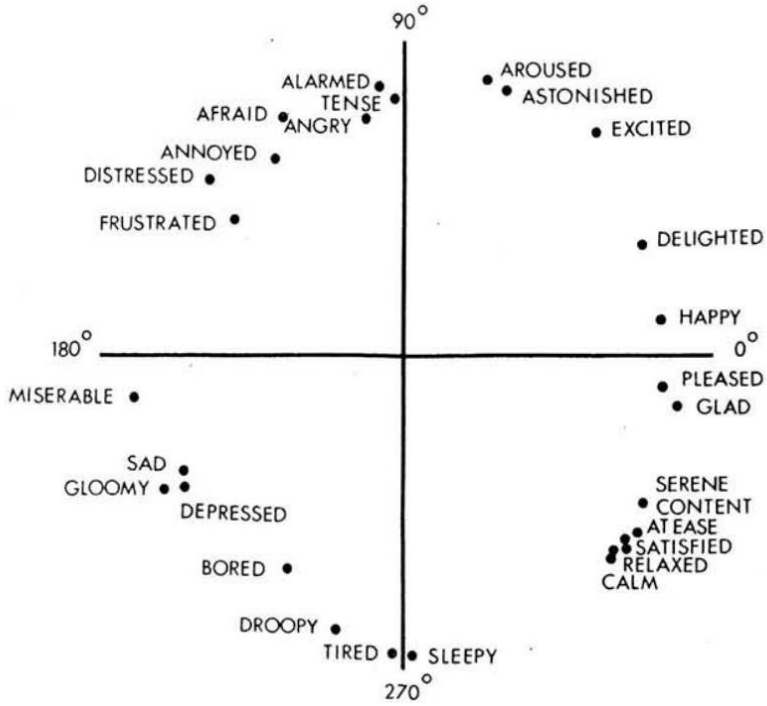

Figure II: *"Direct circular scaling coordinates for […] 28 affect words"* (Russell 1980).

The dimensional approach to classifying emotions is based on the idea that all emotions can be described by the combination of scalar values. The most common application, shown in Figure 1, is to use two scales: arousal (or how energetic an emotion is) and valence (or whether an emotion is positive or negative) (Russell 1980). By plotting arousal against valence, a two-dimensional space is created comprising four quadrants (high arousal, high valence; high arousal, low valence; low arousal, high valence; low arousal, low valence). This approach is standardised in MER, making it possible for datasets and results from different models to be applied to new contexts. However, it has been criticised for lacking the fidelity of a categorical approach in defining emotions (Yang and Chen 2012). This can be seen in the diagram above; fear and anger, for example, are two emotions which are experienced in very different ways, which can be observed in the different physiological differences they elicit (Ax 1953). However, when using four quadrants, these emotions are

contained within the same category, thereby losing nuance in emotion categorisation. This flaw is sometimes mitigated by the inclusion of a third dimension, however the increased complexity in using a three-dimensional model rather than a two-dimensional one is not matched by the increase in accuracy of emotion classification (Eerola et al. 2009).

Many MER tools use a machine learning approach. As a result, two factors to consider during their development are the selection of an appropriate dataset and the design of an effective model architecture.

To train a model for MER, a dataset of sentiment-annotated musical data is required. The dataset selection is dependent on the approach being used, since there is not a consensus on how musical data should be formatted or how sentiment should be classified. For instance, the MTG-Jamendo dataset (Bogdanov 2019) uses a categorical classification approach for audio representations of music; this would not be suitable for MidiBERT-Piano, which classifies emotion in MIDI representations of music dimensionally (Chou et al. 2021).

In addition to selecting a suitable dataset, it is essential to choose an appropriate model architecture. In the field of MER, it is customary practice to select an existing model architecture and modify it to suit the task, as opposed to creating a completely new architecture (Yang and Chen 2012). Support vector machines (SVMs) have demonstrated high accuracy in emotion classification, but newly developed emotion classification tools tend to implement modern models specifically designed for natural language processing (NLP).

IV. Audio Attributes for Music Emotion Recognition

In 2020, Panda et al. conducted a survey into the relationships between musical features and sentiment. This survey identified eight large-scale musical features (melody, harmony, rhythm, dynamics, timbre, expressivity, texture, and form), and multiple attributes for each feature. Several papers exploring the relationships between musical attributes and

perceived emotion were referenced in drawing conclusions about the relationships between these features and categorical sentiment. Additionally, methods to extract these attributes from audio are described.

This survey found that for many attributes, different studies found contradicting associations with sentiment. For example, two studies found that monophonic music was viewed as happier than homophonic music, while two different studies found the opposite to be true (Panda et al. 2023). However, there were also many attributes found to have clear associations with specific sentiments consistent across multiple studies. An example of this can be found in dynamics, as louder dynamics are consistently associated with high arousal emotions such as happiness and anger. In several cases, it is suggested that the relationship between a given attribute and perceived sentiment is more complex than a direct correlation, and that the attribute should be considered in its surrounding musical context to better understand its implications.

The results of this survey indicate that while some musical attributes have a clear and direct relationship to sentiment, others need to be considered in broader the context of the music, through taking other attributes into account or through considering how means of expressing emotions can vary across genres. Within the field of MIR, when considering musical attributes, it is important to be aware of the distinction between those that directly correlate to sentiment, and those which have a relationship to sentiment more affected by other musical features. Initial searches did not result in finding any research into attempting to model the relationships between perceived sentiment and shifts in multiple attributes; research in this area would likely be greatly beneficial to the field of MER.

V. Style and Sentiment Transfer in Symbolic Music

Sentiment transfer falls within the larger topic of style transfer, a process in which the content of one source is modified to match the style of another. For example, in the visual

domain, a style transfer may take a photograph and a famous artist as input and generate a variation of the photograph in the style of the artist. In the context of music, style transfer is a process in which an input piece is reimagined in a different style. In recent years, there have been a growing number of papers concerned with style transfer in music, however background research yielded only one paper addressing sentiment transfer (or the transfer of the perceived emotion in a piece while maintaining its overall structure).

One project, focussed on audio-domain one-shot timbre transfer, uses the application of a vector-quantised variational autoencoder (VAE) (Cífka et al. 2021). VAEs traditionally work by encoding input data into a latent space representation, then using the latent space to reconstruct the input. The approach used by Cifka et al. varies from traditional VAEs by using discrete instead of continuous latent variables (2021). For this timbre transfer task, two encoders were used; one to encode the input, and the other to encode the target style. This approach followed previous symbolic music-domain work, Groove2Groove (Cífka et al. 2020). Groove2Groove aimed to conduct style transfer in the accompaniments of popular music and jazz by similarly using two encoders and one decoder.

A recent paper approached the topic of sentiment transfer in symbolic music (Sigel et al. 2021). This approach is based on CycleGAN (Zhu et al. 2020), a generative adversarial network (GAN) which uses two sets of generator and discriminator modules to generate new data and attempt to reconstruct the original input. In their work, Sigel et al. applied this principle, though full results have not yet been published. However, the authors acknowledged that further work would benefit from the inclusion of music composition knowledge in the tuning of parameters.

Though not concerned with sentiment transfer, MuseMorphose approaches this aim by using attribute extraction and modification as means for controlling style transfer (Wu and Yang 2022). In MuseMorphose, two attributes (rhythmic intensity and polyphonicity) are

extracted from symbolic music and each organised into eight equally sized bins. These attributes can then be controlled during generation, which is undertaken by a model which combines a VAE with a transformer, to overcome the issue of VAEs performing poorly over longer sequences (Wu and Yang 2022). The model developed in MuseMorphose is the basis on which this project has been built.

VI. Attribute Specification and Modification

As demonstrated in MuseMorphose, it is possible to control the output of a VAE-based generation model by manipulating attributes extracted from the input file (Wu and Yang 2022). As identified by Panda et al., there are a substantial number of attributes or features that could be extracted from music, many of which have strong relationships with perceptions of different sentiments in the Western music tradition (2023). By carefully selecting which musical attributes are selected as input for a VAE, the generated music can be more directly controlled. However, not all attributes are suitable for use in a VAE, since VAEs perform better when using continuous attributes rather than categorical ones.

In the case of MuseMorphose, the two attributes selected (rhythmic intensity and polyphony) were selected for their ease of identification for listeners, and their association with musical arousal (Wu and Yang 2022). Owing to these factors, modifying rhythmic intensity and polyphonicity produces a recognisable change in arousal between the input piece and the new generation.

In their review of audio features for music extraction, Panda et al. identified many musical attributes, and their relationships to different sentiments (2023). While several attributes have associations with contradictory emotions (for example, both monophonic and homophonic music are identified as being happier than their counterparts), several have clearer associations (such as a major key being associated with positive emotions and a minor key being associated with negative emotions). By extracting continuous attributes with clear

emotional connotations from a musical input, a VAE could be used to transfer the sentiment expressed by a piece.

VII. Symbolic Representations of Music in MIR

To facilitate data processing for symbolic music, data is often reformatted. One symbolic representation of music used in many MER programs, including MuseMorphose, is REMI (Revamped MIDI Derived Events), proposed by Hsiao et al. (2021).

To understand REMI, we must first have a basic understanding of MIDI data. MIDI data contains tempo and time signature data at the start and whenever changes in these values occur. At regular intervals there is a temporal event to indicate the current position within the MIDI file. The notes within the piece are then described by note-on and note-off events which accompany the relevant temporal event. A note-on event indicates the start of a note, while a note-off event indicates the end. One way to understand these events as they relate to acoustic instruments is by considering a single key on a piano; note-on events correspond to the action of pressing down a piano key, while note-off events correspond to the action of releasing the piano key. Note-on and note-off events contain the pitch of the note being played, represented by a number in the range 0-127, and the velocity of the event, which corresponds to the note's volume, represented in the range 0-127.

REMI shares the use of time signature and tempo data, containing bar events at the start of every bar, and sixteen evenly spaced beat events per bar. It is worth noting that there are four beat events per musical beat; this is to facilitate the representation of notes which do not occur precisely on a musical beat.

REMI also stores note events in a separate way from MIDI. In REMI, a note is comprised of three events: note pitch, note duration, and note velocity. Note pitch contains the pitch of the note, while note velocity contains the velocity. Note duration contains the length of the note in number of REMI beats; for example, a note which lasts one beat in a

common time bar would be represented by the number 4. In MuseMorphose, a slightly altered implementation of REMI is used, which multiplies the duration by 120 (Wu and Yang 2022).

The final difference between MIDI and REMI is that REMI contains chord events at the start of each bar, to represent the underlying harmony in each bar of music.

VIII. Gap Analysis

A background to several research areas has been covered in this section. This dissertation brings together these research areas to close a gap in the existing literature concerning sentiment transfer in symbolic music by drawing from all these areas. An understanding of music theory and musical attributes aids in their implementation in MER tasks, which has been found to improve the accuracy of emotion recognition (Yang and Chen 2012). Existing MER approaches are summarised, as this dissertation will be building on these approaches to attempt sentiment transfer. Specific existing tools which demonstrate success in similar applications have been discussed, with a particular focus on MuseMorphose, which will form the basis for the program developed in this dissertation (Wu and Yang 2022).

2. Problem Statement

I. Problem Definition

As outlined in Section 1, there have been several MER projects concerned with using NLP machine learning architectures to achieve improved results in classifying symbolic music by sentiment. There have also been applications of style transfer models (such as GANs and VAEs) to achieve symbolic music style transfer, and one attempt at sentiment transfer by valence. However, there has not yet been a program developed which approaches the task of sentiment transfer across the four sentiment quadrants commonly identified in MER. The closest attempt at time of writing attempts sentiment transfer only across the

valence plane and has not yet published results (Sigel et al. 2021). As such, this dissertation is concerned with closing this research gap.

MuseMorphose proves that style transfer can be controlled through the manipulation of musical attributes (Wu and Yang 2022), and there is a strong relationship between some musical attributes and emotion (Panda et al. 2023). In addition, Sigel et al identify in their research that the consideration of musical features may strengthen their model's capacity for sentiment transfer (2021). These factors in combination lead to the central research question for this dissertation: can symbolic music sentiment transfer across Russell's four quadrants be achieved through musical attribute extraction and control?

II. Primary Goal

It is beyond the scope of this dissertation to train a new model from scratch. Therefore, a suitable existing model must be identified which can be adapted to fit this project's goal. Sentiment transfer in symbolic music can be regarded as a style transfer task, therefore a model for style transfer is the most appropriate choice to make. Given the intention to base this transfer on attribute extraction and control, MuseMorphose is the most appropriate choice of a model on which to build from those reviewed in the background (Wu and Yang 2022).

With the decision made to use the MuseMorphose architecture as the base of this project, the approach becomes more concrete. Since MuseMorphose extracts attributes from symbolic music and then manipulates those attributes in generation, this project will attempt sentiment transfer by identifying attributes that have a strong and independent correlation with perceived sentiment and controlling those attributes during symbolic music generation. The success of this approach will be analysed by measuring how often a generated piece of symbolic music is accurately classified as representing its target sentiment. Analysing the quality of generated symbolic music by this model is beyond the scope of this dissertation but

would be an interesting area for potential future research.

III. Supporting Secondary Goals

To approach the primary goal, secondary supporting goals can be identified which aid in dividing the project into more manageable steps. First, a suitable dataset of sentiment annotated MIDI data will need to be selected so that it can be used as input for sentiment transferred generations. Next, attributes suitable for sentiment transfer must be identified. Attributes will be selected for their correlation to variances in perceived sentiment. Once attributes have been selected, a means to extract them from input MIDI data will need to be created. MuseMorphose's architecture will need to be modified and the model will need to be retrained so that it can support the addition of these new attributes. Having retrained the model, sentiment transferred symbolic music will then be generated from the sentiment annotated dataset previously selected. The generated data will be analysed to determine how accurately each sample reflects its target sentiment. Finally, these results will be analysed and discussed.

IV. Methodology

The methodology of this dissertation involves selecting an appropriate sentiment annotated dataset from which to generate sentiment transferred samples; selecting attributes with a clear association with music emotion perception to consider in music generation; implementing programmatic extraction of these attributes from symbolic music and adding these methods to the MuseMorphose model; retraining the MuseMorphose model to be able to use these methods in new music generation; and finally assessing the accuracy of the sentiment transfer through using an existing MER tool to classify the generated music by sentiment. Each of these stages will be described in more detail in the following sections.

a. Sentiment Annotated Dataset Selection

To generate sentiment-transferred symbolic music, the model will require sentiment

annotated symbolic music input. To this end, an appropriate sentiment annotated dataset of symbolic music must be selected. As described in the background portion of this dissertation, the dataset selected will be informed by the selected emotion classification method. For this project, a dimensional categorisation approach based on two dimensions has been selected as it is more standardised than categorical approaches, which vary from project to project. Given these limitations, there are two publicly available datasets suitable for this project: VGMIDI, which contains 200 sentiment-annotated MIDI representations of music from video game soundtracks (Ferreira and Whitehead 2021); and EMOPIA, which contains 1,087 sentiment-annotated MIDI clips of piano covers of popular songs (Hung et al. 2021). A relevant recently developed dataset of expert-annotated Bach chorales was recently created by Grekow; however, the dataset does not appear to be publicly available (2023).

b. Attribute Selection

To select which attributes to control for sentiment transfer, attributes which have a strong correlation with perceived sentiment need to be identified. To achieve this goal, a review of the relationships between attributes and perceived sentiment will be conducted, to identify attributes which have a straightforward correlation with perceived sentiment. This will be conducted by first reviewing which attributes are identified in Panda et al.'s review as having consistent associations with given sentiments (Panda et al. 2023). Then, for each identified attribute, a function for extracting said attribute from a REMI file will be developed. These functions will be developed in Python and designed to match the output format of existing MuseMorphose attribute extraction functions. Both decisions have been taken with the aim to minimise the change in complexity of the model resulting from the introduction of new attributes.

Once all attribute extraction functions have been developed, samples from the selected sentiment annotated dataset will have the selected attributes extracted. Then, using

standard Python statistical analysis libraries (such as Pandas, NumPy, and Matplotlib), the distribution of attribute values across different sentiment quadrants will be determined. This process will enable identification of which selected attributes have clear differences in distribution across different sentiment quadrants.

c. Model Architecture Modification and Retraining

Having identified appropriate attributes and developed functions to extract them from the input REMI data, the next step of the project will be to modify the MuseMorphose architecture so that it can support these new attributes, and then to retrain the model on the updated architecture. The training will be done on an Nvidia V100 Tensor Core GPU, accessed remotely via Google Colab. While the university provides students access to computers with high-powered GPUs, limited memory allocation per student user presents difficulties in using these computers for model training purposes.

The initial training will be conducted across 60 epochs, to match the training process of the original MuseMorphose paper (Wu and Yang 2022). However, given the changes made to the model, it may be that best results occur after more or less training steps; for that reason, further training may be conducted if predicted improvements in results does not outweigh the computational costs. The training step which achieves the best balance between a low Kullback-Leibler divergence loss and a low reconstruction loss will be selected and the checkpoints from that training step stored.

d. Generation Modification and Results Analysis

After completing model training, output will need to be generated so that the accuracy of the model can be assessed. To generate new files based on sentiment, the MuseMorphose generation method will need to be updated. In the implementation provided in the code accompanying Wu and Yang's paper, attributes are randomly modulated by +/- 3 attribute classes (2022). So that this method can be applied to sentiment transfer, it will be modified

such that attributes are modulated by a fixed amount depending on the difference in average attribute class between the input sentiment and the target sentiment.

Using this modified generation modification process, three new MIDI pieces will be generated per piece in the selected dataset: one for each sentiment quadrant, except for the input sentiment quadrant. These generated samples, along with the original samples from which they were generated, will then be passed through the EMOPIA classification tool. By feeding the original samples through a classification tool, a baseline accuracy will be identified; then, the classification accuracies for different subsets of the generated data can be compared to the classification tool's accuracy on real data. The accuracy will be measured by target quadrant, as well as by unique transfers (e.g. Q1 transferred to Q2 considered separately from Q3 to Q2) and by transfers across only one dimension (e.g. high arousal to low arousal) so that any variations in accuracy can be identified.

3. Features of MuseMorphose and Potential for Extension

I. Introduction

In this section, the MuseMorphose model is discussed with a particular focus on its potential for extension (Wu and Yang 2022). This is undertaken to demonstrate that the MuseMorphose model is suitable for extension with an aim to approach the task of sentiment transfer, and to highlight ways in which this extension will be approached.

II. MuseMorphose Architecture

MuseMorphose is a Transformer-based VAE model for conducting symbolic music style transfer through attribute control (Wu and Yang 2022). First, bar-level attributes are extracted and classified into one of eight discrete classes. These bar-level attributes, along with bar-level latent conditions, are passed into the model's Transformer encoder as segment-level conditions. The in-attention Transformer decoder then considers the input as a whole while frequently referring back to the segment-level conditions. This asymmetry is how the

model achieves both fine-grain attribute control and long-term coherence.

The proposed extension of this model for this dissertation concerns only modifying the attributes which are being extracted from the input data, and minimally changing the architecture of the model itself to support these newly added attributes. Many hyperparameters for the model are stored in a YAML configuration file, as shown in this extract:

```
model:
  enc_n_layer:      12
  enc_n_head:       8
  enc_d_model:      512
  enc_d_ff:         2048
  dec_n_layer:      12
  dec_n_head:       8
  dec_d_model:      512
  dec_d_ff:         2048
  d_embed:          512
  d_latent:         128
  d_polyph_emb:     64
  d_rfreq_emb:      64
  cond_mode:        in-attn
  pretrained_params_path:      null
  pretrained_optim_path:       null
```

In this sample, `d_polyph_emb` and `d_rfreq_emb` refer to the embedding dimensions for the two attributes. Here, adding support for new attributes only requires adding consistently named embedding dimensions for new attributes; for instance, if volume was an attribute being controlled, it could be referred to as `d_volume_emb`, and given a value of 64 to maintain consistency with the existing attribute embedding dimensions. Implementing these changes to the model itself, however, involves updating values in several more places. Consider the following sample code of the MuseMorphose class:

```
class MuseMorphose(nn.Module):
  def __init__(self, enc_n_layer, enc_n_head, enc_d_model, enc_d_ff,
    dec_n_layer, dec_n_head, dec_d_model, dec_d_ff,
    d_vae_latent, d_embed, n_token,
    enc_dropout=0.1, enc_activation='relu',
    dec_dropout=0.1, dec_activation='relu',
    d_rfreq_emb=32, d_polyph_emb=32,
    n_rfreq_cls=8, n_polyph_cls=8,
```

```
    is_training=True, use_attr_cls=True,
    cond_mode='in-attn'
  ):

    # 42 lines of code omitted for brevity

    self.d_rfreq_emb = d_rfreq_emb
    self.d_polyph_emb = d_polyph_emb
    self.rfreq_attr_emb = TokenEmbedding(
      n_rfreq_cls, d_rfreq_emb, d_rfreq_emb
    )
    self.polyph_attr_emb = TokenEmbedding(
      n_polyph_cls, d_polyph_emb, d_polyph_emb
    )
```

In this sample, there are multiple references to these attribute embedding values; to extend the model to support additional attributes, variables for the new attributes must be added at every point where the existing attribute variables are used. Although having to make these changes manually increases the potential for human error, the variables are named consistently and meaningfully, making it more straightforward to find references to them and add new variables as needed.

III. Processing of Existing Attributes

To facilitate the addition of new attributes to MuseMorphose, the methods for extraction and the ways they are represented will aim to maintain congruity with those already included in MuseMorphose. The new attributes added will be selected for their associations with changes in perceived sentiment, then functions for extracting these attributes and means to handle them in the model itself will be added to MuseMorphose.

The existing attributes in MuseMorphose are both considered at bar-level and organised into eight numerical classes. For instance, in the existing attribute of polyphonicity, the more notes there are playing at any given point, the higher the mean notes per sub-beat for that bar will be. The bar is then assigned a polyphonicity class (a number in the range 0-7 inclusive) depending on how high the mean is; the higher the mean, the higher the bar's class. Boundaries between these eight classes were determined so that the distribution between the

classes was as even as possible, and these boundaries were provided as variables in the code itself.

There are a couple of ways the approach used in attribute extraction lends itself well to extension, which can be understood by looking at the code itself.

```
for p in pieces:
  polyph_raw = np.reshape(
    compute_polyphonicity(events, n_bars=len(bar_pos)), (-1, 16)
  )
  rhythm_raw = np.reshape(
    get_onsets_timing(events, n_bars=len(bar_pos)), (-1, 16)
  )


  polyph_cls = np.searchsorted(polyphonicity_bounds,
np.mean(polyph_raw, axis=-1)).tolist()
  rfreq_cls = np.searchsorted(rhym_intensity_bounds,
np.mean(rhythm_raw, axis=-1)).tolist()
```

In this sample code, taken from the attribute extraction program within MuseMorphose, `compute_polyphonicity` and `get_onsets_timing` are functions which return 1-dimensional arrays with a value that correspond to every sub-beat within the piece (Wu and Yang 2022). These arrays are then reshaped to be 2-dimensional arrays with each second level array corresponding to a single bar in the piece. This facilitates getting a bar-level mean for each attribute, which can be stored for later use in the project.

The consistency of this approach enables the addition of new attributes, as it is clear how attributes should be formatted. However, the code could be made more readable by abstracting the approach shown above and minimising repetition. This change would also greatly increase the simplicity of extracting additional attributes at this stage of the project, although the manual changes discussed previously would still need to be made. Below is an example of how the code shown above may look with these changes implemented, and the relevant data for each attribute stored in a dictionary. Additionally, the boundaries for each

attribute are updated to look for relevant data in a YAML configuration file, rather than being hard-coded.

```
all_attrs = {
   'polyph': {
      'bounds': config['attribute_bounds']['polyphonicity'],
      'function': compute_polyphonicity
   },
   'rhythm': {
      'bounds': config['attribute_bounds']['rhythm_intensity'],
      'function': get_onsets_timing
   }
}
```

```
for p in pieces:
  for attr in all_attrs:
    attr_raw = np.reshape(
      all_attrs[attr]['function'](
        events, n_bars = len(bar_pos)
      ), (-1, 16)
    )

    attr_cls = np.searchsorted(
      all_attrs[attr]['bounds'], np.mean(attr_raw, axis=-1)
    ).tolist()
```

With these changes made, adding an additional attribute to the attribute extraction process becomes significantly simpler, requiring only a new attribute extraction function to be written, and the relevant data to be added to the attribute dictionary and the configuration YAML file.

IV. Training Process

Once extraction methods for new attributes have been developed, and the architecture of the model has been updated to be able to operate on these attributes in generation, the model will need to be retrained to work with the additional attributes.

The MuseMorphose model is based on a Transformer encoder paired with a Transfer decoder, with a Kullback-Leibler divergence regularised latent space in between the two (Wu

and Yang 2022). A Transformer encoder encodes an input sequence to a sequence of continuous representations, along with positional encodings, which then provide context to the decoder in constructing output data (Vaswani et al. 2023). Both the encoder and the decoder contain attention mechanisms, which enable the model to consider the surrounding context of any point in the sequence when determining a suitable output. Both the encoder and the decoder contain multiple layers; in the original proposal, 6 layers were used. MuseMorphose combines this Transformer architecture with a Kullback-Leibler divergence regularised latent space to represent data about attributes extracted from the music (Wu and Yang 2022).

For MuseMorphose, training the model took place over two broad steps; first, the model (particularly the Transformer decoder) was conditioned to determine the best architecture and hyperparameters to use. These best results were then implemented, and the updated model was trained on a dataset more relevant to the program's purpose of generating solo piano symbolic music.

The initial training phase for MuseMorphose involved using a portion of the LPD-17-cleansed dataset to conduct model conditioning (Dong and Yang 2018; Wu and Yang 2022). The LPD-17-cleansed dataset contains over 20,000 songs, and the subset used for training MuseMorphose consisted of 10,026 songs, selected with the parameters that a piano was present for at least half of the song, and the song's time signature was 4/4 (Wu and Yang 2022). This subset contained 650 hours of music, and training with this dataset took 2 days to complete 20 epochs. The purpose of this training stage was to identify which transformer decoder model provides the best results in the context of generating symbolic music, and to select hyperparameters for further training. The training was conducted with different conditioning mechanisms applied in the model (pre-attention, in-attention, and post-attention conditioning), to determine which conditioning mechanism yielded the best results. The

results indicate that in-attention conditioning provides the best results for symbolic music generation. This conditioning was carried out on the transformer decoders before introducing the use of bar-level attributes; for this reason, along with the amount of computational resources required for the conditioning, further model conditioning will not be carried out for this project.

The purpose of the next stage of training was to fine-tune the model for solo piano symbolic music generation, as opposed to being trained more generally on a larger dataset. For the purposes of this project, this stage of training will need to be conducted from scratch so that the model can correctly work with newly added attributes. Early experiments with retraining the model used the EMOPIA dataset to train the model, which is smaller than the AILabs.tw-Pop1k7 dataset. This decision was made under the belief that training on the AILabs.tw-Pop1K7 dataset would take several days, following a misinterpretation of the information presented in the MuseMorphose paper; the paper states that training over 20 epochs took 2 days, however this information only relates to conditioning the model on the LPD-17-cleansed dataset, with no information provided on time taken to train the model on the AILabs.tw-Pop1k7 dataset. Once this misunderstanding was identified, further model training tasks were undertaken using the AILabs.tw-Pop1k7 dataset, since it contains significantly more data than the EMOPIA dataset (108.8 hours' worth of music, as opposed to 12.0 hours of music) (Chou et al. 2021). Training with the AILabs.tw-Pop1k7 dataset does take longer than the EMOPIA dataset, with 60 epochs on the EMOPIA dataset taking approximately 4.5 hours, and training on the AILabs.tw-Pop1k7 dataset taking approximately 7.5 hours. However, 7.5 hours is still substantially less than 48 hours.

V. Generation

The final portion of MuseMorphose which will be modified for this project is the symbolic music generation functionality. In its existing state, symbolic music is generated by

selecting a random file from the dataset specified in the configuration file. Then, each bar-level attribute is randomly offset by +/- 3 attribute classes (with the same offset applied across the whole input sequence), and a new piece is generated by passing the input piece and the new target attributes into the MuseMorphose model. The resulting symbolic music is then saved in both REMI and MIDI format. This process is carried out on a number of pieces specified by the user and repeated for each piece a number of times specified by the user (Wu and Yang 2022).

To achieve sentiment transfer through attribute control, randomly determining an amount by which to offset attributes is not a suitable approach. Instead, the program will need to be able to identify the sentiment of the selected symbolic music being used as a basis; it will need to be able to control the offset of attributes depending on the input and target sentiments; and it will need to repeat this process so that for any input, symbolic music that conveys each sentiment other than the input sentiment is generated.

First, the function will be updated to identify the sentiment of the input piece. While this could be achieved by the addition of a separate MER tool concerned with emotion classification, such an approach would fall outside the scope of this project. Since the samples from which output will be generated will already be classified by sentiment, this information will be stored in the input's filename and read directly from there.

The next change to the existing symbolic music generation concerns determining the amount by which to offset each attribute. The method implemented in MuseMorphose is to determine offset by a randomly generated number. A random number is generated in a given range (default -3 to 4) for each new sample to be generated. This number represents the amount by which each bar-level attribute will be offset. To enable control over the offset of attributes, the random offset function will be replaced with one in which the amount to offset each attribute is passed in as a variable. This function can be used in the generation of any

sentiment, using an offset value that depends on its starting and target sentiments. The exact values for this will be determined by conducting data analysis on the average attribute classes for attributes across different sentiments.

Finally, the program will need to be updated so that the process of offsetting attributes and generating new symbolic music is conducted for every sentiment quadrant except the input sentiment. In MuseMorphose, the user can specify a number of samples to generate from an input piece where each time, a new offset amount is generated. This offset amount is then added to each bar-level attribute class in the input piece, and from the newly generated attribute classes, an output is generated. To maintain user control, the ability to specify a number of samples to generate for each piece will be retained; however, instead of generating one new sample each time the programme runs, it will generate three: one for each sentiment quadrant besides that of the input piece.

4. Extending MuseMorphose for Sentiment Transfer

I. Introduction

While the previous section outlines how MuseMorphose is suited to extension, and details ways in which it is extended or modified to achieve sentiment transfer, this section focusses more specifically on the selection and implementation of new attribute extractions, the processes and results from retraining the model with the addition of new attributes, and how offset of these attributes is controlled in the generation of sentiment transferred music.

II. Dataset Selection and Issues

As noted earlier, there are two publicly available sentiment-annotated symbolic domain datasets which would be applicable for use in this project: VGMIDI (Ferreira and Whitehead 2021), which contains 200 sentiment-annotated MIDI files representing piano arrangements of music from video game soundtracks; and EMOPIA (Hung et al. 2021), which contains 1087 sentiment-annotated MIDI files representing solo pop piano

performances. Both datasets have pros and cons. VGMIDI features significantly more datapoints per piece than EMOPIA, giving the ability to track changes in sentiment throughout the duration of a piece; however, this data is presented in JSON format with an individual object for data from each participant who provided sentiment annotation. This increases the amount of pre-processing required before the dataset can be meaningfully used in this application. EMOPIA, meanwhile, only provides a single sentiment for each piece, presented as a simple label (Q1, Q2, Q3 or Q4 for high arousal high valence, high arousal low valence, low arousal low valence, or low arousal high valence respectively). The reason change in sentiment over time is not measured is that pieces in the EMOPIA dataset are split into emotionally consistent short segments, averaging 14.8 bars of music per piece (Chou et al. 2021).

For this project, EMOPIA is a more suitable choice for several reasons. The foremost of these reasons is the greater number of samples per quadrant, which means more output samples can be generated and the accuracy of the sentiment-dependent generation can be better assessed. Additionally, the EMOPIA dataset contains files already in the correct format for this project. While a MIDI-to-REMI method is provided in the public repository associated with the paper in which REMI was first described, using that method results in a slightly different format than the one used in the AILabs.tw-Pop17k dataset (Huang and Yang 2020). The process used requires beginning from audio domain music and going through several stages of processing. The REMI files presented in the EMOPIA dataset are presented in the format achieved from using this process, making it immediately suited to use in MuseMorphose. The REMI files are provided for 1064 of the 1087 samples in the EMOPIA dataset. These REMI files in the EMOPIA dataset will be used as the input for this project.

III. Review of Attributes

a. Method

In their review of musical attributes and their relationships to perceived emotions, Panda et al. provided tabular displays of attributes alongside associated emotion categories from various sources about the perception of emotion in music (2023). To understand how these relate to Russell's four quadrants, these categorical emotions have been translated to quadrants based on the categorisation of language associated with specific emotions given in Russell's paper (1980). Q1 refers to high arousal, high valence pieces; Q2 to high arousal, low valence; Q3 to low arousal, low valence; and Q4 to low arousal, high valence. These are the same quadrant labels used in the EMOPIA dataset (Hung et al. 2021).

b. Melodic Elements

Attribute	**Associated Quadrants**
High pitch	Q1, Q4 (high valence)
Low pitch	Q2, Q3 (low valence)
Large pitch variation	Q1 (high arousal)
Small pitch variation	Q3, Q4 (low arousal)
Wide pitch range	Q1, Q4 (high valence)
Narrow pitch range	Q2 (low valence)
Ascending melodic contour	Q1, Q2 (high arousal)
Descending melodic contour	Q3, Q4 (low arousal)

Table II: quadrants of emotions perceived associated with melodic musical attributes.

The table above represents most data on melodic elements and their relationship to perceived sentiment as found in Panda et al.'s review (2023). Pitch variation refers to the number of unique pitches in a piece of music, while pitch range refers to the distance between

the lowest and highest notes in the piece. This table omits the relationship between specific melodic intervals and perceived sentiment, as the data in the review is not representative of all possible melodic intervals, which presents issues in how to categorise such intervals. This table also omits data on melodic motion (whether concurrent melody notes tend to be sequential in the scale of the underlying tonality or not) as this attribute always falls into one of three categories: all melodic notes move stepwise (to an adjacent tone), all move skipwise (not to an adjacent tone), or a melody contains a combination of the two. This shape of data would not be suitable to map to eight categories, as would be required to maintain congruity with the shape of the original MuseMorphose attributes.

It can be seen from the above table that different melodic traits have clear associations with different emotional dimensions; pitch height and pitch range are both associated with valence, while pitch variation and melodic contour are both associated with arousal. For each dimension, there is one element which is associated with sentiments across both quadrants of each plane; these two attributes (pitch height and melodic contour) will be selected for development and further testing. It is important to note that both attributes are relevant only to the melody of the piece, and not every note in the piece; for that reason, a melody extraction method will also be created.

c. Harmonic Elements

Consonant harmonies	Q1, Q4 (high valence)
Dissonant harmonies	Q2, Q3 (low valence)
High-pitched harmonies	Q1
Low-pitched harmonies	Q4
Major mode	Q1, Q4 (high valence)
Minor mode	Q2, Q3 (low valence

Table III: quadrants of emotions perceived associated with harmonic musical attributes.

The table above represents data from Panda et al.'s review on harmonic elements and relationship to perceived sentiment (2023). There is one omission, concerning whether the harmony implies an underlying tonality or not. This has been omitted as the data is separated into tonal, atonal, or chromatic (using a scale comprising all 12 tones). This results in a scale between purely tonal music (where all notes present fit within the underlying scale of the piece) and purely atonal music (where the distribution of notes and harmonies is such that an underlying tonality cannot be identified). However, chromatic harmonies can be present in a piece without the piece being fully atonal. This adds a complexity that makes the feature less suitable for use in this project. Of the three attributes shown above, two indicate a relationship with perceived valence, while variation in harmonic pitch height seems to result in a shift in both arousal and valence. Given the time scale of this project, a method for extracting the consonance of harmonies will not be developed, as it would likely be more complex than extracting either of the remaining two attributes.

d. Rhythmic Elements

Fast tempo	Q1, Q2 (high arousal)
Slow tempo	Q3, Q4 (low arousal)
Fast tempo and short notes	Q1, Q2 (high arousal)
Slow tempo and long notes	Q3, Q4 (low arousal)

Table IV: quadrants of emotions perceived associated with rhythmic attributes.

The table above shows only two of the four rhythmic elements identified by Panda et al in their review (2023). The first of the omitted rhythmic elements is rhythmic type, which is unsuitable for this application as it is sorted categorically without a clear definition for each category. The categories identified are “regular/smooth,” “irregular/rough,” “complex,” “varied,” “firm,” and “flowing/fluent.” While the regularity of a rhythmic pattern could be given a position on a scale, there would still be an issue with the perceived sentiments themselves; regular rhythms have associations that cover three of the four quadrants (Q1, Q2, and Q4), and irregular rhythms are associated with two of those three quadrants (Q1 and Q2). The other omitted attribute concerns the use of rests (or absence of notes) in the piece; however, the emotive effect of this attribute is dependent on co-occurring harmonic elements (Panda et al. 2023).

Tempo and note density (short notes at a high tempo or long notes at a low tempo) seem to both have a straightforward association with arousal. The latter attribute is already integrated into the MuseMorphose model, described as “rhythmic intensity” (Wu and Yang 2022). It was selected for its straightforward relationship with perceived arousal and for its ability to be perceived by a casual listener (Wu and Yang 2022). Analysis on this existing attribute will be conducted to confirm its relationship to sentiment in the EMOPIA dataset. Given the similarity in perception between the two attributes, tempo will not be developed

further.

e. Remaining Musical Attributes

Of the remaining musical attributes discussed by Panda et al, there are few which stand out as suitable for use in this project. There are three contributing factors to this. Firstly, for attributes relating to expressivity (which covers articulation, ornamentation, and presence of vibrato), none were found to be suitable for use in this project. Articulation (which in this paper refers to whether a note is long and overlaps with other notes, or short and detached) tends towards one extreme or the other; notes are predominantly attached or predominantly detached. Ornamentation concerns specific musical gestures, which are not common in pop music, and vibrato is not possible on a piano.

Secondly, few remaining attributes have relationships with valence and arousal that can be as simply described as those outlined above; for example, monophonic samples are identified as being perceived as "happier" than homophonic pieces, but homophonic pieces are also identified as being perceived as happier than monophonic pieces (Panda et al. 2023).

Finally, textural information does not contain any attributes with clear sentiment associations besides polyphonicity, which is already extracted in MuseMorphose. Polyphonicity refers to how many layers or simultaneous musical lines a piece has, with more layers being associated with higher valence (Panda et al. 2023). This attribute's relationship to sentiment in the EMOPIA dataset will be analysed and, if significant differences across quadrants are found, it will be included in the generation of sentiment-conditioned symbolic music.

IV. Development of Attribute Extraction Methods

a. Melodic Extraction

To be able to conduct analysis on melodic elements, the melody from a piece of music must first be extracted. This is a task which has been covered before, with

Uitdenbogerd and Zobel making a significant contribution through their development of a skyline algorithm which takes the highest note at any given point in a MIDI piece and considers it part of the melody (2002). At times, note durations are shortened if the next onset occurs before the end of the previously identified melodic note.

An issue found with this algorithm is the inclusion of harmonic notes with unique onset times; to demonstrate this issue, consider the following extract from Mozart's Fantasy in D Minor:

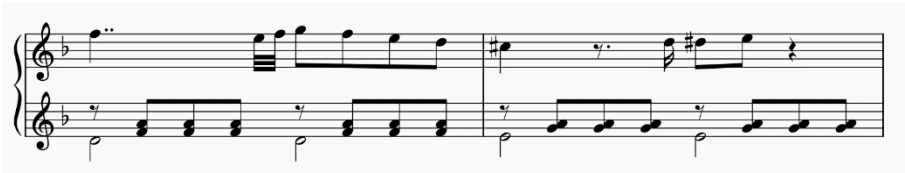

Figure III: Fantasia in D Minor by W.A. Mozart, bars 12-13

All notes are depicted by circles positioned on the horizontal lines, with notes that are vertically aligned occurring simultaneously. The notes are read from left to right. In this extract, the melody notes are all on the upper stave (the higher set of five lines), while the accompaniment notes are in the bottom stave (the lower set of five lines). However, the algorithm as proposed by Uitdenbogerd and Zobel would yield this melody:

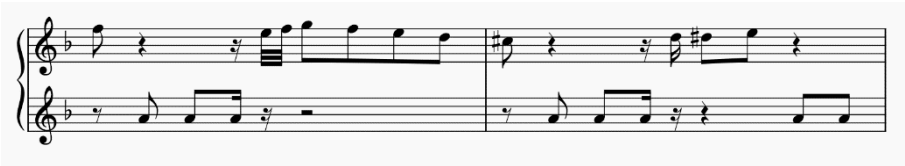

Figure IV: Melody extracted from Fantasia in D Minor by W.A. Mozart bars 12-13 using the skyline method.

In this melodic extraction, the sustained first note is shortened to include notes from the lower accompaniment (indicated by the addition of rests, the short horizontal lines in between notes), and non-melody notes are also included in the second bar of music. This distorts both the rhythm and the contour of the melody. To avoid this effect, an alternative

approach is to take a first melodic note and skip ahead to the note's end point, not considering any notes that occur in between. While this does not achieve perfect melodic extraction, as non-melody notes that occur while there is a rest in the melody will still be included, and in some cases overlap with the next melody note, it avoids the problem of sustained notes being shortened to include notes from the accompaniment.

To create this algorithm, first the existing REMI structure was modified; rather than three distinct events creating an event, these events were replaced with a single object, created from a new Note class which contained the note's pitch, velocity, and duration. The updated events list, containing single note events instead of three note events per note, is then fed into the melodic extraction function, which iterates through the modified event list, passing all non-note events to a new list of REMI events, and adding note events only if they meet the requirements of the skyline algorithm defined above.

```
FOR event in events list:
        IF event is a beat and duration of last note has
        elapsed:
                Append note with highest pitch in
                notes_in_beat to output list
                Clear notes_in_beat
        IF event is a note:
                Append note to notes_in_beat
                Continue
        Append event to output list

RETURN output list
```

Pseudocode algorithm for Skyline melody extraction method

In this approach, each time a melody note is found, it is added to a new REMI event list containing only melody notes, and a new melody note is not checked for until the previous melody note has run its full duration. At the next beat after the end of the previous melody note, all notes are tested and the highest is selected as the next melody note.

This function gets the pitches from all notes passed into it and returns the note with

the highest pitch. The resulting REMI list of events from the melody extraction function can then be passed forwards into melodic feature extraction methods.

b. Melodic Pitch Height

Determining melodic pitch height is one of the more straight forward attributes to extract, once the melody has been isolated. Since all attributes are considered at the bar level, for each bar, an average height of melodic pitch notes can be determined. This is achieved through adding the pitch of each melodic note to a list, and at the end of each bar, adding the average of all pitches to a list.

```
FOR event in melodic events list:
        IF event == bar:
                Append mean of all pitches in bar to
                output list

RETURN output list
```

Pseudocode for pitch height extraction

This is a relatively straightforward approach. Some complexity, however, is introduced in cases when there are bars which do not contain any melody notes. One approach could be to pass a value of `None` in these cases, however this may pose issues when the attribute data is passed on to the model, as this would not match the input data format the model expects from classified attributes. Another option would be to provide an average of zero, but depending on where boundaries are identified in between classes, this could result in bars where there are no melody notes being considered categorically the same as bars where all melodic notes are low in pitch. It also means that decreasing the attribute class for melodic pitch height may result in the loss of melodic notes instead of a lower average pitch.

The code implemented in this project simply does not store any data for bars which do not contain any melodic notes. While this approach does not account for situations where there are bars without any melodic notes, this edge case is not one that occurs in either of the

datasets used in this project.

c. Melodic Contour

Extracting data about melodic contour from a sample of music is a more complex process than extracting melodic height.

The first attempt at extracting this attribute created an array of zeros, with a zero for each beat in the input piece. Then for each melody note, if the preceding note was higher than it, every list item between the melody note's position and the previous melody note's position was set to -1; if the preceding note was lower, every list item in between was set to 1; and if the two notes were the same, every list item was set to 0.

```
pos = 0
output list = [0 for beat in events list]
FOR event in melodic events list:
        IF event == beat:
                pos += 1
        IF event is a note pitch:
                IF note pitch > previous note pitch:
                        list[pos] = 1
                IF note pitch < previous note pitch:
                        list[pos] = -1
                ELSE:
                        List[pos] = 0

RETURN output list
```

First algorithm to determine melodic contour

While this conveyed melodic direction throughout each bar, from which contour could be gleaned, this method was not compatible with the requirement that attributes need to be categorisable into eight similarly sized classes, as bar-level averages were very heavily skewed towards 0, with less extreme biases towards averages of 1 and -1. A different approach attempted was to get the pitch of every melodic note in a bar and normalise the values to be centered around the average pitch of all melodic notes in the bar; for example, if the bar contained the notes [60, 62, 68, 61, 57, 70], an average of 63 would be determined,

and all values would have 63 subtracted from them, to result in [-3, -1, 5, -2, -6, 7]. The intention of this approach was to achieve, for each note, a distance from the average, in order to determine melodic direction. However, this approach still resulted in bar-level attributes heavily skewed towards zero.

At this stage, a new approach was considered. Goldstein et al were able in their melodic contour research to identify a number of common contour shapes: concave (descending then ascending), convex (ascending then descending), ascending, descending, and wavy (a combination of ascending and descending) (2023). By splitting the wavy contour into two shapes, depending on whether the melody ascends or descends first; and adding flat contour (all melodic notes are the same) and no contour (no melodic notes found), eight classes of melodic contour can be identified. These can also be ordered by valence by considering no/flat contour as neutral, and waves which ascend before descending as more positive due to the primacy effect (in order: descending; concave; wave which descends before ascending; no contour; flat contour; wave which ascends before descending; convex; ascending). Further experimentation in the attribute analysis and output analysis stages will need to be carried out to determine whether considering melodic contour in this way still has a relationship with sentiment, and whether the model is able to reconstruct data for this attribute. Below is the pseudocode of the function used to determine the melodic contour class, which is called from within another function to get melody notes by bar:

```
IF no notes in bar:
        RETURN no contour (class 3)
IF length of unique pitches in bar == 1:
        RETURN flat contour (class 4)
IF first note in bar is highest and last note in bar is lowest:
        RETURN descending contour (class 0)
IF first note in bar is lowest and last note in bar is highest:
        RETURN ascending contour (class 7)
IF all notes between first and last note are higher than first note and last note:
        RETURN convex contour (class 6)
IF all notes between first and last note are lower than first note and last note:
        RETURN concave contour (class 1)
IF lowest note occurs before highest note:
        RETURN descending-first wave contour (class 2)
IF highest note occurs before lowest note:
        RETURN ascending-first wave contour (class 5)
```

Algorithm of approach to get melodic contour type from notes in bar

This approach involves testing first for directionless contours, then for one-direction contours (ascending or descending), then bi-direction contours (concave or convex). Finally, having determined that the melody both ascends and descends, the sample is categorised into one of two wave shapes depending on whether it reaches its highest value or its lowest value first.

d. Harmonic Pitch Height

Having already written a function to extract melodic pitch height, extracting harmonic pitch height involves simply modifying the existing function to be able to handle concurrent notes. However, given the issues concerning handling cases where there are no melody notes in a given bar, and that melodic and harmonic pitch height have the same relationship to perceived sentiment (Panda et al. 2023), a single function was instead created to determine average pitch height at each bar of the input data. This is functionally identical to the previous melodic height function, but takes the full REMI events list as input instead of only the melody notes. While this means melodic notes are considered when determining the average harmonic pitch height, melody notes are a smaller sampling of notes than harmonic

notes; therefore, the resulting pitch height class will still be more heavily biased towards the harmony than the melody.

This function has the same issue as the melodic pitch height function of what to do in the trivial case of bars that do not contain any notes. In the code presented here, such a case is simply ignored, as it is unlikely to occur in most music, and has not presented an issue when tested on the EMOPIA dataset.

e. Tonality

There are two challenges to extracting the bar-level tonality from REMI. First, the underlying tonality of a bar needs to be determined; and second, a method to organise the extracted tonality into eight ordinal classes needs to be determined. A straightforward way to extract bar-level tonality from REMI is to look at the chord event within the REMI, which contains the root of the chord, the quality of the chord (major, minor, diminished, augmented, and whether a seventh is present), and the position of the chord. For each bar, the chord present at the start of the bar is considered to represent the underlying harmony for the bar. This decision is informed by the understanding that the first beat of a bar is perceived as stronger than other beats. Where a chord cannot be found or identified, REMI uses a chord event with the value "n_n" instead of a chord symbol to indicate this. Having decided how to identify the tonality of a bar, the next task is to order that information.

In Western music, there are seven "modes," or types of scale, that are commonly used.

Scale	Pitch class set	Sharps/flats required in C Major
Lydian	0, 2, 4, 6, 7, 9, 11	1 sharp
Ionian (major)	0, 2, 4, 5, 7, 9, 11	None
Mixolydian	0, 2, 4, 5, 7, 9, 10	1 flat
Dorian	0, 2, 3, 5, 7, 9, 10	2 flats
Aeolian (minor)	0, 2, 3, 5, 7, 8, 10	3 flats
Phrygian	0, 1, 3, 5, 7, 8, 10	4 flats
Locrian	0, 1, 3, 5, 6, 8, 10	5 flats

Table V: Pitch class sets of the seven commonly used modes in Western music.

Vincent Persichetti proposed a method of organising these modes from “brightest” to “darkest,” defined by the number of sharps or flats required to play the mode starting and ending on C (Persichetti 1961). This concept can also be applied to the chords used in REMI. The first consideration in the brightness of a chord is the quality of its third. The only difference between a major and a minor chord is whether the third scale degree (or the middle note of the triad) is natural or flattened; therefore, scales with flattened third degrees are perceived as darker than scales with natural thirds. Beyond the quality of the third scale degree, chords can be organised following Persichetti’s approach of defining chords with more sharps as brighter, and chords with more flats as darker (when the root of the scale is normalised to C).

Chord Quality	Pitch class set	Third degree Quality	Sharps/flats required in C Major
Augmented	0, 4, 8	Natural	1 sharp, 0 flats
Major/Major 7	0, 4, 7 (11)	Natural	0 sharps, 0 flats
Dominant 7	0, 4, 7, 10	Natural	0 sharps, 1 flat
Minor	0, 3, 7	Flat	0 sharps, 1 flat
Minor 7	0, 3, 7, 10	Flat	0 sharps, 2 flats
Diminished	0, 3, 6	Flat	0 sharps, 3 flats

Table VI: Pitch class sets and third qualities of chord qualities found in REMI

Additional to the chords outlined here, there are also sustained chords used in REMI – these are chords in which the third scale degree is replaced with either the fourth or second. Since there is no third scale degree, the chord cannot be easily identified as major or minor; therefore, it can be positioned in between the dominant 7 chord and the minor chord. The final possible chord value in REMI is “n_n” referring to no chord identified. This could either be a case where less than three notes are playing, or the underlying harmony cannot be accurately described by any of the chords available in the vocabulary. Therefore, chords of this quality will be positioned in between dominant 7 and minor again; but assumed to be darker than sustained chords, since monophonic lines are perceived as “more lonely” than polyphonic lines (Panda et al. 2023). Below is a table showing how the quality of each chord was arranged into classes for this project.

Chord Quality	Attribute Class
Augmented	0
Major or Major 7	1
Dominant 7	2
Sustained	3
No identified chord	4
Minor 7	5
Minor	6
Diminished	7

Table VII: Chord qualities and their corresponding tonality attribute classes

V. Attribute Analysis

To understand how to best use the extracted attributes to control sentiment during symbolic music generation, the relationships between attribute classes and perceived sentiment needs to be understood. By determining if there are significant differences in attribute class distributions across different sentiment quadrants, the relationship between the selected attributes and their affect on perceived sentiment can be confirmed. Additionally, an understanding of how to modify these attribute classes in sentiment transfer will be gleaned from being able to establish the difference in attribute classes across different quadrants. For instance, if it is found that Q1 tends to have higher attribute classes for polyphonicity than Q2, then generating a Q1 piece from a Q2 input would involve offsetting the polyphonicity classes by a positive number.

In aid of this, some data analysis has been carried out. First, the mean classes for all attributes across each quadrant was calculated. Then for each attribute, it was determined whether there is a significant difference in the mean attribute class across different quadrants.

The test used to determine if the difference is significant depends on whether both sets of data are normally distributed or not; therefore, the distribution of each attribute by quadrant must also be determined. The samples are all independent from each other. Therefore, when data is normally distributed, the independent samples T-test will be used; otherwise, the Mann-Whitney U test will be used. Since this data requires sentiment-labelled input, attributes are extracted from the EMOPIA dataset.

Throughout this section, all values are given to 4 significant digits.

a. Rhythmic Intensity

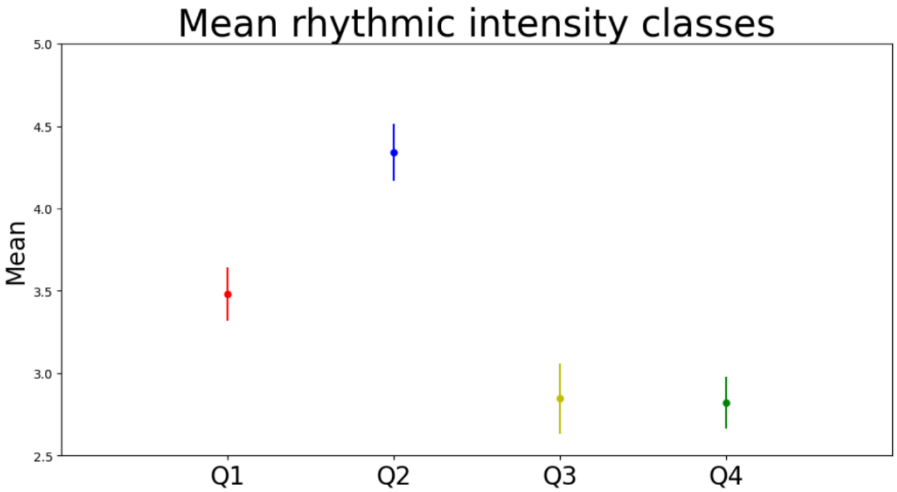

Figure V: A graph showing the mean rhythmic intensity class by quadrant.

The mean rhythmic intensity class for Q1 is 3.480; for Q2 is 4.340; for Q3 is 2.848; and for Q4 is 2.820. The classes are not normally distributed across any of the four quadrants. For this reason, the Mann-Whitney U test is used for all comparisons. The following table shows the difference in mean rhythmic intensity between pairs of quadrants where a significant difference was found.

	Q1	Q2	Q3	Q4
Q1	--	0.8595	0.6323	0.6603
Q2	0.8595	--	1.492	1.520
Q3	0.6323	1.492	--	No significant difference
Q4	0.6603	1.520	No significant difference	—

Table VIII: Significant differences in mean rhythmic intensity class by quadrant

These results indicate that rhythmic intensity does not differ significantly between low arousal quadrants. However, the two high arousal quadrants have significantly different rhythmic intensity mean values. This suggests that arousal plays a role in rhythmic intensity, with both high arousal quadrants being significantly different from the two low arousal quadrants. It also seems that valence plays a role in rhythmic intensity, as Q1 and Q2 are significantly different from each other, despite both being high arousal quadrants.

b. Polyphonicity

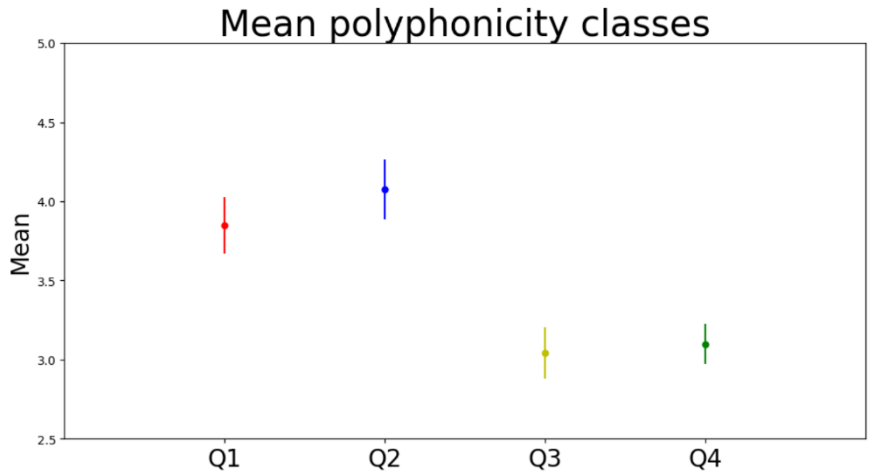

Figure VI: A graph showing the mean polyphonicity class by quadrant

The mean polyphonicity class for Q1 is 3.839; for Q2 is 4.077; for Q3 is 3.042; and for Q4 is 3.099. The classes are not normally distributed across any of the quadrants. The following table shows the difference in mean polyphonicity between pairs of quadrants where a significant difference is found.

	Q1	Q2	Q3	Q4
Q1	--	0.2276	0.8071	0.7503
Q2	0.2276	--	1.035	0.9780
Q3	0.8071	1.035	--	No significant difference
Q4	0.7503	0.9780	No significant difference	—

Table IX: Significant differences in mean polyphonicity class by quadrant

Similarly to rhythmic intensity, the low arousal quadrants are not significantly different from each other, while the high arousal quadrants are. A difference in this case is that while Q1 is significantly different from all other quadrants, the absolute difference between Q1 and Q2 is lower than the absolute difference between both of the low arousal quadrants. However, since there is still a significant difference between Q1 and Q2, it can be concluded that valence does still play some role in polyphonicity, as well as arousal.

c. Melodic Contour

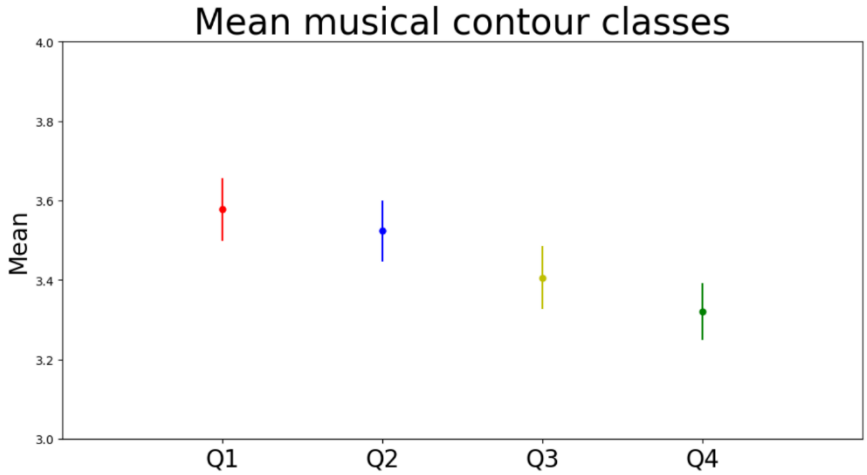

Figure VII: A graph showing the mean musical contour class by quadrant

The mean musical contour class for Q1 is 3.578; for Q2 is 3.523; for Q3 is 3.406; and for Q4 is 3.321. The classes are normally distributed across all quadrants except Q4; for this reason, the independent samples T-test is used when comparing classes from different quadrants, except when Q4 is one of the quadrants; in those cases, the Mann-Whitney U test is used instead. The following table shows the difference in mean musical contour between pairs of quadrants where a significant difference is found.

	Q1	Q2	Q3	Q4
Q1	--	No significant difference	0.1719	0.2569
Q2	No significant difference	--	0.1172	0.2023
Q3	0.1719	0.1172	--	No significant difference
Q4	0.2569	0.2023	No significant difference	—

Table X: Significant differences in mean melodic contour classes

Since there is no significant difference between Q1 and Q2, and there is no significant difference between Q3 and Q4, it can be concluded that arousal does not influence the melodic contour of a piece.

d. Pitch Height

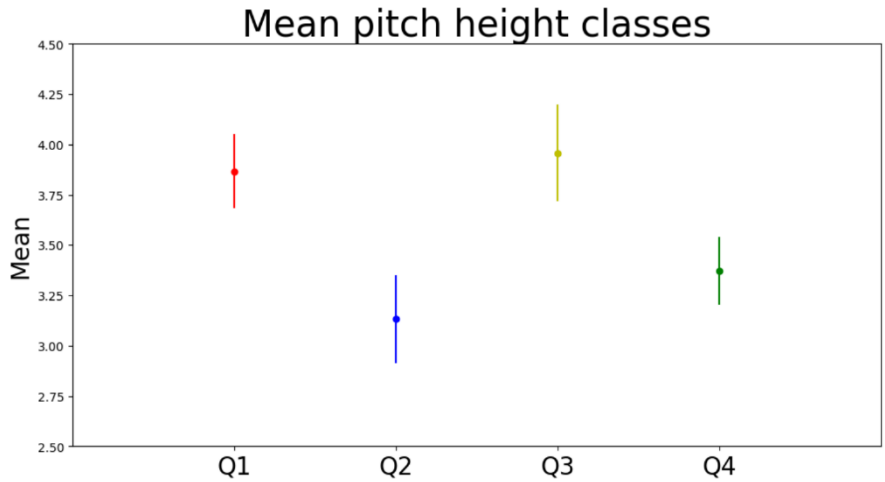

Figure VIII: A graph showing the mean pitch height class by quadrant

The mean pitch height class for Q1 is 3.867; for Q2 is 3.133; for Q3 is 3.957; and for Q4 is 3.371. The classes are not normally distributed across all quadrants except Q4; since T-tests require that both datasets be normally distributed, the Mann-Whitney U test is used for all comparisons. The following table shows the difference in mean pitch height between pairs of quadrants where a significant difference is found.

	Q1	Q2	Q3	Q4
Q1	--	0.7339	No significant difference	0.4957
Q2	0.7339	--	0.8244	No significant difference
Q3	No significant difference	0.8244	--	0.5862
Q4	0.4957	No significant difference	0.5862	--

Table XI: Significant differences in mean pitch height class by quadrant

The results indicate an unclear relationship between pitch height and changes in quadrant, as the two pairs of attributes with similar mean values contain attributes which vary by both arousal and valence. There does, however, seem to be some indication of relationship with perceived sentiment and pitch height. To better understand this, further analysis is undertaken.

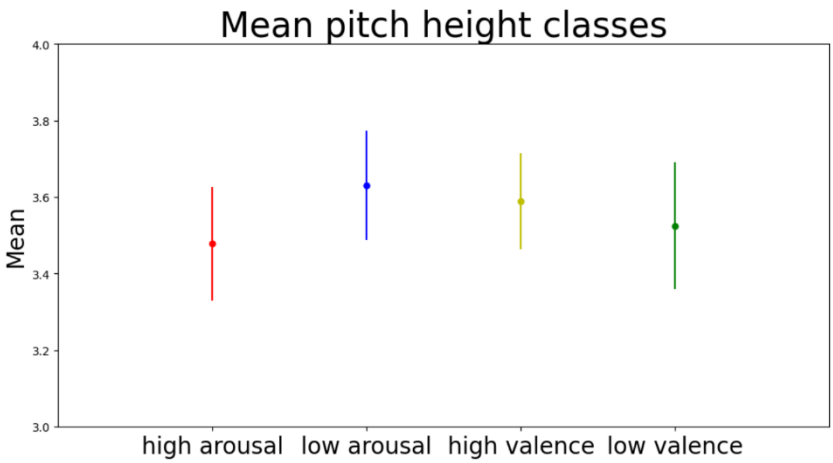

Figure IX: A graph showing the mean pitch height by plane

These results show that when grouped by plane, there is a less obvious difference between mean pitch height and attribute class. Further significant difference testing confirms that there is no significant difference between any of these four results. This seems to indicate that the relationship with pitch height and perceived sentiment depends on both the arousal and valence of the music.

e. Tonality

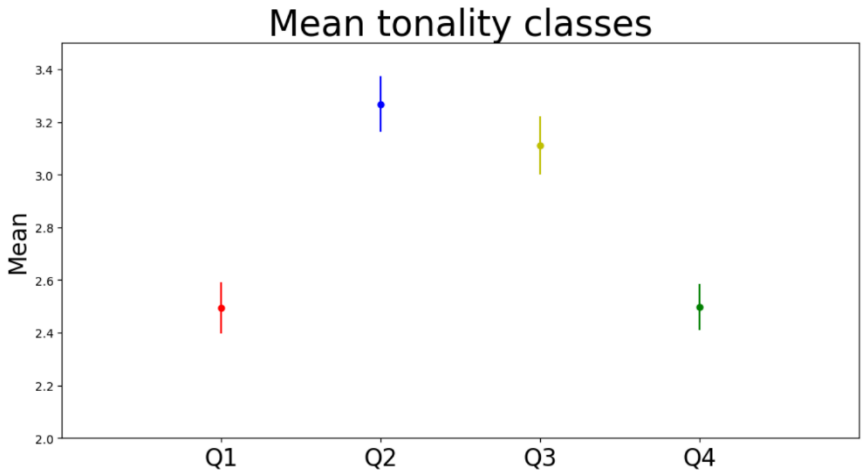

Figure X: A graph showing the mean tonality class by quadrant

The mean tonality class for Q1 is 2.494; for Q2 is 3.268; for Q3 is 3.112; and for Q4 is 2.497. The classes are normally distributed across all quadrants except Q2; for this reason,

the Mann-Whitney U test is used in all tests. The following table shows the difference in mean tonality class between pairs of quadrants where a significant difference is found.

	Q1	Q2	Q3	Q4
Q1	--	0.7746	0.6182	No significant difference found
Q2	0.7746	--	0.1565	0.7710
Q3	0.6182	0.1565	--	0.6145
Q4	No significant difference found	0.7710	0.6145	--

Table XII: Significant differences in mean tonality class by quadrant

There is no significant difference found between Q1 and Q4, which are two high valene quadrants with differing arousals. There are significant differences in tonality found between all other quadrant pairings. The significant differences in most quadrants suggests that there is a relationship between tonality and perceived sentiment, but it is unclear from these results how that relationship is associated to valence and arousal. However, although there is a significant difference between the mean classes of Q2 and Q3, the two low valence quadrants, there is overlap in the confidence intervals of these mean attribute classes. This suggests that there is a greater difference in tonality between pieces of different valence levels.

VI. Model Retraining

Having developed the new attributes, the next step in the process was to retrain the model to work with the new attributes. Several attempts at this process were made, with varying results.

In training the model, reconstruction loss and Kullback-Leibler divergence loss were measured, both for training data and for validation data. Kullback-Leibler (KL) divergence loss is a metric used to measure the difference between two data distributions. In the context of MuseMorphose, the KL divergence loss measures the difference between the posterior distribution of each bar's encoding as estimated by the encoder, and the prior distribution of each bar's encoding. In the case of MuseMorphose, the prior distribution for each bar's encoding is set to the standard gaussian $N(0, I_{d_z})$, where d_z represents the dimensionality of the latent space, and I_{d_z} represents the identity matrix of that dimensionality (Wu and Yang 2022). The reconstruction loss, meanwhile, is the conditional negative log-likelihood (NLL) for the model to reconstruct the input given the Transformer encoding of the piece and the values of the bar-level attributes.

a. Experiment One

In the first attempt at retraining the model, only the melodic contour and tonality attributes were added to the program. This was because the function for mean pitch height extraction had not yet been developed. The methodology in training the model was adopted to maintain as much similarity as possible to the training process used for the original MuseMorphose, since the model conditioning was undertaken before attributes were added to the model (Wu and Yang 2022). For that reason, the model was trained with the same hyperparameters in place as used on the original model, and it was trained across 60 epochs as a starting point. The EMOPIA dataset was used, as the misunderstanding on how long training with the AILabs.twPop-1k7 dataset would take had not yet been identified. An 80/10/10 train/test/evaluation split was created by randomising the order of samples in the EMOPIA dataset and selecting the first 80% for training, then the next 10% for testing, and storing the remaining 10% for evaluation.

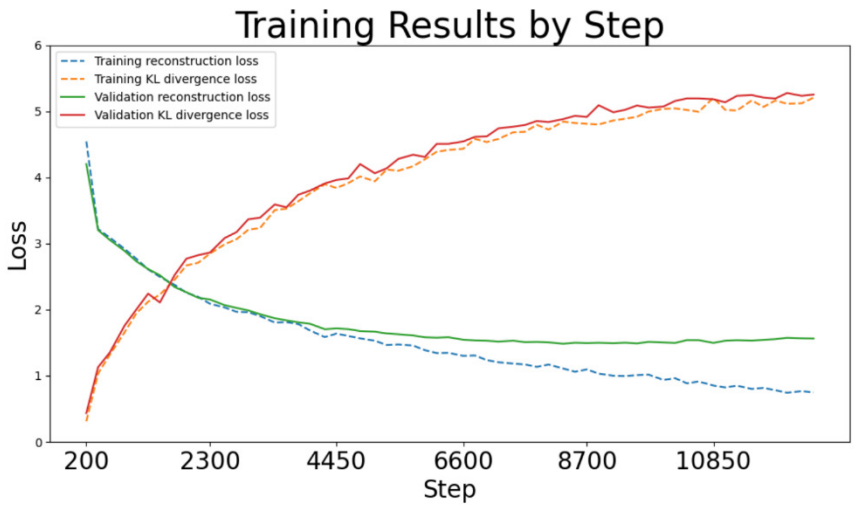

Figure XI: A graph showing the reconstruction loss and KL-divergence loss across training steps.

This graph shows the training and validation loss values from training the model across 60 epochs. The steps marked indicate the last point at which training results were validated in every ten epochs. The graph shows that while the reconstruction loss was consistently descending (with a levelling out in validation at later training steps), the KL divergence loss was almost constantly increasing. These results indicated that the model was not learning how to recreate input pieces. Additionally, the continuing decrease in training reconstruction loss towards the end of training while the validation reconstruction loss becomes constant suggests that the model is starting to overfit to the training data. To attempt to determine a root cause, the next experiment was to retrain the model, replacing all classes for the added attributes with 0.

b. Experiment Two

In the second experiment, almost everything was kept the same, except for setting the values for the melodic contour attribute and the tonality attribute to zero. The purpose of this was to determine whether the cause of the poor results in the previous step was to do with the dataset or the attributes.

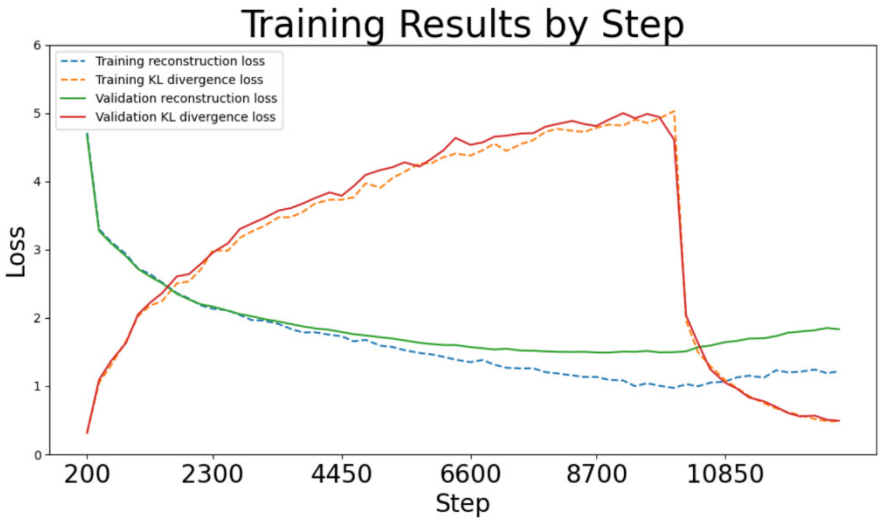

Figure XII: A graph showing the training results from the second training experiment.

Similar to the first experiment, the reconstruction loss initially fell while the KL divergence loss rose. However, in this experiment, the KL divergence loss began rapidly decreasing after 10,000 steps. Meanwhile, after initially falling the reconstruction loss began to slightly increase beyond step 10,000, and the gap between the training and validation reconstruction loss gradually increased. This is indicative of the model beginning to overfit to the training data, and since this is occurring relatively early in the training, it suggests that the EMOPIA dataset does not provide the model with enough varied training data to be able to deduce the underlying patterns in the data.

The results of this experiment seem to indicate that while the initial increase in the KL divergence loss is not caused by the attributes themselves, there is still some issue, since when given the real attribute data the model does not demonstrate the same dramatic decrease in divergence loss after 10,000 steps.

This realisation informed the decision to reassess the attributes being extracted. The largest change resulting from this reassessment was the decision to remove the melodic contour attribute from the model. This decision was based on a number of factors; first, the model had no way of differentiating between melody and non-melody notes, increasing the

complexity in learning trends in the attribute; and secondly, the attribute classification system was not as well defined as the other attributes, since the distinction between attributes with more than one change in direction had been made for the purpose of making the attribute fit into eight classes, instead of being based on any research.

At this stage, the decision to include the pitch height attribute was made, as it was the next best candidate to be an attribute added to the model. Since the model had already been modified to support four attributes instead of two, it was straightforward to replace the contour extraction function and rename contour attributes to reflect the new attribute.

The final change made was due to a mistyped character discovered in the tonality extraction function. The function was written to first give an early return for chords containing “m7,” and to then return a class of 6 if an “m” was present and the function was still active. However, instead of returning a 6 if “m” was present, the function was returning 6 if “7” was present. Due to the structure of the function, all chords containing a “7” were filtered out if the chord reached this stage; therefore, a class of 6 was never being returned for this attribute. Once this error had been identified, it was very simple to fix.

c. Experiment Three

By the time the changes recognised as necessary from the second experiment had been implemented, the misunderstanding regarding which dataset took 2 days to complete 20 epochs had been resolved. Therefore, to avoid the apparent overfitting issues from the previous training experiments, the training dataset was changed from EMOPIA to AILabs.tw-Pop1k7. While this significantly increased the amount of time to complete 60 epochs of training, the training still took substantially less than the anticipated 6 days. Since the AILabs.tw-Pop1k7 dataset contains more samples than the EMOPIA dataset (1747 as opposed to 1078), more training steps were required to complete 60 epochs of training.

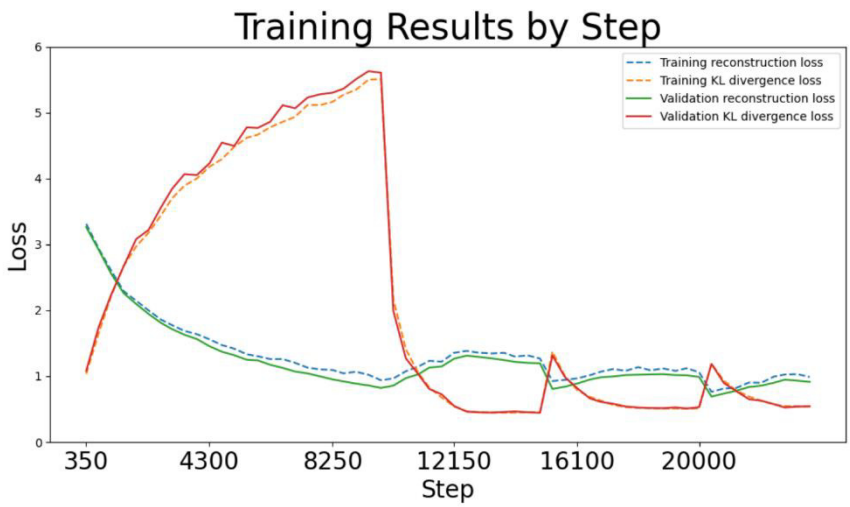

Figure XIII: A graph showing the training results from the third training experiment

In this training experiment, the same initial trend emerged: the KL divergence loss increased until training step 10,000, and then rapidly decreased. The same pattern of the reconstruction loss initially decreasing before increasing slightly is also present; however, there is not a significant increase in the difference between the training and validation results for the reconstruction loss.

A pattern seems to emerge in further training steps: both losses remain fairly constant for a number of steps, then the KL divergence sharply increases as the reconstruction loss sharply decreases, before both values begin to gradually decrease and increase respectively. The first time this occurs, at step 15090, the KL training divergence increases to 2.066 (to 4 significant figures), and the reconstruction loss has decreased to 0.9286 (to four significant figures). The second time this occurs, at step 20043, the KL training divergence increases to 2.355 (to 4 significant figures), while the reconstruction loss has decreased to 0.8347 (to 4 significant figures). The training ended before the third time this pattern could occur, however depending on how this pattern continues, there may be better results if training continues over further steps. The pattern seems to occur every 5000 steps; therefore, to see if it continues to be true, training was carried out for a further 16000 steps. This was so that the

pattern would occur a further three times (if it remained consistent), and to allow trailing steps, in order to determine if the pattern continues further. Since each epoch lasts for 393 steps, training for a further 16000 steps would take 40.71 epochs. This was rounded up to 41 epochs.

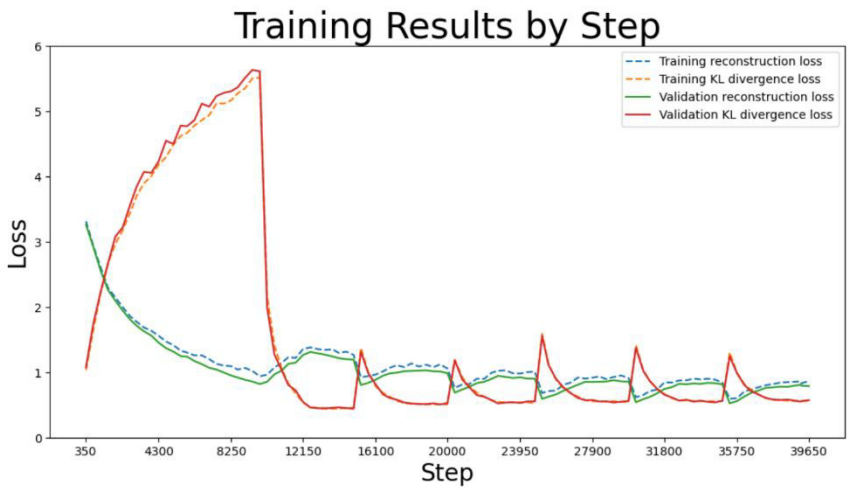

Figure XIV: A graph showing training results from training for a further 41 epochs

After training for a further 41 epochs, it is apparent that the pattern noticed was indeed continuing. It also became clear that there was an overall decrease in the reconstruction loss, while the lowest point to which the KL divergence loss was returning in between peaks was remaining largely consistent. From every validation point across the training, the checkpoint at which the sum of the KL divergence loss and the reconstruction loss was lowest was at step 39300, with a reconstruction loss of 0.8084 and a KL divergence loss of 0.5728. For that reason, this checkpoint is the one that will be used to generate output in the next stage of the project.

While it seems likely from the trends in the training results that further training would yield better results, after training for 101 epochs, the cost of time and computational resources to the benefit of reduced loss is diminishing.

VII. Output Generation

For output generation, changes to the structure of the generation program need to be made; this is partially discussed in Section 4. Further to these changes, methodology for generation of any sentiment from any input sentiment needs to be implemented. To target a specific sentiment, the mean attribute classes for different sentiments will be used.

To determine the offsets for each input and target sentiment pair, the difference between attribute means where a significant difference was found is rounded up to the nearest whole number. The decision to round up the difference was informed by the number of attributes where the differences between mean attribute classes were either predominantly or entirely below 1. For each attribute, this information was organised into a two-dimensional array. The following table represents this data.

	Q1	Q2	Q3	Q4
Rhythmic Intensity	To Q2: +1 To Q3: -1 To Q4: -1	To Q1: -1 To Q3: -2 To Q4: -2	To Q1: +1 To Q2: +2 To Q4: 0	To Q1: +1 To Q2: +2 To Q3: 0
Polyphonicity	To Q2: +1 To Q3: -1 To Q4: -1	To Q1: -1 To Q3: -2 To Q4: -1	To Q1: +1 To Q2: +2 To Q4: 0	To Q1: +1 To Q2: +1 To Q3: 0
Pitch height	To Q2: 0 To Q3: -1 To Q4: -1	To Q1: 0 To Q3: -1 To Q4: -1	To Q1: +1 To Q2: +1 To Q4: 0	To Q1: +1 To Q2: +1 To Q3: 0
Tonality	To Q2: +1 To Q3: +1 To Q4: 0	To Q1: -1 To Q3: -1 To Q4: -1	To Q1: -1 To Q2: +1 To Q4: -1	To Q1: 0 To Q2: +1 To Q3: +1

Table XIII: Attribute offset amounts for each attribute when transferring from each input sentiment.

For each sample being generated, the type of transfer is determined from the input and target sentiment, and the relevant attribute offset amounts added to each bar-level attribute for the input data. The initial intention, since the AILabs.tw-Pop17k dataset was used in training, was to use the full EMOPIA dataset to generate 3192 sentiment-transferred sample (with three new samples generated for each of the 1064 samples in the EMOPIA REMI dataset), however for each sample in the dataset, on a V100 Tesla GPU, it takes

approximately 40 seconds to generate three sentiment-transferred pieces of symbolic music, resulting in a total time of around 12 hours. Therefore, only a subset of the EMOPIA REMI dataset will be used. To exceed the 1064 samples in the original dataset, 356 samples will be selected from the EMOPIA dataset, with 89 samples randomly selected from each input quadrant. This reduces the generation time to slightly under 4 hours.

In the following section, the results from this modified generation will be discussed.

5. Results and Discussion

I. Results

a. Results from First Generation Experiment

Having decided to use 89 samples for each quadrant to generate output, the generation program was updated accordingly. First, 89 samples were randomly selected from each input quadrant in the EMOPIA dataset. The ID from each piece was determined by extracting the integer ID from the filename. Then for each sample selected, output was generated modifying that sample to match each other sentiment quadrant in Russell's four quadrants.

Once three samples for each input piece had been generated, all generated pieces were used as input to the EMOPIA classification tool, using the pre-trained weights available as a supplemental download in the Github repository (Hung et al. 2021). This was called repeatedly using the Python subprocess module. The number of successes for each transfer type was stored as a value in a dictionary to be used for analysis. To determine the success rates for each quadrant, a percentage of successes was calculated alongside a ratio of successes to total number of pieces in each transfer category. A few drawbacks from this attempt became apparent during this processing. First, the input pieces are not sorted into which quadrant they belong to. This means there is no way to determine whether the classification tool achieves different success rates in different quadrants. Second, and more pressingly, several input samples had their quadrants mislabelled during the generation

process.

For the purposes of this project, files in the REMI EMOPIA dataset were renamed to contain the piece's quadrant and a unique ID number. For example, the first piece in the dataset when sorted alphabetically is 'Q1_0001.pkl'. By taking the ID as an integer value, there were cases where searching for the piece by ID would return multiple results; for instance, the piece with ID 1 would return matches for any piece with a '1' at any position in its 4-digit ID. This error resulted in many samples being incorrectly labelled, and therefore the attribute offsets being inaccurate for the real input sentiment and target sentiment. The results obtained from this experiment, therefore, cannot be considered accurate.

The table below shows the results from this experiment, except for accuracy in determining the quadrant of unmodified samples, as that data was not captured per quadrant in this experiment. Across all unmodified samples, 209/356 pieces, or 58.71%, were identified as matching the quadrant in their input string. However, it is worth bearing in mind that due to the previously described error, an unknown number of these samples were mislabelled. Since Q1 and Q2 samples have lower IDs, they are more likely to be mislabelled; however, since Q3 and Q4 are both as likely as the other quadrants to contain mislabelled samples, they cannot be considered more accurate.

Input quadrant	Q1	Q2	Q3	Q4
Output quadrant				
Q1	N/A	21/101 20.79%	12/98 12.24%	17/102 16.67%
Q2	25/55 45.45%	N/A	21/98 21.43%	9/102 8.824%
Q3	3/55 5.455%	14/101 13.86%	N/A	43/102 42.16%
Q4	0/55 0%	3/101 2.970%	11/98 11.22%	N/A

Table XIV: Number of successful sentiment categorisations and percentage accuracies for each transfer type.

Before the error in sentiment labelling was realised, a few features about these results were noticed. First, the areas with the best transfer results are Q1 to Q2; Q4 to Q3; and to a lesser degree, Q2 to Q1. These are all transfers where only valence is changing. The worst results are seen in Q1 to Q4, Q2 to Q4, and Q1 to Q3. These are all high arousal to low arousal transfers.

On reviewing the results of attribute analysis, it was found that while there was a significant difference between Q2 and Q3 in average tonality attribute class, they were the two attributes closest to each other, and there was no significant difference between Q1 and Q4. This supports the findings of Panda et al. that changes in tonality are related to changes in perceived valence (2023). Based on this observation, and the improved accuracy when transferring by valence rather than arousal, the tonality attribute was fully removed to see

how arousal transfers were affected.

Given the flaws in sentiment labelling for this experiment, the results will not be further discussed.

b. Results from Second Generation Experiment

For the second experiment, the error in sentiment labelling was fixed by storing the piece ID as a four-character string. This ensured that the correct piece was identified, so therefore the correct quadrant was identified. In addition, the tonality attribute was removed from consideration, to see how transfers by arousal were affected. The table below shows the results from this experiment.

Input quadrant	Q1	Q2	Q3	Q4
Output quadrant				
Q1	54 60.67%	19 21.35%	12 13.48%	17 19.10%
Q2	39 43.82%	62 69.66%	16 17.98%	4 4.494%
Q3	7 7.865%	12 13.48%	60 67.41%	24 26.97%
Q4	0 0.0%	3 3.371%	11 12.36%	53 59.55%

Table XV: Number of successful sentiment categorisations and percentage accuracies for each transfer type.

These results again show lower accuracy in all categorisation tasks where a transfer has taken place, and the most accurate results in transferring Q1 to Q2, Q4 to Q3, and Q2 to

Q1, which are all cases where sentiment is being transferred only by valence and not by arousal. The least accurate results are found in transferring Q1 to Q4, Q2 to Q4, and Q4 to Q2. These are all cases where sentiment is being transferred by arousal, although valence is sometimes also a factor. Another similarity between these results is that they all involve transfer to or from Q4.

Given the errors in quadrant labelling in the previous experiment, the impact of removing the tonality attribute cannot be assessed. Therefore, it was restored to the program for the next round of experiments. However, further consideration of the attribute led to a decision to change how it was transferred. The table below shows the quality of chords along with their class.

Tonality Class	Chord Quality	Pitch class set
0	Augmented	0, 4, 8
1	Major or Major 7	0, 4, 7, optionally 11
2	Dominant 7	0, 4, 7, 10
3	Sustained	0, 5, 7 or 0, 2, 7
4	No chord identified	Unknown
5	Minor 7	0, 3, 7, 10
6	Minor	0, 3, 7
7	Diminished	0, 3, 6

Table XVI: Tonality class, corresponding chord quality, and pitch class set

In the tonality attribute, the scale moves from brightest (lowest class) to darkest (highest class), with the central classes being relatively neutral. However, in generation, the tonality class is modified by +/- 1-2 attribute classes without consideration of the change in tonality when moving from one side of the scale to the other.

Given this observation, combined with the findings of Panda et al. that pieces with a major tonality are perceived as high valence while pieces with a minor tonality are perceived as low valence (2023), an alternative method is adopted for changes in the tonality attribute. If there is a change in valence between the input sentiment and the target sentiment, the tonality attribute values are reflected around the centre of the scale; for instance, class 0 would become class 7, and class 5 would become class 2. This is achieved by taking the absolute value of subtracting 7 from each bar-level attribute. This is not a perfect approach, as not all high valence pieces are in a major tonality, and not all minor valence pieces are in a minor tonality; however, since the overall tonality of a piece is not known while carrying out generation, it isn't possible to differentiate by that factor.

c. Results from Third Generation Experiment

In the final attempt at generating output, the described updated method for modifying the tonality attribute was implemented. Additionally, given the poor performance in arousal transfer, the attribute offset for the remaining attributes was increased by a factor of 2. Rather than manually changing all offset amounts, an optional factor argument was added to the function to set attribute offsets with a default value of 1; all return values would be multiplied by this value. This value was set to 2 in all cases for this generation.

With these changes made, a further 1068 samples were generated and classified.

Input quadrant	Q1	Q2	Q3	Q4
Output quadrant				
Q1	55 (+1) 61.80%	25 (+6) 28.09%	11 (+1) 12.36%	11 (+6) 12.36%
Q2	44 (+5) 49.44%	58 (-4) 65.17%%	17 (+1) 19.10%	7 (+3) 7.865%
Q3	12 (+5) 13.48%	19 (+7) 21.35%	61 (+1) 68.54%	38 (+14) 42.70%
Q4	5 (+5) 5.618%	1 (-2) 1.124%	12 (+1) 13.48%	52 (-1) 58.43%

Table XVII: Number of successful sentiment categorisations and percentage accuracies for each transfer type.

These results show an increase in success rates across many transfer categories, as well as a decrease in success rate in one category. The largest increases in success rate can be observed in transferring from Q4 to Q3, although increases in accuracy can be observed across all transfer categories except for transferring from Q2 to Q4. The largest increases in success rates are all observed where either valence or arousal transfer is occurring.

The sum of increased positive categorisations in transfers only by valence was 26, while the sum of increased positive categorisations in transfers only by arousal was 19. This indicates that transfer by arousal had increased accuracy, but not to the same extent as the increase in transfer by valence.

II. Discussion

a. Second Generation Experiment

From the results of the second experiment, there are no cases where the accuracy reaches the same accuracy seen when classifying real data. However, there are a few transfers which show promising results. For example, a Q2 sample generated from a Q1 input was correctly categorised 39/81 times, or 43.82% of the time. Considering that there are four possible quadrants which an input could be sorted into, an accuracy of 43.82% is significantly higher than a random chance, or 25%. The only other transfer type to achieve an accuracy greater than 25% is transferring Q4 to Q3, In this case, generated samples were correctly classified 24/89 times, or 26.97% of the time. Given that this is fairly close to 25%, it is possible that further tests would reveal that this quadrant is not always correctly categorised this often. However, due to having limited computational resources while working on this project, further runs of the same experiment were not possible.

It is worth noting that for these two categories (Q4 to Q3 and Q1 to Q2), these are both cases where valence is going from high to low, while arousal is remaining constant. This suggests that the generation method used in the experiment has potential to successfully achieve sentiment transfer when going from high valence to low valence.

The remaining accuracies across different transfer types show varying degrees of success, although none exceed 25%. There are a few which show extremely low accuracy. The worst is transfer from Q1 to Q4, in which no generated samples were correctly categorised. Similarly poor results were observed in transferring from Q4 to Q3 (4/89 successes, or 4.494%), and Q2 to Q4 (3/89, or 3.371%). Q2 to Q4 and Q1 to Q4 are both quadrants which involve a decrease in arousal, while Q4 to Q3 involves a decrease in valence. Looking at other transfers where arousal is decreasing (Q1 to Q3 and Q2 to Q3), results are still low (7/89 and 12/89 respectively), although not quite as low as those

transferring into Q4. Transferring from Q3 into Q4 results in similarly low accuracy (11/89, or 12.36%), which suggests that the current generation method performs poorly at generating Q4 samples. Alternatively, it may be that the poor performance is due to the model not successfully producing low arousal samples, which is only mitigated in transferring Q4 to Q3 because the model is successful in increasing valence.

b. Third Generation Experiment

In the third generation experiment, accuracy in classifying generated samples increases in almost every type of generation.

The largest increase in accuracy was seen in transferring from Q4 to Q3, with 38/89 successes for an accuracy of 42.70%. Increases in accuracy were seen in all cases of transfer by valence, which suggests that mirroring each tonality class around the centre of the list of possible classes is contributing to successfully transferring by valence. In every case of transfer only by valence except for transfer from Q3 to Q4, accuracy rose above 25%, indicating slight success in valence transfer.

Accuracies when going from low valence to high valence (49.44% and 42.70%) remain higher than accuracies found when going from high valence to low valence (28.09% and 13.48%), suggesting that the model is better at transferring by increasing valence than decreasing valence. One possible reason for this remaining true, even with the introduction of changes in tonality attribute classes, is the difference in harmonic complexity in high valence and low valence music. Panda et al. found in their review of musical attributes that more complex harmonies were associated more with low valence sentiments (2023). Complex harmonies are more likely to contain a wider variety of tonality classes, therefore low valence pieces may be more likely to contain a wide variety of tonality classes than high valence pieces, which could be a reason for this discrepancy in accuracy depending on direction of valence transfers.

Increases were also observed in transfers where arousal was changed, with the largest increase found in transferring from Q2 to Q3, with 19/89 samples being accurately categorised, or 21.35%. However, all transfers involving arousal still had a lower accuracy than 25%.

The increases in accuracy from the changes made to the generation model suggest that there is potential for sentiment transfer through attribute control, although further work would be required to achieve meaningful results.

6. Conclusion

This dissertation set out to answer the question: can symbolic music sentiment transfer across Russell's four quadrants be achieved through musical attribute extraction and control? The approach to answering this question was to modify an existing model for music generation by attribute control, MuseMorphose, by adding attributes associated with changes in perceived sentiment.

Once methods for extracting these attributes were written, the model was updated to be able to handle these attributes, and then retrained across 101 epochs. Three retraining experiments were undertaken. First the model was retrained to accept a melodic contour attribute and a pitch height attribute and retrained using the EMOPIA dataset. However, results indicated that the model was not learning how to model these attributes. To determine whether the issue was the dataset or the attributes, a second training experiment was carried out with all attribute classes in the two added attributes set to zero. The model achieved better results in this experiment, indicating that the issue likely was with the attributes being extracted. However, in later training stages the model was showing a decrease in reconstruction loss in training but a constant reconstruction loss in validation. Therefore, in the final model training experiment, the melodic contour attribute was removed due to its high level of abstraction, and replaced with a tonality attribute which provided a way to

differentiate between high valence and low valence music, and the EMOPIA dataset was replaced with the AILabs.tw-Pop17k dataset. The model was trained over 101 epochs, and stopped as decreases in loss over time began to plateau.

Finally, output was generated and fed into the EMOPIA sentiment classification tool (Hung et al. 2021). This was also done over three distinct experiments. The first experiment was of limited use, due to an error in labelling sentiment of input samples. The second experiment removed changes in tonality classes from generation to see if there was an impact on accuracy in arousal transfer. In the final experiment, a new method for transferring tonality attribute classes was introduced, and the amounts by which other attributes were offset was increased. Results from this final experiment indicated increased accuracy in classifying generated music by sentiment, although further work is required to achieve high accuracy sentiment transfer.

Although this dissertation did not result in a program which could consistently transfer music across Russell's four quadrants, the increasing accuracy in classification through repeated experiments suggest that sentiment transfer through attribute control has potential to be successful and warrants further research. Some possible approaches may include implementing methods to change the distribution of attributes classes across an entire song; testing offsetting attributes by different amounts; and attempting to achieve lower reconstruction loss and KL loss values. This dissertation also does not assess the quality of generated samples; future work to assess this parameter, and to try to improve the quality if necessary, will also be beneficial.

7. Learning Reflection

This dissertation has been a substantial undertaking, and I have learned a great deal about both machine learning and project management.

When I chose this dissertation topic, I understood very little about machine learning

models and methods. In the months leading up to beginning the dissertation I began some background reading about transformers and VAEs, so that when I began the work in earnest I would have at least some understanding of the programs I was working with. However, this background reading only began to feed into concrete understanding once I began working on the dissertation and engaging with the MuseMorphose model.

Throughout the project, I found myself regularly running into problems, finding the solution was far more straightforward than I had thought it was, and cursing myself for not realising the issues sooner! This was particularly the case when it came to tasks which required a GPU, such as model training and symbolic music generation (which, while not strictly requiring a GPU, took far less time to complete when one was used). It was the case, more often than I would have liked, that a minor oversight in the code would not be realised until several hours of training or generation had been completed. This was a valuable process to go through, despite the frustrations it caused, as it has taught me the importance of thoroughly checking and double-checking code before running it - especially when it involves computationally heavy processes, or when it is overwriting a file. I have also developed a deeper appreciation of the value of version control, to save me from irreparable file changes that go wrong.

Another major takeaway from this project has been the importance of good time management. I owe both of my supervisors a great debt of gratitude for being willing to meet regularly, as these regular meetings gave me an incentive to ensure I was making enough progress to discuss from week to week. Before beginning the project I had outlined steps I would take in developing the program, and approximate amounts of time to allow to each of these steps, but it often took having my supervisors telling me that it was time to move on before I would step away from a section where I knew there was more I could do, even once time allowed for it had passed.

Working on a large project like this has involved a less linear timeline than I had anticipated. Before beginning the project, I had planned allotted timelines for each stage of the process. However, as I was working on the project, it became clear that it wouldn't be a case of moving neatly from one stage to the next. Instead, it would be more a case of moving through several steps, making observations about the project, what worked and what didn't, then stepping back considerably and taking another run at it. I had to learn to open myself up to be more flexible to changes in my planned timeline for the project.

This lesson was really driven home after I encountered a major setback in the project. I became quite unwell, and for close to a month I was unable to make meaningful progress on the project. I was quite fortunate that this came after I had completed the bulk of the programming itself, however it was still incredibly frustrating. However, despite the added complexity presented by this situation, it was still an opportunity for learning. Having to push back my schedule by a substantial amount meant that I had to accept that I wouldn't meet the internal project deadlines I'd set myself, which was something I struggled with initially. This setback caused me quite a bit of uncertainty, and for a few weeks I wasn't confident that I would be able to complete the project on time, even once an extension had been granted. However, with the continued encouragement of my supervisor, I continued to make progress through the project and those uncertainties waned.

Overall, throughout this project I feel I have learned a great deal about both computing and project management. These are skills that I am confident I will be able to carry forward with me as I engage with new projects and challenges.

References

Abboud, R. and Tekli, J. 2020. Integration of nonparametric fuzzy classification with an evolutionary-developmental framework to perform music sentiment-based analysis and composition. Soft Computing 24(13), pp. 9875–9925. doi: 10.1007/s00500-019-04503-4.

Ax, A.F. 1953. The Physiological Differentiation between Fear and Anger in Humans: Psychosomatic Medicine 15(5), pp. 433–442. doi: 10.1097/00006842-195309000-00007.

Bogdanov, D. 2019. The MTG-Jamendo Dataset for Automatic Music Tagging.

Chou, Y.-H., Chen, I.-C., Chang, C.-J., Ching, J. and Yang, Y.-H. 2021. MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding. Available at: http://arxiv.org/abs/2107.05223 [Accessed: 25 July 2023].

Cífka, O., Ozerov, A., Şimşekli, U. and Richard, G. 2021. Self-Supervised VQ-VAE for One-Shot Music Style Transfer. In: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 96–100. Available at: http://arxiv.org/abs/2102.05749 [Accessed: 1 November 2023].

Cífka, O., Şimşekli, U. and Richard, G. 2020. Groove2Groove: One-Shot Music Style Transfer With Supervision From Synthetic Data. IEEE/ACM Transactions on Audio, Speech and Language Processing 28, pp. 2638–2650. doi: 10.1109/TASLP.2020.3019642.

Dong, H.-W. and Yang, Y.-H. 2018. Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation. Available at: http://arxiv.org/abs/1804.09399 [Accessed: 21 December 2023].

Eerola, T., Lartillot, O. and Toiviainen, P. 2009. Prediction of Multidimensional Emotional Ratings in Music from Audio Using Multivariate Regression Models. pp. 621–626.

Ferreira, L.N. and Whitehead, J. 2021. Learning to Generate Music With Sentiment. Available at: http://arxiv.org/abs/2103.06125 [Accessed: 13 August 2023].

Goldstein, M., Granot, R., Ripolles, P. and Farbood, M.M. 2023. Exploring Melodic Contour: A Clustering Approach. Available at: https://osf.io/pkfwx/ [Accessed: 8 November 2023].

Grekow, J. 2023. Generating Polyphonic Symbolic Emotional Music in the Style of Bach Using Convolutional Conditional Variational Autoencoder. IEEE Access 11, pp. 93019–93031. doi: 10.1109/ACCESS.2023.3309639.

Han, X., Chen, F. and Ban, J. 2023. Music Emotion Recognition Based on a Neural Network with an Inception-GRU Residual Structure. Electronics 12(4), p. 978. doi: 10.3390/electronics12040978.

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C. and Yang, Y.-H. 2021. Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs. Available at: http://arxiv.org/abs/2101.02402 [Accessed: 30 November 2023].

Huang, Y.-S. and Yang, Y.-H. 2020. Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions. Available at:

http://arxiv.org/abs/2002.00212 [Accessed: 23 October 2023].

Hung, H.-T., Ching, J., Doh, S., Kim, N., Nam, J. and Yang, Y.-H. 2021. EMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-based Music Generation. Available at: http://arxiv.org/abs/2108.01374 [Accessed: 25 July 2023].

Kalinak, K. 2010. Film Music: A Very Short Introduction. Oxford: Oxford University Press. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=311723&site=ehost-live&scope=site [Accessed: 12 August 2023].

Kong, Q., Li, B., Song, X., Wan, Y. and Wang, Y. 2021. High-resolution Piano Transcription with Pedals by Regressing Onset and Offset Times. Available at: http://arxiv.org/abs/2010.01815 [Accessed: 3 November 2023].

Lane, D., Haney, S. and Joshi, S. 2021. DearDiary.ai. Available at: https://magenta.tensorflow.org/dear-diary [Accessed: 23 December 2023].

Panda, R., Malheiro, R. and Paiva, R.P. 2023. Audio Features for Music Emotion Recognition: A Survey. IEEE Transactions on Affective Computing 14(1), pp. 68–88. doi: 10.1109/TAFFC.2020.3032373.

Persichetti, V. 1961. Twentieth-Century Harmony: Creative Aspects and Practice. New York, NY, US: W. W. Norton & Company.

Russell, J. 1980. A Circumplex Model of Affect. Journal of Personality and Social Psychology 39, pp. 1161–1178. doi: 10.1037/h0077714.

Scruton, R. 1999. Representation. In: Scruton, R. ed. The Aesthetics of Music. Oxford University Press, p. 0. Available at: https://doi.org/10.1093/019816727X.003.0005 [Accessed: 23 December 2023].

Sigel, M., Zhou, M. and Luo, J. 2021. Music Sentiment Transfer. Available at: http://arxiv.org/abs/2110.05765 [Accessed: 12 August 2023].

Uitdenbogerd, A. and Zobel, J. 2002. Manipulation of Music For Melody Matching. Proceedings of the 6th ACM International Conference on Multimedia, MULTIMEDIA 1998. doi: 10.1145/290747.290776.

Vaswani, A. et al. 2023. Attention Is All You Need. Available at: http://arxiv.org/abs/1706.03762 [Accessed: 26 July 2023].

Wu, S.-L. and Yang, Y.-H. 2022. MuseMorphose: Full-Song and Fine-Grained Piano Music Style Transfer with One Transformer VAE. Available at: http://arxiv.org/abs/2105.04090 [Accessed: 25 July 2023].

Yang, Y.-H. and Chen, H.H. 2012. Machine Recognition of Music Emotion: A Review. ACM Transactions on Intelligent Systems and Technology 3(3), p. 40:1-40:30. doi: 10.1145/2168752.2168754.

Zhu, J.-Y., Park, T., Isola, P. and Efros, A.A. 2020. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. Available at: http://arxiv.org/abs/1703.10593

[Accessed: 1 November 2023].